UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Engineering and Information Technology

Development of Robust and Scalable Hyperbox based Machine Learning Algorithms

by

Thanh Tung KHUAT

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Sydney, Australia

2021

Certificate of Original Authorship

I, Thanh Tung KHUAT, declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Computer Science, Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature:

Production Note:
Signature removed prior to publication.

Date: 01 July 2021

Abstract

Together with the rapid development of digital information and the increase in amount of data, machine learning (ML) algorithms have been developed and evolved constantly to discover new information and knowledge from different data sources. The use of hyperbox fuzzy sets as fundamental representational and building blocks in learning algorithms forms an important branch of ML. Hyperbox-based algorithms have a huge potential for high scalability and incremental adaptation to applications working in the dynamically changing environments. Additionally, learning algorithms based on hyperbox representations can form interpretable models, which are highly desirable for areas with the requirement of safety and trust. This study aims to develop and expand robust, scalable, and transparent learning algorithms for hyperbox-based classification models with a specific focus on a general fuzzy min-max neural network (GFMMNN).

First of all, a comprehensive survey on hyperbox-based machine learning models together with empirical assessments of the GFMMNN on pattern classification problems were conducted. Next, a new online learning algorithm was proposed for the GFMMNN and improved the robustness of the whole family of GFMMNN learning algorithms to work effectively with mixed-attribute data by introducing a new learning mechanism for categorical features. In terms of scalability, the main steps of the learning algorithms were reformulated so they can be effectively executed on graphics processing units using matrix operations, simultaneously proposing mathematical lemmas to reduce the redundancies of hyperbox candidates in the learning process. This thesis also proposed a novel method to enhance the transparency of classifiers while maintaining a good classification performance by using hierarchical granular representations from hyperbox fuzzy sets. The last contribution was a simple but powerful ensemble model built from many individual hyperbox-based classifiers trained on random subsets of both sample and feature spaces. Extensive empirical analyses indicated that the proposed solutions are highly competitive with other evaluated learning algorithms.

To my loved ones

Acknowledgments

The Ph.D. study is a long journey, and this thesis would not have been possible without the support and encouragement of my supervisors, friends, and relatives during my Ph.D. research journey. Maybe the hardest part when I write this thesis is how to express my sincere gratitude to them.

First and foremost, I would especially like to express my sincerest gratitude to my principal supervisor, Professor Bogdan Gabrys for his continuous support, motivation, enthusiasm, inspiration to my Ph.D. study, together with his invaluable advice and discussion during our weekly meetings. His guidance is likely to positively affect me throughout my future career path. He will definitely be the first person to whom I seek advice for the difficult research problems that I have to deal with in the future. My Ph.D. research has faced a difficult time because of the spread of the COVID-19 pandemic. However, under his supervision and instructions, I has been quick to change research plans and adapt to the new ways of working using remote working tools. As a result, my research progress has not been negatively impacted by the pandemic. Those are precious experiences that will never be forgotten.

I would also like to thank my co-supervisors, Distinguished Professor Fang Chen and Dr. Dymitr Ruta for their valuable comments and discussion on the research manuscripts that I have written for my Ph.D. projects. My thanks also go to my Ph.D. assessment panel, Professor Paul Kennedy and Associate Professor Wei Liu, for their constructive feedback. Special thanks also to Dr. Hanh Le for encouraging me to pursuit Ph.D. research. I wish to thank UTS for awarding me scholarships so that I can conduct my Ph.D. project.

A large part of this thesis comes from publications which have been peer-reviewed by anonymous reviewers. Therefore, I also wish to send a special thank to them who made valuable suggestions to enhance the quality of my research papers.

Additionally, I would like to thank all my best friends who have accompanied me throughout my Ph.D. journey: Tien-Dung Nguyen and his family, Thong Do, Thac Do, Cong Nguyen, Tung Huynh, An Le, Dung Duong, Xuan Yang, Joakim Skarding, Sunny Verma, and many others. Happy conversations and all the fun we have had during the past three years will be unforgotten memories in my life.

Finally, I would like to express my love and boundless gratitude to my parents and younger sister for their unconditional support and sacrifice. They have supported me both financially and emotionally so that I can completely concentrate on my Ph.D. research.

Thanh Tung Khuat

Sydney, Australia, 2021.

List of Publications

Published Papers

1. **Thanh Tung Khuat**, and Bogdan Gabrys, "Random hyperboxes," *IEEE Transactions on Neural Networks and Learning Systems* (Early Access). (*Chapter 8*)
2. **Thanh Tung Khuat**, and Bogdan Gabrys, "An in-depth comparison of methods handling mixed-attribute data for general fuzzy min-max neural network," *Neurocomputing*, vol. 464, pp. 175-202, 2021. (*Chapter 5*)
3. **Thanh Tung Khuat**, Fang Chen, and Bogdan Gabrys, "An effective multi-resolution hierarchical granular representation based classifier using general fuzzy min-max neural network," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 2, pp. 427- 441, 2021 (**IEEE CIM Publication Spotlight paper**). (*Chapter 7*)
4. **Thanh Tung Khuat**, and Bogdan Gabrys, "Accelerated learning algorithms of general fuzzy min-max neural network using a novel hyperbox selection rule," *Information Sciences*, vol. 547, pp. 887-909, 2021. (*Chapter 6*)
5. **Thanh Tung Khuat**, Dymitr Ruta, and Bogdan Gabrys, "Hyperbox based machine learning algorithms: A comprehensive survey," *Soft Computing*, vol. 25, pp. 1325–1363, 2021. (*Chapter 2*)
6. **Thanh Tung Khuat**, and Bogdan Gabrys, "A comparative study of general fuzzy min-max neural networks for pattern classification problems," *Neurocomputing*, vol. 386, pp. 110-125, 2020. (*Chapter 3*)
7. **Thanh Tung Khuat**, Fang Chen, and Bogdan Gabrys, "An improved online learning algorithm for general fuzzy min-max neural network," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1-9,

2020 (**IEEE CIS Outstanding Student Paper Conference Registration Grant**). (*Chapter 4*)

8. **Thanh Tung Khuat**, and Bogdan Gabrys, "Accelerated training algorithms of general fuzzy min-max neural network using gpu for very high dimensional data," in *Proceedings of the 26th International Conference on Neural Information Processing (ICONIP)*, pp. 583-595, 2019 (**Best paper award finalists**). (*Chapter 6*)

Under Reviewed Papers

9 **Thanh Tung Khuat**, and Bogdan Gabrys, "An online learning algorithm for a neuro-fuzzy classifier with mixed-attribute data," Submitted to *IEEE Transactions on Fuzzy Systems* (Revised and Resubmit). (*Chapter 5*)

Contents

List of Figures

List of Tables

List of Abbreviations

2lv-CSWCE	Two-level classification system with testing in dynamically changing environment
2-pHC	Two-phase hyperbox classifier
ACO	Ant colony optimization
Adaboost	Adaptive boosting
AGGLO-SM	Agglomerative learning algorithm using the full similarity matrix
AGGLO-2	Agglomerative algorithm version 2
ANNs	Artificial Neural Networks
ARC	Adaptive resolution classifier
ART	Adaptive Resonance Theory
AUC	Area under the curve
CACL	Concept adapting contraction less
CBA	Class balanced accuracy
CD	Critical difference
CLFMNN	Contraction-less fuzzy min-max neural network
CNs	Compensatory Neurons
DCFMN	Data core based fuzzy min–max neural network
DPS	Density-preserving sampling
ECTs	Ensemble model of clustering trees
EFMNN	Enhanced fuzzy min-max neural network
EFMNN-ACO	Enhanced fuzzy min–max neural network with ant colony optimization
EFMMDT	Evolving fuzzy min–max decision tree
EFMNN-II	Enhanced fuzzy min-max neural network with K-nearest hyperbox expansion rule and pruning
EFMNN-UL	Evolved fuzzy min-max neural network for unknown labeled data
EIOL-GFMM	Extended improved online learning algorithm for general fuzzy min-max neural network
EMILP	A enhanced version of mixed integer linear programming model-based hyperbox classifier
Esb-GFMMNN	Ensemble of neuro-fuzzy classifiers
Ens-IOL-GFMM(DL)	Ensemble of GFMMNNs using the IOL-GFMM algorithm at the decision level
Ens-IOL-GFMM(ML)	Ensemble of GFMMNNs using the IOL-GFMM algorithm at the model level
FMCN	Fuzzy min-max neural network with compensatory neurons
FMM-CART	Offline and online fuzzy min–max neural network and classification and regression trees
FMM-CT	Fuzzy min-max clustering neural network with the clustering tree
FMM-GA	Fuzzy min-max neural network with genetic algorithms
FMMDT	Fuzzy min–max neural network based decision tree
FMM-PSO	Fuzzy min-max neural network with the particle swarm optimization
FMM-ECT	Fuzzy Min–Max neural network with an ensemble of clustering trees
FMNWSM	Fuzzy min-max neural network with symmetric margin
FMNN	Fuzzy min-max neural network for classification
FMNN-clu	Fuzzy min-max neural network for clustering
FPGAs	Field-programmable gate arrays

GA	Genetic algorithms
GFMM	General fuzzy min-max
GFMMNN	General fuzzy min-max neural network
GFMMNN-CD1	General fuzzy min–max neural networks for categorical data
GFMMNN-CD2	Enhanced general fuzzy min–max neural networks for categorical data
GNB	Gaussian Naive Bayes
GPU	Graphics Processing Unit
GRFMN	General reflex fuzzy min-max neural
HACO	Hyperbox based clustering with ant colony optimization
HACO2	Hyperbox classifier with ant colony optimization
HFC	Hyperbox fuzzy classifier
HNN	Hyperbox neural network algorithm
IEFCN	Inclusion/exclusion fuzzy hyperbox classification network
IGs	Information granules
IOL-GFMM	Improved online learning algorithm for general fuzzy min-max neural network
KNEFMNN	Enhanced fuzzy min-max neural network with K-nearest hyperbox expansion rule
KNN	K-nearest neighbors
LightGBM	Light Gradient Boosting Machines
LOO	Leave-One-Out
MILP	Mixed integer linear programming model-based hyperbox classifier
MDCFMN	Modified data-core-based fuzzy min–max neural network
MEFMNN	Modified-enhanced fuzzy min–max neural network
MFMC	Modified fuzzy min-max neural network for clustering
MFMNN	Modified fuzzy min-max neural network for two-stage pattern classification
EFMNNC	Enhanced fuzzy min-max neural network for clustering
MFMNN-GA	Modified fuzzy min-max neural network with genetic algorithms
MLF	Multi-level fuzzy min-max neural network
MMM-BL	Modified fuzzy min-max neural network with a new batch learning algorithm
MRHGRC	Multi-resolution hierarchical granular representations based classifier
Onln-GFMM	Original online learning algorithm for general fuzzy min-max neural network
PARC	Pruning Adaptive resolution classifier
RBF	Radial Basis function
ReFMN	Reflex Fuzzy Min Max Neural Network
ReFMN-FN	Reflex Fuzzy Min Max Neural Network with floating neurons
RFMNN	Refined Fuzzy Min-Max Neural Network
RH	Random Hyperboxes
RMILP	Refined mixed integer linear programming model-based hyperbox classifier
SAS	Smart adaptive systems
SFMN	Stochastic fuzzy min-max neural network
SS-FMM	Semi-supervised classification method based on fuzzy min–max neural network
SVM	Support Vector Machines
TDFMM	Top-down fuzzy min-max
TDFMMR	Top down fuzzy min-max regressor
TEH-GFMMNN	Tree ensemble hyperboxes via general fuzzy min-max neural network
XGBoost	Extreme Gradient Boosting
WFMM	Weighted fuzzy min-max neural network

List of Notations

If there is no specific definition in each section, the following are the default meanings of notations used in this thesis.

Symbol	Meaning
B_i	The *i-th* hyperbox in the list of hyperboxes
$X = [X^l, X^u]$	An input sample in the form of lower and upper bounds
$\mathbf{X}$	A sample space contains all input samples
$\mathcal{T}_N = \{(X_i, c_i)\}_{i=1}^N$	A training data set with N samples
c_i	Class label of sample X_i
c_X	Class label of sample X
$\mathcal{C}$	A set of categorical variables denoting classes to which the observations fall into
n	Number of features of input samples. If a input sample contains both continuous and categorical features, then n denotes the number of continuous features
r	Number of categorical features
$d(A, B)$	Euclidean distance between two vectors A and B
$X^l = (x_1^l, \ldots, x_n^l)$	Lower bound of an input sample in n-dimensional space
$X^u = (x_1^u, \ldots, x_n^u)$	Upper bound of an input sample in n-dimensional space
$X^d = (x_1^d, \ldots, x_r^d)$	A vector contains r categorical attributes for an input sample
$V_i = (v_{i1}, \ldots, v_{in})$	A vector represents the minimum point of the hyperbox B_i in n-dimensional space
$W_i = (w_{i1}, \ldots, w_{in})$	A vector represents the maximum point of the hyperbox B_i in n-dimensional space
G_i	Sample centroid of the hyperbox B_i
n_i	The number of current samples included in the hyperbox B_i
$b_i(X, V_i, W_i)$	Membership value between the input sample X and the hyperbox B_i
$\mathcal{V}$	A matrix contains all minimum points V_i of all hyperboxes B_i generated in the learning process
$\mathcal{W}$	A matrix contains all minimum points W_i of all hyperboxes B_i generated in the learning process
$\mathcal{L}$	A vector contains all class labels for all hyperboxes B_i generated in the learning process
θ	Maximum hyperbox size
Θ	A list of maximum hyperbox sizes
σ	Minimum similarity threshold so that two hyperboxes B_i and B_k can be aggregated
γ	Sensitivity parameter to control the decreasing speed of membership values
$H_j(B_i)$	The current entropy of hyperbox B_i for the *j-th* categorical feature
Ω_{ij}	A set of categorical values on the *j-th* categorical attribute of the hyperbox B_i
s_{ik}	Middle gap similarity measure between two hyperboxes B_i and B_k
$\widetilde{s}_{ik}$	shortest gap similarity measure between two hyperboxes B_i and B_k
$\widehat{s}_{ik}$	longest gap similarity measure between two hyperboxes B_i and B_k

Symbol	Meaning
Φ	A randomizing vector
Φ_i	An independent and identically distributed random vector
M	Number of base learners in an ensemble model
$\mathbb{1}(\cdot)$	Indicator function
$\mathbb{E}$	Expectation
r_s	Sampling rate to find the number of training samples for base learners in an ensemble model
m_f	The maximum number of used features for each base learner in an ensemble model
$\mathcal{E}^*$	Upper bound of the generalization error
$\overline{\rho}$	Average correlation between base learners in an ensemble model
$\overline{K}$	The average number of hyperbox candidates with the same class as the input sample
$\overline{R}$	The average number of hyperboxes representing classes different from the class of the input pattern in each iteration

Chapter 1

Introduction

We are living in the world with the rapid development of digital information, where the data volume generated by humans and machines is growing exponentially. In the era of big data, a question of how different data sources can be consumed and transformed into valuable, actionable knowledge has become critically important. Over the last few decades, many data mining methods have been studied and expanded aiming to invent an effective way for discovering meaningful knowledge and information from raw data. These techniques have contributed to mining diverse patterns hidden in data repositories (Zakaryazad and Duman 2016). By comprehending the knowledge underlying data, one can make important decisions more accurately in numerous fields ranging from business, finance, medicine to manufacturing sectors. There have been a large number of studies conducted on the subjects of data mining, data analytics, as well as predictive modeling over the last 50 years with remarkable enhancements of the computing equipment and the algorithms (Gabrys et al. 2005). Within all these studies, many machine learning algorithms have been developed with an emphasis on pattern clustering and classification.

Machine learning is a field of research concerned with the formulation and development of algorithms which provide the machine with the capability of learning and evolving their behaviors based on data coming from a variety of sources such as sensors or databases. Mitchell (Mitchell 1997) gave a formal definition of machine learning algorithms, which are software programs being able to do some tasks T by learning from experience E and their performance assessed by P. Therefore, when designing a new machine learning algorithm, one needs to think of what data to collect (E), what decisions the algorithm must generate (T), and what metrics are used to assess its performance (P).

Learning algorithms using hyperboxes as fundamental representational and building blocks are a branch of machine learning methods. These algorithms have enormous potential for high scalability and online adaptation of predictors built using hyperbox data representations to the dynamically changing environments. This thesis focuses on developing and extending the learning algorithms for a specific type of universal hyperbox-based classifiers, i.e., general fuzzy min-max neural network.

1.1 Background and Motivation

Most prevalent classes of machine learning techniques for pattern classification are various types of Artificial Neural Networks (ANNs) (Mukhopadhyay et al. 2002) because of their ability to learn from different types of input data, immunity to noise, generalization capability, and relatively high accuracy (Kim et al. 2011; Mohammed and Lim 2017b). It has also been observed that the traditional machine learning models are trained and their parameters are tuned on a given set of training data, and then the best model is deployed for dealing with specific problems without performing any updates afterward until the maintenance phase (Fontama et al. 2015). With the exponential growth of digital content and information leading to the rise in data volume, such conventional batch learning or offline learning techniques face many limitations because they adapt poorly to the rapid changes of data and suffer from costly re-training when adaptation is needed. It is desirable to develop robust and scalable machine learning algorithms with the aim of robust adaptation to evolving data and changing environments. Learning algorithms need to provide the models with the ability to capture the new features of data for decreasing the loss of predictive performance. One of the main issues with respect to the ANNs and conventional classifiers using both incremental and batch learning is catastrophic forgetting also known as stability-plasticity dilemma (McCloskey and Cohen 1989), which relates to the inability of the classifier to retain previously learnt patterns when new patterns are absorbed by that classifier (Polikar et al. 2001). Hence, classifiers frequently forget learned information while learning new information (Grossberg 2013). To tackle this issue, a classifier has to be able to remain plastic enough to absorb new information and simultaneously be stable enough

to maintain previously acquired knowledge while learning new information (Grossberg 2013). Resolving the stability-plasticity dilemma problem is especially essential when using online learning for classifiers (McCloskey and Cohen 1989; Yang et al. 2004).

In addition to online learning, the effective machine learning model should possess several properties as follows (Simpson 1992):

- Non-linear separability: This is ability to construct non-linear class boundaries (Sonule and Shetty 2017).
- Overlapping region: The model is capable of formulating non-linear decision boundary to minimize the misclassification error for all overlapping classes.
- Soft and hard decisions: The algorithm should offer both soft and hard decisions. The hard decision allocates a sample to a single class, while the soft decision outputs the degree-of-fit of that pattern to a given class.
- Training time: The learning algorithm to train the model should be fast and have the ability to learn arbitrarily complex class decision boundaries.
- Verification and validation: This property imposes that each machine learning model should have the mechanisms to verify and validate its performance.
- Adjusting parameters: The model should have as few parameters that need to be adjusted during the training process as possible.

The online learning ability of the effective learning systems with above properties is highly expected to deal with the huge amount of data. With an increasing in computing power and the availability of the huge amount of data, it is now feasible and even expected to build robust algorithms and systems leveraging all available resources and performing intelligent tasks, such as language understanding, adaptive pattern recognition, reasoning with uncertainty, and many others (Gabrys et al. 2005). Such a system can develop and modify its functionality and architecture in a continuous, self-organized, adaptive, and interactive manner (Kasabov 2019;

Gabrys et al. 2005). Seven major characteristics of smart adaptive systems (SAS) were proposed in (Kasabov 2019):

1. Capability of fast learning from a vast amount of data with one-pass learning mode or beginning from little or no prior knowledge.

2. Having an evolving, open, and adjustable architecture. Particularly, the system can add new input and output variables as well as evolving or changing connections among components and modules during the operation in an incremental way.

3. The system should adapt and accommodate the new data in a real-time, online, and incremental approach.

4. Active interaction with other learning systems and the working environment in the multi-modular or hierarchical ways is employed to enhance the performance of the system

5. Representing both time and space in their different scales, long and short-term memory, forgetting, age of data, etc.

6. Having the ability of self-improvement, self-assessment in terms of behaviors, analyzing its own performance, and explaining learned knowledge.

7. Data learning and knowledge representations are performed by comprehensive and flexible mechanisms ranging from supervised and unsupervised learning, evolving clustering, forgetting/pruning to fuzzy rule insertion and extraction. The system should deploy a memory-based method to facilitate adding, retrieving, and eliminating individual pieces of data and information.

Such a system can be built based on multi-dimensional hyperbox fuzzy sets. Hyperboxes can be used to deal with the pattern classification and clustering problems effectively by partitioning the pattern space and assigning a class label or cluster associated with a degree of certainty for each region. Each fuzzy min-max hyperbox is represented by minimum and maximum points together with a fuzzy membership

function. The membership function is employed to compute the degree-of-fit of each input sample to a given hyperbox. Meanwhile, the hyperboxes are continuously adjusted during the training process to cover the input patterns. The idea of using hyperbox fuzzy sets for building machine learning models was proposed in the fuzzy adaptive resonance theory neural networks (Carpenter et al. 1991), and then it was enhanced to formulate the fuzzy min-max neural network (FMNN) in Simpson's work for classification (Simpson 1992) and clustering (Simpson 1993). Since then, many studies have focused on enhancing this type of neural network, in which a general fuzzy min-max neural network (GFMMNN) (Gabrys and Bargiela 2000) was a big improvement and generalization of the FMNN combining both classification and clustering in a single framework. This study aims to develop and extend learning algorithms using hyperbox fuzzy sets as basic representational units inspired by the GFMMNN. Therefore, the main algorithms using hyperbox representations will be reviewed in this study to identify the open problems in this field of research.

Existing hyperbox-based machine learning algorithms have addressed some aspects of smart adaptive system requirements mentioned above. However, there are still a plethora of issues needing to be tackled to formulate a real smart adaptive system. This study is built on the general fuzzy min-max neural network and principles of developing classifiers with good generalization performance discussed in Gabrys (2004). The main purpose of this study is to formulate robust and scalable learning algorithms, which form a solid base to be able to construct a real smart complex adaptive system in the near future.

Robustness. Jen (2003) defined that robustness refers to the feature persistence of systems against perturbations. A robust predictive machine learning algorithm can result in the same performance even when a small perturbation occurs in the data or internal parameters. Robustness provides a general framework to study the coupling between the behavior and organizational architecture of the system and dynamically changing environment. Robustness prevents systems from deteriorating too much in their performance when there are some changes in data by some factors such as noise. As a result, the learning algorithms using hyperbox repre-

sentations proposed in this study will be less affected by noise and disturbance of data. The definition of robustness can also be expanded in the direction that learning algorithms need to maintain a good performance for input patterns with both continuous and categorical features and data with high dimensionality.

In terms of learning from data with mixed-attribute, current learning algorithms of the GFMMNN only work well on data with continuous features. However, the data generated by many real-world applications are usually in the form of mixed-type features. For example, the mixed-attribute data are more and more popular in a wide range of applications from the credit approval data to medical diagnostic data (Huang et al. 2019). Hence, to apply the GFMMNN to such problems, it is necessary to extend its current learning algorithms so that they can deal effectively with mixed-attribute data. This is one of the motivations to make the current learning algorithms of the GFMMNN become more robust and flexible in this study.

It is difficult to build a good classifier in high dimensional spaces because of the decrease in the generality and representativeness of training samples when the number of dimensions increases (Domingos 2012). Certainly, the hyperbox-based classifiers also face such curse of dimensionality problem. One of the solutions to build a high performing model in the high dimensional space is the use of ensemble learning, in which base learners are trained on a subset of feature spaces. Ensemble models help to reduce variance of individual learners while only slightly increasing bias of the final model (Domingos 2012). Therefore, ensemble models are usually more reliable than individual models. This is another motivation to construct an ensemble model from many hyperbox-based classifiers.

Scalability. Nowadays, the rapid increase in both amount and dimensionality of data imposes new challenges on data-driven machine learning models. Learning algorithms should have the capability of dealing effectively with large-sized and/or very high dimensional datasets. Therefore, scalability is a crucial issue affecting the applicability and usage of every learning model to real-world applications. As a result, the parallelism abilities and processing in many cores/GPUs or clusters are considered when designing learning algorithms. The computational performance of

the algorithms can be significantly improved by adding more resources to handle the problems in parallel. The different techniques to accelerate learning algorithms for the GFMMNN are also sources of motivations for this study.

Interpretability and Transparency. In addition to robustness and stability, transparency is one of the crucial factors leading to the applicability of machine learning algorithms in the practical applications (Rudin 2019). Transparency of a model means that both experts and non-experts may easily comprehend its parameter values, representations, assumptions, workflow, and predictive outcomes (Sampson et al. 2019). Nauck (2005) claimed that the transparency characteristic is essential for a predictive model operating in the following situations: supporting human in making decisions in critical fields with a need for explaining and taking responsibility for those decisions such as health-care or criminal justice; verifying the changes in knowledge used as a base for prior analytics; interpreting the predictive results for non-experts. In a recent study, Rudin (2019) has highlighted that there is a high demand for interpretable models to substitute black-box models in assisting decision-makers in areas with the requirement of high safety and trust. Most of the current high performing models use ANNs and deep learning mechanisms for pattern recognition problems. However, the main disadvantage of these ANNs is that they do not have the capability of giving explanations of their predictive results to humans explicitly. This drawback restricts the widespread use of the ANNs for critical domains such as health-care and criminal justice. In contrast, one of the advantages of using hyperbox representations for classifiers is that it can lead to interpretable models. However, model accuracy is usually reduced when increasing model transparency (Nauck 2003). As a result, this thesis proposes a method of maintaining accuracy in an acceptable range while reducing the complexity of the predictive model. This method is suitable for applications with a high need for transparency and accuracy.

In summary, this thesis researches methods of constructing classifiers based on hyperbox representations, in which the characteristics of scalability, robustness, and transparency are considered.

1.2 Research Aims and Objectives

The overall aim of this study is to build robust and scalable classifiers towards forming smart adaptive systems based on and using hyperboxes as fundamental building blocks and representations. To achieve this overall aim, this thesis focuses mainly on addressing the particular objectives as follows:

1. **Conducting a comprehensive survey and empirical assessment of existing hyperbox-based learning algorithms**. Recognizing and evaluating the existing methods is a very important step to develop novel solutions. Hence, this objective is to perform an assessment of existing learning algorithms and categorize them in a meaningful manner. Completing this objective will create a solid foundation to develop new learning algorithms in further studies.

2. **Development of new online learning algorithms for single GFMM models**. This objective is to build new robust learning algorithms for the GFMMNN aiming to address the weak points identified in the literature and empirical analyses. The results of this objective contribute to formulating learning algorithms more robust for the GFMMNN on datasets with different types of features, e.g., mixed attributes, as well as noisy data. The robustness of the algorithms will be analyzed in detail via many datasets.

3. **Investigation of solutions to accelerate the current learning algorithms of the GFMMNN**. This objective is to address the long training time of learning algorithms for the GFMMNN. Different techniques related to the changes in the learning steps and parallel implementation mechanisms need to be developed and assessed on variety of large-sized datasets.

4. **Development of methods to increase the transparency of hyperbox-based classifiers but still maintaining an acceptable accuracy**. This objective is motivated by the practical demands for simplification of data towards being consistent with human abstract thinking and problem solving as

well as tolerance of uncertainty. This objective is to propose a method of constructing more interpretable learning models using hyperbox fuzzy sets. The effectiveness and robustness of the method will be analyzed across experiments on synthetic and real-world datasets.

5. **Development of robust ensemble models using hyperbox representations**. As mentioned in the background section, redundant or irrelevant features usually make a negative impact on the model performance. Additionally, the ensemble model is usually more robust and high-performing than single models. Therefore, this objective is to construct robust ensemble models from many single hyperbox-based models.

1.3 Original Contributions

This thesis contains a number of key contributions as highlighted below and expanded upon in the following chapters:

- **A comprehensive survey on hyperbox-based machine learning algorithms**. This contribution focuses on reviewing and categorizing existing algorithms in terms of theory and application. This thesis also analyzes in depth the advantages and disadvantages of each algorithm. Through analyses and evaluations, the potential research directions in this field of research is identified. (*Chapter 2*). This is linked to Objective 1.

- **Conducting a comparative study of performance influencing factors, advantages, and drawbacks of the GFMMNN on pattern classification problems**. This study is built on the learning principle of the general fuzzy min-max neural network. Therefore, it is necessary to assess the impact of user-defined hyper-parameters on classification performance of the GFMMNN. This contribution also compares the performance of GFMMNN to other FMNNs and prevalent machine learning models. Empirical outcomes have informed potential research directions of this class of machine learning algorithms in the future. (*Chapter 3*). This is linked to Objective 1.

- **Proposing a new online learning algorithm for the GFMMNN working properly on continuous features**. This contribution proposes an improved version of the current online learning algorithm for the GFMMNN to tackle existing issues concerning expansion and contraction steps as well as the way of dealing with unseen data located on decision boundaries. The proposed approach eliminates the contraction process in the original online learning algorithm, which is more likely to increase the error rate as shown in the literature when using high values of hyperbox sizes. This contribution enables to build the learning models robust to the disturbance of internal parameters. (*Chapter 4*). This is linked to Objective 2.

- **Proposing a new online learning algorithm for the GFMMNN able to work on mixed-attribute data**. One of the downsides of the original learning algorithms for the GFMMNN is the inability to handle and learn from the mixed-attribute data. The existing approaches in the literature for handling mixed-attribute data are not suitable for online learning algorithms working in the dynamically changing environments without ability to retrain or access full historical data, which are usually required for many real world applications. Hence, this contribution builds an extended version of the previously proposed online learning algorithm. The proposed method can handle the datasets with both continuous and categorical features. It uses the change in the entropy values of categorical features of the samples contained in a hyperbox to determine if the current hyperbox can be expanded to include the categorical values of a new training instance. An extended architecture of the original GFMMNN and its new membership function are also introduced for mixed-attribute data. Important mathematical properties of the proposed learning algorithms are also presented and proved. (*Chapter 5*). This is linked to Objective 2.

- **Proposing the approaches to accelerating training algorithms of GFMMNN**. One of the problems of learning algorithms for the GFMMNN is a long training time even when the number of samples is relatively low because

the time complexity of the learning algorithms is proportional to the number of dimensions of samples. This is a quite common problem shared by many prototype-based methods requiring frequently repeated distance or similarity calculations. Therefore, the first solution is the use of matrix operations and GPUs to accelerate learning algorithms for very high dimensional datasets. It redefines the learning algorithms to facilitate the parallel execution on the GPUs by taking advantage of intrinsic parallelization characteristics of the GFMMNN. Another problem related to the current learning algorithms is the redundancy of the selected hyperbox candidates in the learning steps, which also lead to the increase in training time. Hence, the second solution is to propose and prove mathematical lemmas to reduce significantly the considered hyperboxes during the learning algorithms of the GFMMNN. (*Chapter 6*). This is linked to Objective 3.

- **Building a new data classification model based on the multi-resolution of granular data representations in combination with the online learning ability of the GFMMNN**. This contribution introduces a method of constructing classifiers from multi-resolution hierarchical granular representations using hyperbox fuzzy sets. The proposed approach forms a series of granular inferences hierarchically through many levels of abstraction. Therefore, it can reuse the learned knowledge from the lowest abstraction level to construct new classifiers at higher abstraction levels with the low trade-off between the simplification and accuracy. In addition, the proposed approach can reduce the data size significantly as well as handle the uncertainty and incompleteness associated with data in real-world applications. (*Chapter 7*). This is linked to Objective 4.

- **Developing a simple yet powerful ensemble classifier, called Random Hyperboxes, constructed from individual hyperbox-based classifiers trained on the random subsets of sample and feature spaces of the training set**. A generalization error bound of the proposed classifier based on the strength of the individual hyperbox-based classifiers and the correlation

among them is also proved. The effectiveness of the proposed classifier is analyzed using a carefully selected illustrative example and compared empirically with other popular single and ensemble classifiers. (*Chapter 8*). This is linked to Objective 5.

A summary of research motivation, research objectives, and contributions is presented in Figure 1.1.

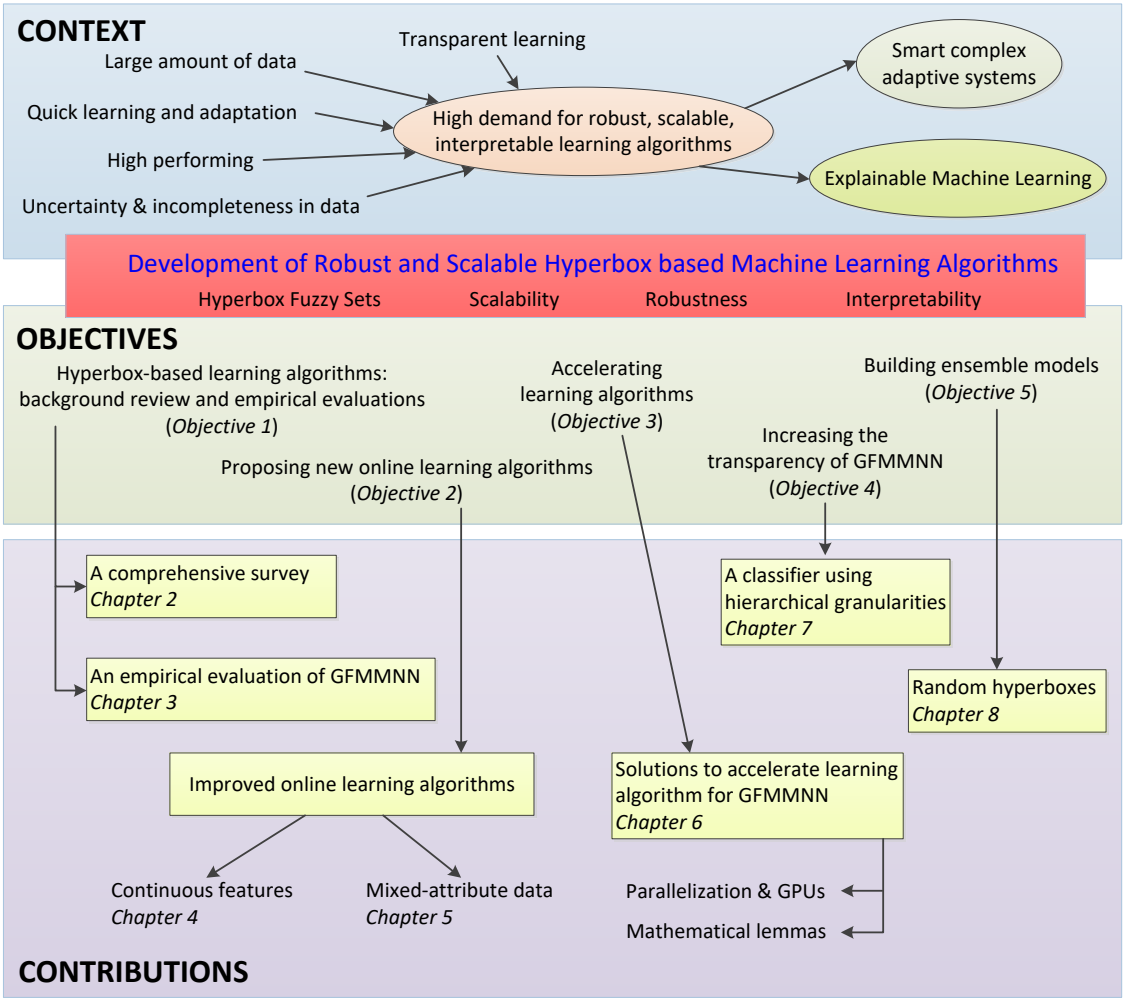


Figure 1.1 : A summary of motivation, objectives, and contributions in this thesis.

1.4 Thesis Organization

The organization of this thesis is shown in Figure 1.2. This introduction chapter has described motivation, aims, objectives, and contributions of my Ph.D. project.

The main content of next chapters is organized as follows.

Chapter 2: This chapter reviews the progression of hyperbox-based machine learning algorithms and their applications to real-world problems. A taxonomy of existing learning algorithms using hyperbox representations as basic building blocks has been conducted. For each group, main characteristics of algorithms are summarized. Main drawbacks of current learning algorithms and open research directions are also identified in this chapter.

Chapter 3: This chapter presents a comprehensive empirical study of performance influencing factors, advantages, and shortcomings of the GFMMNN on pattern classification problems. The subjects of this chapter include (1) the impact of maximum hyperbox size, (2) the influence of the similarity threshold and measures on the agglomerative learning algorithm, (3) the effect of data presentation order, (4) comparative performance evaluation of the GFMMNN with other types of fuzzy min-max neural networks and prevalent machine learning algorithms. This chapter also indicates potential research directions for this class of machine learning algorithms in the future.

Chapter 4: This chapter proposes an improved version of the current online learning algorithm for the GFMMNN to overcome existing issues regarding expansion and contraction steps, as well as the problem of unseen data located on decision boundaries. These shortcomings decline the classification performance of the current online learning algorithm, thus its improved version is proposed in this chapter to address the above limitations. In order to reduce the sensitivity to the training samples presentation order of the new online learning algorithm, an simple ensemble method is also proposed in this chapter.

Chapter 5: This chapter reviews different methods of handling mixed-attribute data for classification problems using the GFMMNN. A disadvantage of most of the current learning algorithms for the GFMMNN is that they can handle effectively numerical valued features only. Therefore, this chapter provides some potential approaches to adapting GFMM learning algorithms for classification problems with mixed-type or only categorical features as they are very common in practical ap-

plications and often carry very useful information. From the obtained experimental outcomes, this chapter proposes an extended online learning algorithm for the GFMMNN. The proposed method can handle the datasets with both continuous and categorical features.

Chapter 6: This chapter shows the solutions to accelerate the learning algorithms for the GFMMNN. The first solution aims to reformulate and represent the learning algorithms in a format allowing for their parallel execution and subsequently leveraging the computational power of the GPUs. Therefore, the original implementation of the GFMMNN is modified by matrix computations to be executed on the GPUs with respect to the very high-dimensional datasets. The second solution reduces the unsuitable hyperboxes selected as the potential candidates of the expansion step to cover a new input pattern in the online learning algorithms or candidates of the hyperbox aggregation process in the batch learning algorithms. This method is based on the mathematical formulas to form a new solution aiming to eliminate the hyperboxes which are certain not to satisfy expansion or aggregation conditions, and in turn decreasing the training time of learning algorithms.

Chapter 7: This chapter presents a method to build simple hyperbox-based classifiers while still maintaining a good classification performance using learning from different granularity levels. The proposed approach forms a classifier from a series of granular inferences hierarchically through many levels of abstraction. Therefore, it can achieve a high accuracy at a high degree of abstraction thank to reusing the knowledge learned from lower levels of abstraction. This chapter also analyzes and assesses the effectiveness and efficiency of the proposed method on variety of synthetic and real-world datasets.

Chapter 8: This chapter presents a method to build a simple but effective ensemble model, called Random Hyperboxes, in which base learners are hyperbox-based classifiers trained on random subsets of samples and features. This chapter also describes a generalization error bound of the proposed classifier based on the strength of the individual hyperbox-based classifiers as well as the correlation among them. The effectiveness of the proposed classifier is analyzed and compared empirically

with other popular single and ensemble classifiers using a variety of datasets.

Chapter 9: This chapter concludes the thesis by discussing the key findings and potential research directions for further studies.

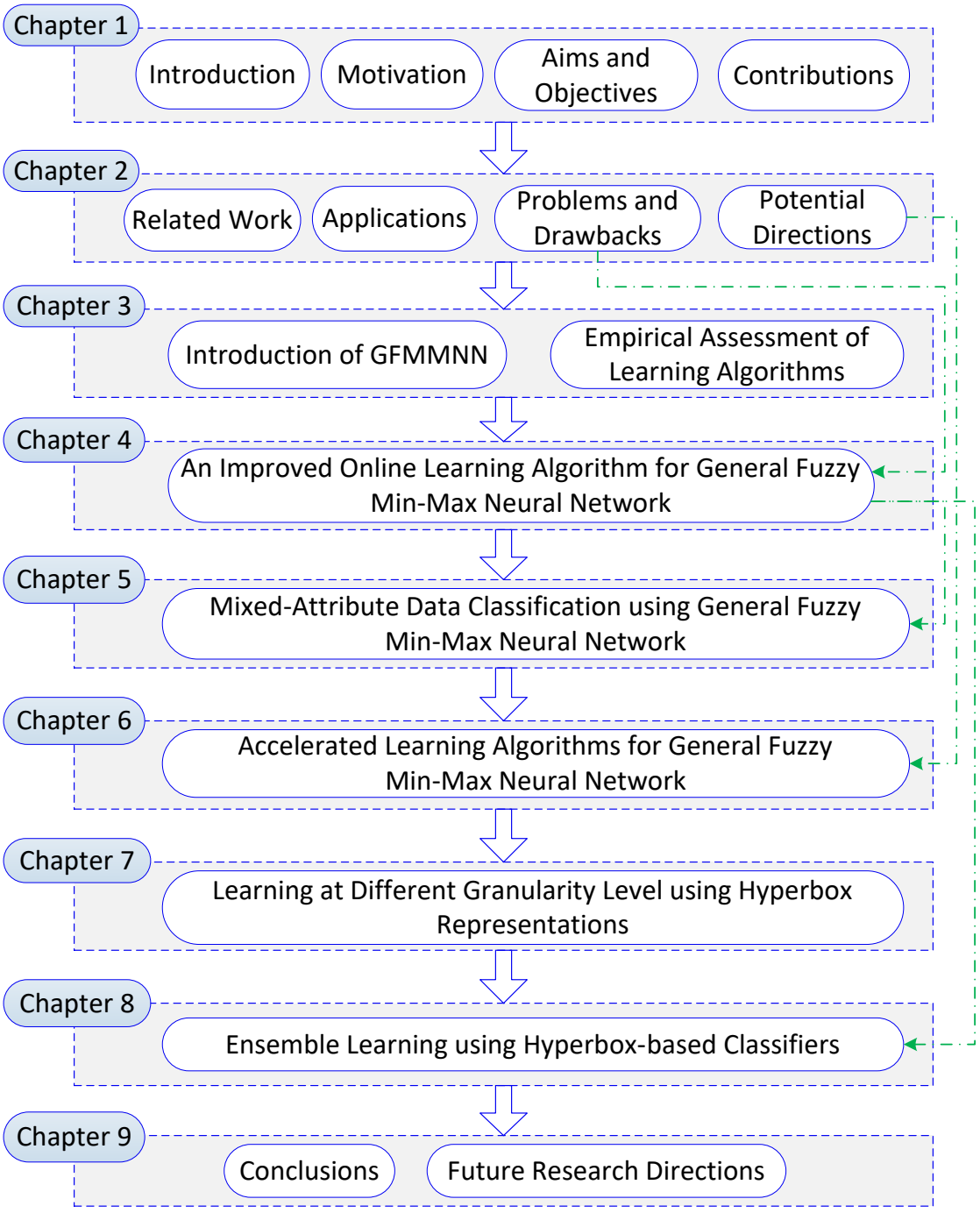


Figure 1.2 : Thesis structure

Chapter 2

Literature Review

This chapter shows a summary of the taxonomy and description of main hyperbox-based machine learning algorithms. In particular, it describes the background knowledge of machine learning models using the hyperbox representations. The overview of the architecture and main content of types of fuzzy min-max neural network and its improved versions are also detailed. The descriptions of hybrid models based on hyperboxes and other hyperbox-based machine learning techniques are presented in this chapter as well. This chapter concludes with a discussion on the main characteristics of hyperbox-based machine learning algorithms and potential research directions. The content of this chapter is taken from the paper (Khuat et al. 2021b):

- **Thanh Tung Khuat**, Dymitr Ruta, and Bogdan Gabrys, "Hyperbox based machine learning algorithms: A comprehensive survey," *Soft Computing*, vol. 25, pp. 1325–1363, 2021.

2.1 Introduction

Motivated by all of the issues regarding the construction of robust learning models overcoming the drawbacks of the traditional ANNs as mentioned in Section 1.1 in Chapter 1, Simpson (1992) suggested deploying hyperbox fuzzy sets to generate and store information as hidden units in the form of a neural network architecture. He introduced two kinds of hyperbox-based fuzzy min-max neural networks, i.e., one supervised learning technique for sample classification (Simpson 1992) and one model for data clustering (Simpson 1993). Due to the benefits of the fuzzy min-max neural network (FMNN), a great deal of its improved variants have been proposed such as general fuzzy min-max neural network (GFMMNN) (Gabrys and Bargiela 2000), weighted fuzzy min-max neural network (Kim et al. 2004), an adaptive res-

olution fuzzy min-max neural network (Rizzi et al. 2002), an inclusion/exclusion fuzzy hyperbox classification network (Bargiela et al. 2004), a fuzzy min-max neural network classifier with compensatory neurons (Nandedkar and Biswas 2004, 2007a), a data-core-based fuzzy min-max neural network (Zhang et al. 2011), and a multi-level fuzzy min-max neural network (Davtalab et al. 2014). Fuzzy min-max neural networks include many hyperboxes, each one covers an area determined by its minimum and maximum coordinates in the n-dimensional sample space. Each hyperbox is associated with a fuzzy membership function calculating the goodness-of-fit of an input sample to a certain class. From the original version proposed by Simpson (1992), learning algorithms of the fuzzy min-max neural networks have been significantly enhanced with the emergence of algorithms combining supervised and unsupervised learning such as the GFMMNN (Gabrys and Bargiela 2000) and the general reflex fuzzy min-max neural network (GRFMN) (Nandedkar and Biswas 2009). Other improvements of the FMNN have been related to the construction of algorithms dealing with missing data and operating on observable subspaces without missing values imputation (Gabrys 2002c; Castillo and Cardenosa 2012) or combination of multiple hyperbox classifiers at a model level taking advantage of ensemble performance while reducing impact of user-defined parameters (Gabrys 2002b).

The traditional neural networks are considered as black boxes due to the fact that they are not able to explain their predicted results. When it comes to data analysis, one of the salient properties is the ability to extract explanatory rules for inference from data samples (Cheng and Miao 2011). Therefore, machine learning models should offer a useful explanatory mechanism of their outcomes to the user. One of such models is the decision tree (Seera et al. 2015). Fuzzy min-max neural networks can generate explanation based on the rules deduced from the hyperbox min–max values, but it cannot form a compact rule set interpretable for end-users because the number of hyperboxes can be large. Therefore, instead of extracting rules directly from the individual hyperbox level, decision trees have been adopted to obtain rules at the global level. As a result, many researchers have introduced hybrid models in combination of hyperbox-based machine learning algorithms with

decision trees or other rule extractors to increase the ability to explain the results for single models such as an enhanced FMNN with an ant colony optimization based rule extractor (Sonule and Shetty 2017), a hybrid model of FMNN and the classification and regression tree (Seera et al. 2012; Seera and Lim 2014), a fuzzy min-max based clustering tree (Seera et al. 2016), a fuzzy min-max decision tree (Mirzamomen and Kangavari 2016), and combining multiple decision trees using the GFMMNNs (Eastwood and Gabrys 2011).

Apart from combination of the fuzzy min-max neural networks and other classification techniques, several researchers have introduced other methods to construct base hyperboxes and evolve them using optimization algorithms such as a differential evolution (Reyes-Galaviz and Pedrycz 2015) and an ant colony optimization (Ramos et al. 2009).

It is noted that in the literature, there have been other min-max machine learning methods which build and evolve the architectures of learning models as training samples. These methods have constructed using fundamental building blocks other than hyperboxes. For example, hypersphere methods, where minimum and maximum radii are defined or evolved in an online mode, were used for implementation of evolving connectionist systems such as dynamic evolving neural-fuzzy inference system (Kasabov and Song 2002) and evolving fuzzy neural networks (Kasabov 2001). However, these classes of learning algorithms are beyond the scope of this study. Several hypersphere-based machine learning methods can be found in Kasabov (2007). The main focus of this thesis is the machine learning algorithms using hyperboxes as fundamental representation blocks. This study seeks to classify and clarify the properties of machine learning models based on the hyperbox representations, learning algorithms, as well as their enhancements. In other words, this chapter aims to provide a comprehensive survey of literature on hyperbox-based machine learning algorithms. The core ideas and key descriptions of typical algorithms, their expansions, as well as their real-world applications are presented in detail. It is expected to clarify issues as follows:

1. How to group the machine learning models using hyperbox representations.

2. What was the development trajectory of the original fuzzy min-max neural networks in the last two decades.

3. What methods have been deployed to generate hybrid models between different types of FMNN and other classification or clustering techniques.

4. Apart from network structures, what other methods have been used to design and evolve hyperbox-based models.

5. Identifying research gaps in the current hyperbox-based machine learning algorithms and propose new future research directions.

Regarding the fuzzy min-max neural networks, which is part of this study, there have been several previously published surveys. Jambhulkar (2014) compared the multi-level fuzzy min-max neural networks with the original FMNN and its four different variants. However, that survey mentioned only six types of fuzzy min-max neural networks, and it did not yet analyze the limitations of existing types of networks. To overcome these drawbacks, Jain and Kolhe (2015) analyzed more details from some additional variants of the original FMNN, and they concluded that multi-level fuzzy min-max neural networks are the best one among the discussed methods. However, the survey was only limited to seven types of fuzzy min-max networks, and the authors used the classification accuracy of training samples as a comparison criterion for fuzzy min-max classifiers. The high accuracy on the training sample does not guarantee the good performance of the constructed classifiers because it may reflect overfitting and the loss of generality. In another study, Kulkarni and Honwadkar (2016) reviewed different types of fuzzy neural networks for classification and clustering. They categorized the networks to three groups consisting the ones for classification, clustering, and hybrid models for both classification and clustering. However, they not only focused on fuzzy min-max networks but also other types of fuzzy neural networks such as hypersphere and hyperline ones. These three surveys have not yet clarified the still existing limitations and applications of the fuzzy min-max neural networks to real-world problems. Recently, there has been a survey on the fuzzy min-max neural networks for pattern classification until 2017

introduced by Sayaydeh et al. (2019). Authors classified the types of fuzzy min-max neural networks into two groups, i.e., ones with and without contraction process. They summarized the use of different types of fuzzy min-max neural networks in tackling real-world applications. Nonetheless, the paper did not present in depth the reasons for the proposals of variants of the original fuzzy min-max classifier and their improvements compared to previously proposed versions. The research direction part only mentioned a small aspect regarding the potential of the family of fuzzy min-max neural networks. This study is not restricted to the fuzzy min-max neural networks but expands to general hyperbox-based machine learning algorithms, the combination of hyperbox fuzzy sets and tree-based algorithms, ensembles of multiple hyperbox-based models, the use of hyperbox fuzzy sets to deal with missing data, and learning algorithms based on hyperboxes without forming the neural network structure.

2.2 Overview and Taxonomy of Hyperbox-based Learning Algorithms

Hyperbox-based machine learning models are constituted by hyperboxes associated with their membership functions which are utilized to generate fuzzy subsets of an n-dimensional sample space (Simpson 1992). Each hyperbox occupies a region in the feature space and is defined by pairs of minimum and maximum points. Figure 2.1 represents an example of a three-dimensional hyperbox together with its minimum point and maximum point. Based on new incoming data samples, the learning model generates a number of hyperboxes incrementally to establish new classes/clusters or tune the existing hyperboxes to cover new samples. It is possible to produce hyperboxes covering an arbitrary value range in each dimension, but the range from 0 to 1 is widely used for each dimension to make the computations simpler. Therefore, each hyperbox is usually determined by a set of minimum and maximum vertices in the n-dimensional unit cube (I^n).

Formally, each hyperbox fuzzy set is given by an ordered set

$$B_i = \{X, V_i, W_i, b_i(X, V_i, W_i)\} \tag{2.1}$$

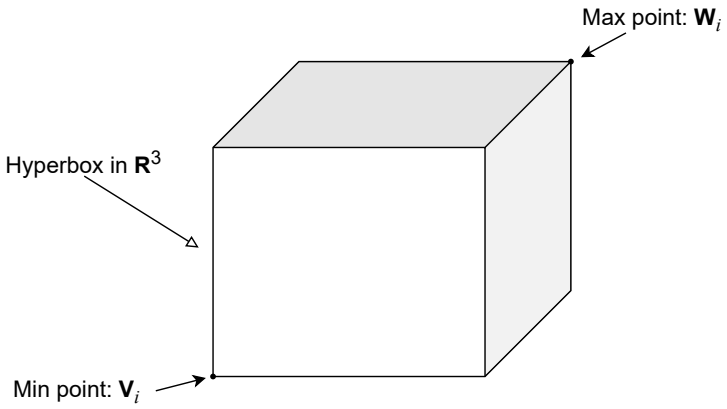


Figure 2.1 : 3-D Hyperbox

where B_i is the i^{th} hyperbox fuzzy set, $X = (x_1, x_2, \ldots, x_n) \in I^n$ is the input sample, $V_i = (v_{i1}, v_{i2}, \ldots, v_{in})$ and $W_i = (w_{i1}, w_{i2}, \ldots, w_{in})$ are minimum and maximum vertices of the hyperbox B_i respectively, and the membership function of B_i is represented by $0 \leq b_i(X, V_i, W_i) \leq 1$.

The membership function is a crucial component in the fuzzy min-max classification and clustering techniques. It is utilized to measure the degree to which the input pattern X belongs to the hyperbox B_i defined by the minimum point V_i and the maximum point W_i. When the sample is completely contained within the hyperbox, the degree of membership of X is one, and it decreases when X moves away from the hyperbox B_i.

For a trained hyperbox-based learning model, to classify or cluster an input sample, it is required to compute the membership value for each class/cluster c_k by accumulating the membership functions of all the hyperboxes representing this class/cluster as follows:

$$c_k = \bigcup_{i \in K} b_i \tag{2.2}$$

where K is the set of hyperboxes associated with class/cluster c_k. It is noted that fuzzy union operator in this equation is usually the maximum of its membership values. Then, the input pattern is assigned to a corresponding class/cluster with the highest membership value.

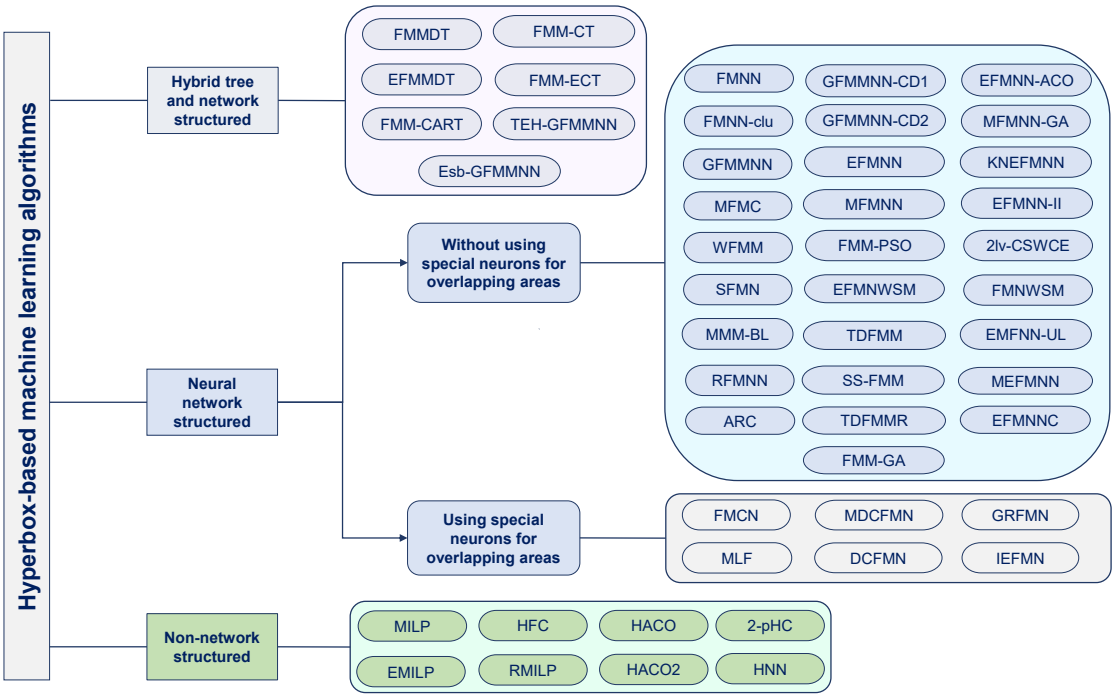


Figure 2.2 : A taxonomy of hyperbox-based learning algorithms

This study classifies the hyperbox-based machine learning algorithms into three groups. The first group comprises studies that construct a neural network architecture from hyperbox fuzzy sets. Learning algorithms are designed to adjust the placement of hyperboxes to cover the training samples in the input space. In the process of expanding hyperboxes to include new training patterns, hyperboxes belonging to different classes are likely to overlap with each other. There are two methods to deal with this overlapping problems. While several studies have introduced specialized hyperboxes to handle overlapping areas, other researchers have used the contraction procedure to avoid overlapping regions or not allow to overlapping areas appearing when building and expanding existing hyperboxes. Therefore, fuzzy min-max neural networks can be divided into two sub-groups depending on the employed mechanism for handling overlapping hyperboxes. The second group of hyperbox-based machine learning models consists of ones that integrate the strong points of various fuzzy min-max neural networks and tree-based classification techniques or the combination of base models to build the ensembles. The last group includes the models without

being connected by network structures. Hyperboxes in these machine learning models are constructed and evolved using different approaches such as mathematical

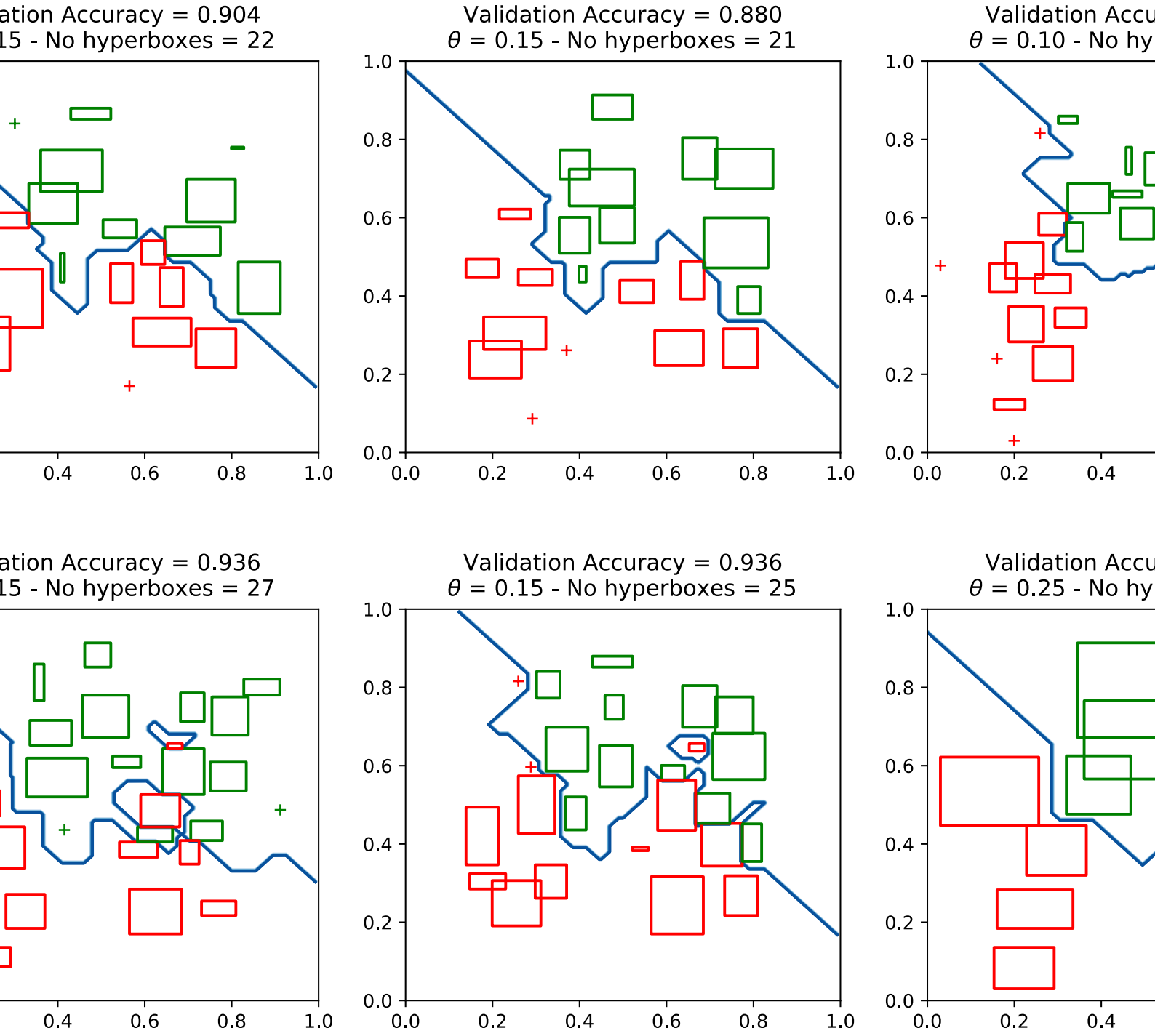


Figure 2.3 : A demonstration of a hyperbox-based classification model.

Table 2.1 : Summary of the main hyperboxed-based machine learning algorithms

ID	Model	Year	Type
2-pHC	Two-phase hyperbox classifier (Bortolan and Pedrycz 2007)	2007	classification
2lv-CSWCE	Two-level classification system with testing in dynamically changing environment (Gabrys and Bargiela 1999)	1999	classification
ARC	Adaptive resolution min-max neural network classifier (Rizzi et al. 1998)	1998	classification
DCFMN	Data-core-based fuzzy min-max neural network (Zhang et al. 2011)	2011	classification
EFMMDT	An evolving fuzzy min–max decision tree (Mirzamomen and Kangavari 2017)	2017	classification
EFMNN	Enhanced fuzzy min-max neural network (Mohammed and Lim 2015)	2015	classification
EFMNN-ACO	Enhanced fuzzy min–max neural network with ant colony optimization (Sonule and Shetty 2017)	2017	classification
EFMNNC	Enhanced fuzzy min-max neural network for clustering (Seera et al. 2015)	2015	clustering
EFMNN-II	Enhanced fuzzy min-max neural network with K-nearest hyperbox expansion rule and pruning (Mohammed and Lim 2017b)	2017	classification
EFMNN-UL	Evolved fuzzy min-max neural network for unknown labeled data (Ma et al. 2021)	2021	classification
EFMNWSM	Extended Fuzzy min-max neural network with symmetric margin (Forghani and Yazdi 2015)	2015	classification
EMILP	A enhanced version of mixed integer linear programming model-based hyperbox classifier (Maskooki 2013)	2013	classification
Esb-GFMMNN	Ensemble of neuro-fuzzy classifiers (Gabrys 2002b)	2002	classification; clustering
FMCN	Fuzzy min-max neural network with compensatory neuron (Nandedkar and Biswas 2004, 2007a)	2004; 2007	classification
FMNN	Fuzzy min-max classification neural network (Simpson 1992)	1992	classification
FMNN-clu	Fuzzy min-max clustering neural network (Simpson 1993)	1993	clustering
FMM-CART	Offline and online fuzzy min–max neural network and classification and regression trees (Seera et al. 2012; Seera and Lim 2014)	2012; 2014	classification; regression
FMM-CT	Fuzzy min-max clustering neural network with the clustering tree (Seera et al. 2016)	2016	clustering
FMM-GA	Fuzzy min-max neural network with genetic algorithms (Azad and Jha 2016)	2016	classification
FMMDT	Fuzzy min–max neural network based decision tree (Mirzamomen and Kangavari 2016)	2016	classification
FMM-ECT	Fuzzy Min–Max neural network with an ensemble of clustering trees (Seera et al. 2018)	2018	clustering
FMM-PSO	Fuzzy min-max neural network with the particle swarm optimization (Azad and Jha 2017)	2017	classification
FMNWSM	Fuzzy min-max neural network with symmetric margin (Forghani and Yazdi 2015)	2015	classification
GFMMNN	General fuzzy min-max neural network (Gabrys and Bargiela 2000; Gabrys 2002a)	2000; 2002	classification and clustering

Table 2.1 : Summary of the main hyperboxed-based machine learning algorithms

GFMMNN-CD1	General fuzzy min–max neural networks for categorical data (Castillo and Cardenosa 2012)	2012	classification
GFMMNN-CD2	Enhanced general fuzzy min–max neural networks for categorical data (Shinde and Kulkarni 2016)	2016	classification
GRFMN	General reflex fuzzy min-max neural network (Nandedkar and Biswas 2009)	2009	classification; clustering
HACO	Hyperbox based clustering with ant colony optimization (Ramos et al. 2009)	2009	clustering
HACO2	Hyperbox classifier with ant colony optimization (Ramos et al. 2008)	2008	classification
HFC	Hyperbox fuzzy classifier (Abe and Ming-Shong 1995)	1995	classification
HNN	Hyperbox neural network algorithm (Palmer-Brown and Jayne 2011)	2011	classification
IEFMN	Inclusion/Exclusion fuzzy min-max neural network (Bargiela et al. 2004)	2004	classification
KNEFMNN	Enhanced fuzzy min-max neural network with K-nearest hyperbox expansion rule (Mohammed and Lim 2017a)	2017	classification
MDCFMN	Modified data-core-based fuzzy min–max neural network with new learning mechanism (Ma et al. 2012)	2012	classification
MEFMNN	Modified-enhanced fuzzy min–max neural network (Upasani and Om 2019)	2019	classification
MFMC	Modified fuzzy min–max neural network for data clustering (Liu et al. 2017)	2017	clustering
MFMNN	Modified fuzzy min-max neural network for two-stage pattern classification (Quteishat and Lim 2008)	2008	classification
MFMNN-GA	Modified fuzzy min-max neural network with genetic algorithms (Quteishat et al. 2010)	2010	classification
MILP	Mixed integer linear programming model-based hyperbox classifier (Xu and Papageorgiou 2009)	2009	classification
MLF	Multi-level fuzzy min-max neural network (Davtalab et al. 2014)	2014	classification
MMM-BL	Modified fuzzy min-max neural network with a new batch learning algorithm (Meneganti et al. 1998)	1998	classification
RFMNN	Refined Fuzzy Min-Max Neural Network (Al Sayaydeh et al. 2020)	2020	classification
RMILP	Refined mixed integer linear programming model-based hyperbox classifier (Yang et al. 2015)	2015	classification
SFMN	Stochastic fuzzy min-max neural network (Likas and Blekas 1996; Likas 2001)	1996; 2001	classification; reinforcement
SS-FMM	Semi-supervised classification method based on fuzzy min–max neural network (Liu et al. 2020)	2020	semi-supervised classification
TDFMM	Top down fuzzy min-max neural network (Tagliaferri et al. 2001)	2001	classification
TDFMMR	Top down fuzzy min-max regressor (Tagliaferri et al. 2001)	2001	regression
TEH-GFMMNN	Tree ensemble hyperboxes via general fuzzy min-max neural network (Eastwood and Gabrys 2011)	2011	classification
WFMM	Weighted fuzzy min-max neural network (Kim et al. 2004; Kim and Yang 2005; Kim et al. 2006)	2004-2006	classification; feature extraction

To better address the stability-plasticity dilemma, the family of adaptive resonance theory (ART) neural networks with the incremental learning ability was proposed by Carpenter et al. (1991, 1992). This type of neural network adapts the learned prototype only if the input pattern is similar enough to the prototype. An input sample which deviates too much in comparison with all existing prototypes is considered a new one, and the ART network will generate a new category with the input sample being the prototype. Inspired by the learning paradigm of the ART networks and to overcome their observed limitations, Simpson (1992) proposed the FMNN, which is a classification technique capable of generating nonlinear boundaries for splitting the input variables space into classes with any size and shapes (Gabrys 2002a). The training phase can be carried out with only one pass through the training samples, and it might be employed for pattern classification tasks (Simpson 1992; Davtalab et al. 2014). There are three main steps in the learning algorithm of the FMNN, i.e., hyperbox expansion, hyperbox overlap test, and hyperbox contraction. The details of the learning algorithm to build the FMNN can be found in Simpson (1992, 1993); Khuat et al. (2021b). This section only focuses on the analysis of its drawbacks and its improved versions.

2.3.1 Analysis of the Original Fuzzy Min-Max Neural Network

In the FMNN, the maximum size of hyperboxes (θ) is the most essential factor determining the number of generated hyperboxes. In general, the larger the value of θ, the fewer hyperboxes produced, and the network may show higher generality, but the overlapping regions increase, and the capability of capturing nonlinear boundaries between classes decreases. This also lowers the predictive accuracy of the trained network. A smaller θ leads to a larger number of generated hyperboxes and potential overfitting, thus reducing the generalization ability (Gabrys and Bargiela 2000; Davtalab et al. 2014). Hence, there is a trade-off between the generality and predictive accuracy of these networks.

Generally, the learning process of the FMNN is based on an online adaptation of the hyperboxes (Simpson 1992; Gabrys and Bargiela 2000). In other words, online learning provides the FMNN with the ability to create new classes and adjust the

existing classes without, generally, influencing information already captured in the network. In principle, this capability allows the FMNN to add new classes and adjust the existing ones without the demand for retraining (Gabrys and Bargiela 1999). Online adaptation is the main feature in a neural network learning to address the stability-plasticity dilemma (Grossberg 1980). Nevertheless, the traditional FMNN has not yet dealt with several issues as follows:

- *Expansion problem.* The winner expandable hyperbox in the conventional fuzzy min–max learning algorithms is randomly identified in the case of existing several winner hyperboxes

- *Hyperbox boundary problem.* Though an overlap eliminating process has been proposed, two newly contracted hyperboxes are still possible to be overlapped on the boundary edge due to the nature of the contraction formulas

- *Problem of contraction process.* The contraction approach inadvertently removes from the two overlapping hyperboxes some unambiguous part of the sample space, while simultaneously retaining some contentious part of the sample space in each hyperbox (Bargiela et al. 2004). This weakness should be analyzed in detail in subsection 2.3.2.

- *Data representation order problem.* The training step conducts a process of dynamic hyperbox creation, expansion, and contraction in the sample space when an individual training instance is presented. Therefore, the predictive accuracy depends on the presentation order of the training samples (Meneganti et al. 1998), and the approach is sensitive to outliers and noise (Gabrys 2002a). As a result, noisy data from real world applications might cause serious stability issue when deploying the model in practice

- *Maximum hyperbox size parameter sensitivity problem.* As analyzed above, the classification performance and generalization ability of the fuzzy min-max neural networks depends on the user-defined maximum hyperbox size threshold. This problem can be partly resolved by using an adaptive maximum hyperbox size mechanism as presented in Gabrys and Bargiela (2000).

- *Membership value problem.* The membership function used in the original learning algorithm assigns a relatively high membership degree to an input pattern being quite far from the cluster centroid as shown in Gabrys and Bargiela (2000). It is necessary to build a membership function that monotonically decreases with the increase in the distance from the input pattern to the cluster prototype.

In conclusion, the learning algorithm introduced by Simpson builds the connections starting by the first example and then adding new hyperboxes by a process of expansion/contraction. The algorithm faces two main problems: the difficulty to determine the threshold value and the dependence of classification performance on the presentation orders of input samples (Meneganti et al. 1998).

2.3.2 Variants Using Specialized Neurons for Overlapping Areas

The contraction manner in Simpson's FMMNN (Simpson 1992) has a drawback in which it removes from the two overlapping hyperboxes several zones of the sample space that was unambiguous while simultaneously maintaining some contentious part of the pattern space in each hyperbox. For instance, as shown in Figure 2.4, some part of the original hyperbox B_1 now are contained in B_2 after the contraction procedure and likewise the original hyperbox B_2 has an overlapping region with contracted hyperbox B_1. Hence, the overlap removal algorithm yields errors during training process. In some cases, additional hyperboxes can be generated to include the deleted portions of the original hyperboxes, for example, the use of adaptive hyperbox size with multiple data presentations in the GFMMNN (Gabrys and Bargiela 2000). However, this leads to the increase in the number of hyperboxes and the reduction in the interpretability of classification results. To tackle these problems, Bargiela et al. (2004) introduced an inclusion/exclusion fuzzy hyperbox classification network (IEFCN), where the overlapping regions of the sample space are explicitly represented as exclusion hyperboxes.

Unlike the classical FMMNN, the inclusion/exclusion fuzzy hyperbox classification model employs two kinds of hyperboxes, which are the inclusion and exclusion

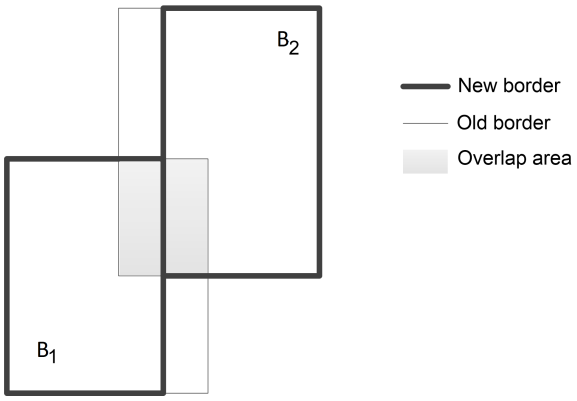


Figure 2.4 : Contraction of hyperboxes B_1 and B_2 with elimination along one coordinate

ones, and does not use contraction to eliminate overlaps. The inclusion hyperboxes cover the input samples of the same class, whilst the exclusion hyperboxes contain other overlapped input samples. By the use of combination of inclusion and exclusion hyperboxes, the learning phase is reduced to two steps (expansion and overlap test) (Davtalab et al. 2014).

The efficiency of the IEFMN is influenced when the number of exclusion hyperboxes is equivalent to that of inclusion hyperboxes, where percentage of data samples categorized as void class can become unacceptably high (Nandedkar and Biswas 2007a). Therefore, Nandedkar and Biswas (2004, 2007a) proposed a fuzzy min-max neural network with compensatory neurons (FMCN), which are generated dynamically during learning process, to tackle hyperbox overlaps and containment problems. The FMCN is also known as the Reflex Fuzzy Min Max Neural Network (ReFMN) since the concept of compensatory neurons (CNs) originates from the reflex system of human brain. The use of compensatory neurons contributes to removing the contraction process for the labeled hyperboxes and controlling membership in the overlapped region. The FMCN still maintains a single pass-through and online learning capability (Nandedkar and Biswas 2007a). The number of nodes in the input layer of the FMCN is equivalent to the number of dimensions of the input vector X. The intermediate layer neurons and output layer neurons are separated into two parts: 1) classifying neuron segment and 2) reflex section. The

classifying part is used for computing membership values for various classes.

The FMCN eliminates the usage of contraction procedure, thus it avoids errors resulting from contraction operation. FMCN may maintain the knowledge of the already trained samples more effectively in comparison with FMNN and GFMMNN due to the fact that already produced hyperboxes are not contracted. The accuracy of FMCN is better in single pass through the data (Nandedkar and Biswas 2007a) as it is able to approximate the complicated structure of data more properly thanks to the efficient capability of tackling hyperbox overlap and containment. Another benefit of FMCN is that it can evade the dependency of systems on the learning parameter compared to FMNN and GFMMNN. Furthermore, the FMCN is also robust to the noise.

In spite of above strength points, the FMCN also faces the following drawbacks. It does not implement a suitable membership function for overlap compensatory neurons, thereby it is unable to precisely classify high percentage of patterns present in overlapping areas. Furthermore, FMCN only deals with simple and containment overlaps, so there would be types of overlaps such that the algorithm cannot remove them, e.g. two hyperboxes crossing each other. Another case is that hyperboxes including just one point, when contained in any hyperbox representing another class, will not expand in next steps (Davtalab et al. 2014). In the learning algorithm, if any overlap between expanded hyperbox and other hyperboxes representing other classes exists, a compensatory neuron is added to the network. Hence, duplicate nodes are more likely to be generated. To overcome these disadvantages, Davtalab et al. (2012) invented a new fuzzy min-max model based on fuzzy min-max neural network with modified compensatory neurons. This classification model is also an online, single-pass and supervised learning method, but the authors made several changes in the structure of FMCN and training phase to reduce time and space complexity. The new structure reduces generating and storing useless hyperboxes during the training process, so it results in a faster classifier.

To construct a system being capable of learning from the mixture of labeled and unlabeled data like the GFMMNN, Nandedkar and Biswas (2006a) introduced

a modified version of RFMN by adding floating neurons (ReFMN-FN). Like the original ReFMN, the ReFMN-FN also uses compensatory neurons to uphold the hyperbox dimensions and control the membership in the overlapping areas among hyperboxes representing different classes. Floating neurons are added to the network to keep the unlabeled hyperboxes and stop them from contributing to the classification. The ReFMN-FN may be trained by two methods, which are supervised learning and semi-supervised learning (Nandedkar and Biswas 2006a). To handle labeled, unlabeled and partly labeled data sets, Nandedkar and Biswas (2007b) proposed a general reflex fuzzy min-max neural network (GRFMN). However, the input pattern to this network is in numeric form.

In the later studies, Nandedkar and Biswas (2009, 2008, 2006b) extended this network for both granular data and numeric data by representing input sample as a hyperbox. GRFMN does not make use of contraction process, but instead, a reflex mechanism is used to tackle the hyperbox overlap and containment issues. The GRFMN enables the unlabeled hyperbox to overlap with labeled hyperboxes, and it only performs a contraction process if there is an overlap between two unlabeled hyperboxes (Nandedkar and Biswas 2007b). The GRFMN implements the same contraction method as the GFMMNN (Gabrys and Bargiela 2000) for the overlap amongst unlabeled hyperboxes, in which overlapped hyperboxes are contracted along a dimension with minimal overlap.

Zhang et al. (2011) introduced a data core based fuzzy min–max neural network (DCFMN) with new structure and learning algorithm. Like FMCN (Nandedkar and Biswas 2007a), DCFMN also removes the contraction procedure to reduce classification errors. While FMCN employs the overlapped compensatory neuron and the containment compensatory neuron to deal with the overlapping region among hyperboxes from different classes, only one type of neurons, i.e., overlapping neurons, is implemented to handle the overlap and containment issue in DCFMN. As for FMCN, the extended hyperbox is not overlapping with any prior hyperbox of different class, so the number of hyperboxes in FMCN may be large and this results in a complex network and waste of time. To overcome this drawback, DCFMN allows

the hyperboxes to be expanded to overlap repeatedly with the previous hyperboxes (Zhang et al. 2011). Hence, the number of hyperboxes in DCFMN is lower than that in the FMCN and it uses less computation time. Furthermore, only three kinds of overlap were handled in FMCN, whereas the DCFMN tackles all kinds of overlap (Zhang et al. 2011). In addition, the authors proposed a new membership formula of classification neurons formed based on the characteristics of data and the impact of the noise which makes DCFMN more robust. A novel training and classifying algorithm was also introduced to make the resulting classifier faster and more accurate. In spite of these advantages, DCFMN cannot accurately classify high proportion of patterns being located in the overlapping areas, and also is unable to classify all learning examples properly either (Davtalab et al. 2014). Moreover, the architecture proposed by Zhang et al. (2011) can only handle numeric data. This method can be extended as GRFMN to deal with granular data.

As mentioned above, FMCN and DCFMN deploy compensatory neurons to deal with the overlapping regions among hyperboxes of different classes. Nonetheless, these networks are unable to classify a high ratio of patterns positioned in overlapping areas accurately and also face several structural issues in their training algorithms leading to increased complexity but decreased efficiency (Davtalab et al. 2014). To increase the classification accuracy in the boundary areas, Davtalab et al. (2014) proposed a multi-level fuzzy min-max neural network (MLF) implementing a multi-level tree structure to construct a homogeneous cascading classifier. Hyperboxes with different sizes are yielded in various network levels to tackle the overlapping issue. Each node in the network is a subnet as well as an independent classifier that can classify patterns belonging to the defined area of sample space. Therefore, MLF is able to classify samples belonging to boundary regions with a high accuracy. Experimental results of Davtalab et al. (2014) indicated that MLF is more proper than original FMNN, GFMMNN, DCFMN, IEFCN and FMCN, and the training process is faster than that of other types of FMM networks in most cases.

Although FMCN and DCFMN do not use the contraction procedures, they supplement numerous new neurons to the fuzzy neural network, which turns the neural

network out to be more complex than before. To cope with all issues, Ma et al. (2012) introduced a modified data-core-based fuzzy min–max neural network (MDCFMN) with new learning mechanism. In that algorithm, the contraction process was removed without adding any new neuron to the network (Ma et al. 2012). The learning algorithm for the MDCFMN only consists of one procedure: identify if hyperboxes need to expand or not and compute the center of gravity of data in the same hyperbox. The modified DCFMN uses a recursive procedure to determine the average value of data located in a hyperbox. Although the authors indicated that their proposal does not yield an special node for each overlapped area, they employed such nodes implicitly (Forghani and Yazdi 2015).

2.3.3 Modified Variants Without Using Specialized Neurons for Overlapping Areas

Two main shortcomings of the learning algorithm developed by Simpson are the sensitivity to the input data presentation order and the difficulty in finding the maximum hyperbox size. Hence, Meneganti et al. (1998) introduced a novel learning algorithm while keeping unchanged the structure of the fuzzy min-max neural network. The algorithm begins with building the minimum size hyperboxes covering all patterns of the same class, in which each class is represented by only one hyperbox. Then, a five-step process including partition, decomposition, recomposition, removal, and expansion is used to minimize the number of hyperboxes while maximizing their dimensions. Authors also proposed a new method to split intersecting hyperboxes, which represent different classes, based on the similarity. A new membership function using Gaussian function was introduced to evaluate the similarity level of each pattern against each hyperbox. This learning algorithm makes the classification performance independent from the sample presentation, and the algorithm does not employ any threshold. In the later study, Tagliaferri et al. (2001) claimed a top-down fuzzy min-max (TDFMM) classifier by simplifying the complexity of the hyperbox splitting approach proposed by Meneganti et al. (1998). Based on the TDFMM algorithm, they developed a top-down fuzzy min-max regressor to deal with the regression problem. First of all, clustering methods are deployed

to assign the labels to input patterns. Next, the TDFMM algorithm is utilized to build hyperboxes. However, these are batch learning algorithms, and so they cannot accommodate the new patterns in real time.

The original Simpson's training algorithm for the FMMNN depends excessively on pattern presentation order and on position as well as size of the hyperboxes generated during training. These parameters impose the same condition on covering resolution in the whole input space. This results in reducing the generalization ability of the neural model. Aiming to tackle these inconveniences, two new learning algorithms were introduced by Rizzi et al. (1998), i.e., the adaptive resolution classifier (ARC) and pruning ARC (PARC) algorithms. In the improved version of these algorithms, Rizzi et al. (2000, 2002) suggested a feasible enhancement of the training process by utilizing a new cutting strategy in order to handle recursively hybrid hyperboxes, namely R-ARC/R-PARC. However, the hyperbox cutting and fusion operations are time-consuming. Hence, a method to improve the ARC algorithm is reduction of the total number of nets created during training. Rizzi et al. (1998) introduced a pruning version of ARC with two subsequent actions: doing an ARC operation without the fusion procedure and doing a pruning procedure.

The first significant extension of the original FMNN is a general fuzzy min-max neural network (GFMMNN) designed by Gabrys and Bargiela (2000). The GFMMNN is built on the basis of expansion and contraction concepts, and it can deal with both labeled and unlabeled data in a single algorithm. The architecture of GFMMNN almost resembles the original fuzzy min-max neural network topology except for two main alterations (Gabrys and Bargiela 2000). The first change is that the number of nodes in the input layer has been extended from n to $2n$. Primarily, it allows an input to be a hyperbox rather than a point in the n-dimensional space. The second modification is that the output layer has been added an extra node to represent all the unlabeled hyperboxes from the intermediate layer. This helps GFMMNN to deal with both supervised learning and unsupervised learning. Other changes in the GFMMNN comprise a new fuzzy hyperbox membership and adaptive modification of the maximum hyperbox size. Two different algorithm

types have been proposed to train the GFMMNN (Gabrys 2004): an incremental learning (Gabrys and Bargiela 2000) and an agglomerative learning (Gabrys 2002a). The incremental (online) learning is a dynamic hyperbox expansion and contraction procedure where hyperboxes are generated and tuned in the sample space after every presentation of a training instance (Gabrys 2002a). Nevertheless, this learning strategy, at all incremental learning approaches, results in the input-output mapping depending on the order of presentation of the training input samples, and the algorithm is sensitive to outliers and noise. The overlapping hyperboxes are also another undesired result derived from the dynamic feature of the algorithm. The agglomerative learning introduced in Gabrys (2002a) is an alternative and a complimentary method to the incremental learning for an off-line training process done on a finite training sets. While the incremental learning is more appropriate for online adaptation and is able to tackle large training data sets, agglomerative learning represents robust behaviour in presence of outliers and noise as well as insensitivity to the order of training samples presentation (Gabrys 2004). However, a drawback of the agglomerative learning algorithms is the long training time, especially for the large-sized training data.

With the aim of generating good classifiers, Gabrys (2004) analyzed five different algorithm-independent model generation schemes from the GFMMNN using the agglomerative learning algorithm. These methods include construction of the predictive models using full training data set, employing a k-fold and multiple 2-fold cross-validation along with different pruning procedures, and an ensemble of various GFMM classifiers at the decision or model level. He claimed that the method of generating the GFMM model using base learning algorithms without any hyperbox pruning steps is swift and able to adapt to the changing environment. However, it is likely to be overfitted and exhibit a poor generalization performance. In the case that the classifiers can be built in the off-line modes, the techniques based on the combination of multiple cross-validation and pruning procedures or the ensemble of base classifiers tend to yield a better classification accuracy.

An interesting property regarding the use of hyperboxes as inputs for the GFMMNN

is the capability of handling missing values in the data. The replacement of missing values with estimated values such as the mean of all patterns may make data set no longer a good representation of the problem and may result in bad solutions (Berthold and Huber 1998). Gabrys (2002c) introduced a method of dealing with missing data within the classification algorithm automatically by employing the GFMM neural network model. If the value of the *i-th* feature is missing, the lower bound of hyperbox on the *i-th* dimension is assigned to one and its upper bound receives a value of zero. This operation makes the hyperbox membership associated with the missing feature to be one, and so it does not lead to a decrease in the overall membership value. This substitution also makes sure that the neural network structure would not be modified when handling missing dimensions. The only changes of the training algorithm relate to the way of performing the overlap test and the usage of the assumption that the missing features are possible to get all values (Gabrys 2002c). The overlap test is conducted after each hyperboxes updating operation for only hyperboxes in which their value of maximum point is larger than or equal to their value of minimum point on every dimension.

In another study, Kim et al. (2004); Kim and Yang (2005) proposed a weighted fuzzy min–max neural network (WFMM) for the sample classification and feature extraction problems. They introduced a new membership function and expansion scheme by considering a weight factor for each dimension of a hyperbox. This improvement makes the WFMM less sensitive to the unusual or noisy features in a data set in comparison with the FMNN (Kim et al. 2004). Therefore, the WFMM may handle better data sets containing highly uneven distribution of features or noisy features (Zhang et al. 2011). The architecture and learning algorithm of the WFMM are similar to the FMNN except for some changes in the hyperbox membership function and expansion mechanism. In the later research, Kim et al. (2006) introduced an enhanced version of the WFMM with changes in the connection weights of features, membership function, and a new contraction method including the weight updating mechanism.

With the aim of handling a number of drawbacks of the original FMNN, Mo-

hammed and Lim (2015) introduced an enhanced fuzzy min–max neural network (EFMNN). They retained the structure of original FMNN and only modified the training algorithm. The authors employed a new constraint, in which each dimension of the $i\text{-}th$ hyperbox is examined individually to identify if it is larger than the maximum hyperbox size θ. This mechanism is the same as that proposed in the GFMMNN (Gabrys and Bargiela 2000). They introduced nine cases for overlap test and corresponding contraction operations instead of four cases in the original FMNN.

In the later study, Mohammed and Lim (2017a) claimed that the expansion rule limitation has still not been resolved thoroughly in the EFMNN. Based on the analysis of drawbacks, Mohammed and Lim (2017a) proposed a novel approach, known as the K-nearest hyperbox expansion rule (KNEFMNN), to decrease the network complexity by reducing the number of small hyperboxes within the surrounding areas of the winning hyperbox during the training phase. Besides the K-nearest hyperbox selection rule, Mohammed and Lim (2017b) introduced a useful strategy to deal with noise, i.e., pruning. This strategy aims to determine and delete hyperboxes giving low accuracy, which are regularly produced because of outliers or noise. Although the EFMN and its improved versions outperform the original FMNN, the use of the contraction step for the training process may lead to classification error as analyzed in subsection 2.3.2.

While the contraction process is more likely to lead to classification errors, the formulation of a special node for each overlapped region in training stage may lead to the complexity in the architecture of neural networks and increase in time and space complexity. There is an interesting finding that the misclassification probability in the case of both training and test patterns being from identical probability distribution is minimized if the classifier has symmetric margin (Vapnik 2000). However, none of above methods is capable of classifying patterns positioned in overlapped regions with symmetric margin. Hence, Forghani and Yazdi (2015) proposed a fuzzy min–max neural network with symmetric margin (FMNWSM). In that approach, only hyperbox expansion procedure is carried out in training phase. It does not

find overlapped regions, does not perform the contraction process to eliminate overlapped areas and does not yield any special hyperbox for overlapped regions. Due to using only expansion procedure, the training time of FMNWSM is lower than that of kinds of conventional FMNNs such as FMNN, GFMMNN, FMCN, and DCFMN. In real-world applications, data can be infected by noise, which results in formulating hyperboxes with wrong boundaries. To cope with the impact of noise on classification results, Forghani and Yazdi (2015) proposed an extended version of FMNWSM, called EFMNWSM, by constructing a new membership function.

In an attempt to reduce the "black-box" property of WFMM, Kim and Lee (2013) introduced a rule extraction technique for sign language recognition. Authors used a combination of the weighted fuzzy min-max model (Kim et al. 2006) and new rule extraction technique for the pattern classification step. The training process of modified weighted fuzzy min-min (MWFMM) neural network comprises only two procedures: creation and expansion of hyperboxes. The connection weights are constantly updated during the learning phase. These values reflect the relevance factors between the features and the hyperboxes. Hence, the network can deal with the hyperbox overlapping problem without performing the hyperbox contraction process (Kim and Lee 2013).

Though hyperbox fuzzy sets can explain their predictive results based on rules extracted directly from the min-max values, the interpretability of classification is usually not friendly for users. It is because the rule sets become extremely complex in case of a large number of hyperboxes and high dimensions. Therefore, it is desired to construct the rule extraction methods from hyperboxes to form a compact rule set, which is capable of accounting for the predictive results. As a result, Quteishat and Lim (2008) introduced a two-stage pattern classification system using a modified FMNN in the first stage and a rule extraction procedure in the second stage. To enhance the prediction performance, however, in the later work, Quteishat et al. (2010) conducted several preprocessing steps before extracting rules from the modified FMNN, i.e., open hyperbox generation and genetic algorithm (GA) rule selection. This method is abbreviated MFMNN-GA. However, their proposed clas-

sifiers have lost single pass-through online adaptation power of FMNN (Forghani and Yazdi 2015).

In another study, Sonule and Shetty (2017) proposed an enhanced fuzzy min–max neural network model with a rule extractor based on ant colony optimization (EFMNN-ACO) for classifying the data samples and decision making by rule lists. The output of the EFMNN (Mohammed and Lim 2015) is used as input of a rule extractor based on AntMinerPlus algorithm to build a graph. Then, the algorithm will perform a process of extracting the rule-list and pruning rules by finding the paths on the graph. The experimental result of Sonule and Shetty (2017) showed that the quality of rules is consistent and the number of obtained rules is reduced due to the optimization algorithm. One of the strong points of this method is that it can optimize simultaneously a list of rules instead of separate rules. However, training time of the system is longer than other classifiers since there are more cases considered and the construction of graph in the learning algorithm takes a long time.

As mentioned in subsection 2.3.1, the original fuzzy min-max neural network encounters several problems in the expansion and contractions steps as well as the overlapped boundary after doing contraction. Hence, Liu et al. (2017) proposed a modified fuzzy min–max neural network for data clustering (MFMC) to deal with those issues. Different from the original fuzzy min–max learning algorithms which only separate the overlapping area into half, Liu et al. (2017) introduced the method to reserve the hyperboxes with higher performance when performing the contraction operation. The hyperboxes which include more data with smaller size are regularly considered to be the ones with higher performance. The reservation ability maintains the good structure of the whole learning algorithm, avoid disturbing the size of hyperboxes because of noise, and refine the robustness for the algorithm (Liu et al. 2017). Although the authors made some changes to handle drawbacks of the original fuzzy min-max clustering neural network, there are still some existing unsolved problems. The use of only four test cases for the overlapping checking is not adequate to verify all situations happening in the practice as described in Mohammed and Lim (2015). The hyperbox selection rule only solves the case of many winner hyperboxes.

If the ultimate winner hyperbox cannot be expanded, it still creates a new hyperbox to contain the input training sample. This issues can lead to the generation of numerous hyperboxes with the small size, which causes the network complexity problem.

Azad and Jha (2016) argued that the series of expansion and contraction steps lead to the change in the sizes of hyperboxes and affect the performance of the predictive model. Hence, authors proposed to use the genetic algorithms to optimize the min-max values of hyperboxes generated by the original FMNN. In later work, Azad and Jha (2017) used the particle swarm optimization instead of genetic algorithms, and they obtained a better performance.

To improve the performance of the original FMNN for clustering problems, Seera et al. (2015) proposed to integrate the centroid information of data samples into each hyperbox to construct an enhanced fuzzy min-max neural network for data clustering (EFMNNC). With the use of centroid, the operations of hyperboxes in the learning process are changed to force the centroid to be inside the hyperbox.

In most recently, there have been several noticeable improvements of the original FMNN. Upasani and Om (2019) proposed a modified version of the EFMNN, denoted by MEFMNN. They used a Manhattan distance measure to deal with the case that there are many winning hyperboxes with the highest membership values in the learning process and classification phase. The author also proposed a solution of using GPUs to accelerate the MEFMNN. In another study, Sayaydeh et al. (2019) introduced a refined fuzzy min-max neural network (RFMNN) with a new general formula for the overlap test procedure and a new hyperbox contraction operation. In many real world applications, one usually faces the lack of labeled data, while the unlabeled data are often abundant. Hence, Liu et al. (2020) proposed a semi-supervised learning algorithm based on the GFMMNNN (SS-FMM) to use both the labeled and unlabeled data to train a classifier. They changed the network structure of the original FMNN with a staged feedback to label the unlabeled data. Additionally, the authors designed a hyperbox pruning process to maximize the use of unlabeled data as well as controlling the number of generated hyperboxes. Similarly,

Ma et al. (2021) introduced an evolved fuzzy min-max neural network to effectively learn from unknown labeled data (FMM-ULD) using the MFMC (Liu et al. 2017) to cluster the unknown data by using a novel threshold function.

An attribute of the fuzzy min–max neural networks mentioned above is that all the input variables for training and classifying processes are continuous numerical values (Castillo and Cardenosa 2012). When the categorical variables are presented, they can be substituted by numerical values and treated as continuous values. However, there is no meaningful correspondence between the continuous values created by this method and the original categorical ones (Brouwer 2002). Hence, it is expected to form a new method for handling categorical variables. Castillo and Cardenosa (2012) proposed a new method to extend the GFMMNN's inputs for dealing with discrete variables (GFMMNN-CD1) by formulating the distance between the categories of categorical variables. The authors defined a new membership function for both numerical and categorical variables. They also extended the architecture of the GFMMNN to include both numerical and categorical inputs. In the later study, Shinde and Kulkarni (2016) introduced another method to refine the GFMMNN for categorical input data (GFMMNN-CD2). They proposed a simpler architecture by adding a set of binary strings, where each binary string represents a given discrete attribute, to each hyperbox fuzzy set of the network. The authors also introduced a new membership function with the logical bitwise 'and' and 'or' operators conducted on two binary strings for categorical variables. The learning algorithm is similar to the GFMMNN with only small changes to accommodate discrete data.

2.4 Hyperbox based Hybrid Machine Learning Models

Mirzamomen and Kangavari (2016) proposed a fuzzy min-max decision tree (FMMDT), in which each internal node of the decision tree includes a contraction-less fuzzy min-max neural network (CLFMNN). The strong points of the decision tree and the fuzzy min-max neural networks are integrated into a hybrid model to mitigate the drawbacks of each algorithm. The shortcomings of the FMNN, i.e., the dependency on the maximum size of hyperboxes and the performance degradation

due to the contraction process, can be tackled based on the hierarchical structure of the decision tree. Unlike conventional decision trees where a single attribute is chosen as the split test, by using of the FMNN, each non-leaf node is split non-linearly based on multiple attributes, and thus it can capture the complicated structures in the data as well as forming decision trees with much smaller depth. The FMMDT is grown according to the top-down method starting from the root node containing all training patterns. Samples in each node are then partitioned recursively until all or most of the samples are of the same class. Splitting a node in the FMMDT is performed by training the CLFMNN on the input sample of the node and using its output for splitting operation. However, the FMMDT constructed according to this method is a batch decision tree learner for the static context (Mirzamomen and Kangavari 2017).

To learn from continuous attributes in data streams, which is more challenging compared to the static context, Mirzamomen and Kangavari (2017) proposed an evolving fuzzy min–max decision tree (EFMMDT) learning algorithm, where each decision node of the tree includes a concept adapting contraction less (CACL) fuzzy min–max neural network. The authors used trainable split checks relied on multiple attributes for each decision nodes by embedding the CACL fuzzy min–max neural networks with concept-drift processing mechanisms to it. Hence, the decision tree is able to be easily adapted to the new concept through employing a forgetting mechanism and training on new information (Mirzamomen and Kangavari 2017). The statistics record constructed on the basis of the sliding window of recent samples consists of hit rate, which represents the number of times a hyperbox has included a new sample, and min-max points describing the minimum and maximum coordinates computed from the matched training samples. The statistics-based updating mechanism frequently shrinks the hyperboxes following the presentation of recent training patterns. Whenever a new training sample is used, the algorithm traverses the tree from the root to a leaf to separate the node using Hoeffding bound (Hulten et al. 2001) for electing the features.

In another study related to combination structures of network and tree, Seera

et al. (2012) proposed a hybrid model comprising the FMNN and the classification and regression tree (CART) (Breiman et al. 1984), namely FMM–CART, for fault detection and diagnosis of induction motors. Although FMNN has online learning capabilities with one-pass training, it does not have the ability to explain its predictions in a user friendly form and the capability of handling categorical data. Meanwhile, CART is able to explain its predictive results with rules as well as handling both numerical and categorical data, but it is less flexible with regard to learning from data patterns. The hybrid model can tackle the drawbacks of both models with the ability to learn from data samples and produce rules for explanation of its predictions simultaneously. The hyperboxes of FMNN form an input data set to construct a tree based on the CART procedure. In the later study, Seera and Lim (2014) extended FMM-CART by proposing modifications of the CART algorithm and FMNN. In the learning algorithm of FMNN, a confidence factor for each hyperbox is identified. This factor contributes to determining hyperboxes used very often and being generally accurate in prediction, or hyperboxes seldom employed but highly accurate. To improve the accuracy of a classification tree, the authors provided each class of the decision tree with a confidence level, called class weight, using the values of the FMM hyperboxes' confidence factors (Seera et al. 2012).

The fuzzy min–max clustering neural network (Simpson 1993) does not require providing the number of clusters in advance since it offers capability of online learning, and its number of clusters can be grown incrementally according to the data distribution. However, the FMM clustering neural network is unable to generate comprehensible rules to explain its clustering results. In contrast, the clustering tree (CT) is capable of explaining the underlying cluster structures. Therefore, Seera et al. (2016) combined the modified fuzzy min-max neural network for clustering (Seera et al. 2015) with the clustering tree to create a new model, called FMM-CT model, for tackling data clustering problems with the strengths of online learning and rule extraction. To further improve the clustering performance of FMM-CT, an ensemble model of clustering trees (ECTs) was introduced by Seera et al. (2018). An ensemble model is able to deal with noise and outliers more effec-

tively in comparison with individual clustering trees (Fauber and Schwenker 2015). It is capable of generating a good balance between good and weak models in the final ensemble model. The authors used the ECT model incorporating multiple CTs by implementing the bagging method with random feature selection, which may decrease the variance by sub-sampling the trees and minimize the error rate (Seera et al. 2018). The majority voting mechanism is then employed to aggregate results of base clustering tree. To evaluate the validity and quality of constructed clusters, authors employed the Cophenetic Correlation Coefficient metric (Mirzaei and Rahmati 2010) in combination with the centroid of each cluster.

Ensemble classifiers using bagging are able to significantly enhance the classification performance, but it makes the classifier system complex, in which computational time and a large memory are required for attaining the final classification outcomes. The classifier ensembles also face loss of transparency the decision making process since an ensemble of numerous individual classifiers would be much more difficult to comprehend by users. In order to reduce the final predictor complexity while preserving good classification performance, Gabrys (2002b) proposed the way of combining the hyperbox fuzzy sets of base models, i.e., GFMMNNs created during repeated 2-fold splitting of the training data, rather than their decisions. The combination of hyperboxes is straightforward and has already been implemented in the agglomerative learning algorithm (Gabrys 2002a). The hyperbox fuzzy sets from different component machine learning models are incoporated as inputs to the agglomerative training algorithm with the aim of removing the redundant hyperboxes and adding or refining hyperboxes covering the regions near the class borders or overlapping areas. The experimental results of Gabrys (2002b) indicated that the combination at the model level gives much better performance than incorporation at the decision level or by utilizing individual component classifiers learned from only part of the training set. However, training time increases compared to combining at the decision level.

One of the main issues ensemble models encounter is the lack of a simple structure (Nugent and Cunningham 2005). This can make the classification systems become

a 'black box' model without capability of explaining their predicted results. It is desired to gain the advantage of incorporating classifiers whilst maintaining a classifier with a relatively simple architecture. Motivated by this goal, Eastwood and Gabrys (2011) proposed a way of building a model level combination scheme in which an ensemble of decision trees is employed to build a set of hyperboxes covering the input instances. These hyperboxes are then used as inputs to form a single, relatively simple classifier within the general fuzzy min-max framework (Gabrys and Bargiela 2000). By constructing the model on robustly labeled hyperbox instances, instead of directly from the data, the classification accuracy of the final model can be significantly enhanced on most data sets. Although experimental results showed that performance of tree ensemble hyperboxes via GFMMNN generally does not reach that of a full ensemble technique, this approach is still a good choice in cases when simplicity of model is a desirable factor (Eastwood and Gabrys 2011).

2.5 Other Hyperbox based Machine Learning Models

Abe and Ming-Shong (1995) proposed an effective method to form a fuzzy classification system by extracting fuzzy rules directly from numerical input and output data through hyperbox representation. Fuzzy rules are determined by activation hyperboxes, which represent the existence region of data for a given class, and inhibition hyperboxes preventing the existence of data in corresponding activation hyperboxes. Two kinds of hyperboxes are constructed recursively. However, the recursive method of forming activation and inhibition hyperboxes to resolve overlapping regions between two classes each time can lead to a complicated architecture for the classifier when tackling difficult classification tasks. Based on the hyperbox construction method of Abe and Ming-Shong, Thawonmas and Abe (1997) proposed a feature selection approach using the analysis of class regions created by hyperboxes. The degree of overlap in the different class areas is defined as the exception ratio to make a measure for feature assessment. The feature with minimum exception ratio is remove permanently.

To use the FMNN as an action selection network in the reinforcement learning

problems, Likas and Blekas (1996) introduced a stochastic fuzzy min-max neural network, where each hyperbox holds a stochastic automaton. The purpose of the learning process of the stochastic FMM network is to adjust the position and border of each hyperbox and the probability vector of each stochastic automaton (Mohammed and Lim 2017a). However, this method has several disadvantages. Firstly, all actions in a randomly generated hyperbox have equal probability of occurring, hence, it is unable to support a certain specific action. Secondly, there is an immediate transition from stochastic to deterministic when a rewarded action is chosen, and this may causes several problems as the rewarded action may not be the best one. Another drawback is that the modification of the action label of a given hyperbox is impossible (Likas 2001). Finally, the creation method of a new random hyperbox within the original hyperbox and the shrinking of both of them to prevent the appearance of overlapping regions result in the formulation of an immoderate quantity of small volume hyperboxes and cause the difficulty in the adaptation of learning algorithm. To cope with these issues, in the extended version, Likas (2001) proposed an improved stochastic fuzzy min-max network, where all hyperboxes are taken into account as random ones, and each of them holds a stochastic automaton. The key role of the automaton is to determine the degree of randomness in the process of choosing suitable actions.

Xu and Papageorgiou (2009) introduced a mathematical programming-based classifier modelling classification boundaries as hyperboxes. Training process in the proposed method tends to construct for each class a number of hyperboxes covering as many training instances as possible. The optimal position and dimension of each hyperbox is identified by a mixed integer linear programming model (MILP) aiming to minimize the total number of misclassifications. In this model, any two hyperboxes of two different classes are not allowed to overlap because hyperboxes cover the unique sample of corresponding classes.

In each iteration of the mathematical programming-based classifier, a new multi-class prediction model needs to be solved regardless of the boundaries of hyperboxes achieved in the previous iteration. As a result, numerous training instances, which

are correctly classified, are repeatedly considered to formulate boundaries of hyperboxes and generate constraints in every iteration. In addition, a weak point of MILP models is that the computational time is relatively long, which results in low efficiency when the method is applied for large size data sets. Hence, Maskooki (2013) proposed a modified version of the training algorithm to significantly reduce the training time. The idea of their algorithm is to take advantage of the boundaries gained from previous iterations and remove the correctly classified samples from the training set in each iteration rather than utilizing whole data set for finding new boundaries. Hence, several constraints corresponding to accurately classified patterns are not created and thus the number of binary variables needed to be considered is decreased through iterations. In this training algorithm of Maskooki, only the misclassified instances from different classes are passed to the next iteration. By this method, several constraints are reduced and therefore the number of binary variables is decreased through each iteration. Another type of improvement for the MILP model was proposed by Yang et al. (2015), where they aimed to refine the quality of the formed hyperboxes. They introduced two new proposals to enhance the performance of the hyperbox based classifier. They firstly extended Xu and Papageorgiou's work by incorporating a sample reweighting scheme, where higher weights are assigned to misclassified patterns included in other hyperboxes to adjust the model aiming to consider those difficult patterns in the next iteration. Moreover, to reduce high computational expense of the original model, they proposed a data space splitting technique to partition the training instances into two disjoint areas. The model is then resolved in this new space.

Most classical statistical models make assumptions on underlying data, their distributions, independence, stationarity, etc. However, such assumptions are usually unrealistic in the real-life applications (Grzegorzewski 2013). Furthermore, the traditional statistical techniques cannot generate ambiguous outputs if their inputs exhibit uncertainty, vagueness, imprecision, or fuzziness. Motivated by the practical demands for simplification, low cost, approximation, and the tolerance of uncertainty in information processing, the granular computing (Bargiela and Pedrycz

2003) was proposed with the aim of the construction of intelligent systems with more human-centric view. Peters (2011) introduced a simple and intuitive method, called granular box regression analysis, to establish a fuzzy granulation generalization of a function between several independent variables and one dependent variable in the context of granular computing by using hyperboxes. Reyes-Galaviz and Pedrycz (2015) introduced a new technique to form a granular model for clustering in which information granules are represented as hyperboxes.

Most studies on the construction of hyperboxes are based on the original fuzzy min-max neural network proposed by Simpson (1992). Hence, hyperboxes are limited by an allowable maximum size and non-overlapping hyperboxes conditions. While the maximum size can lead to a complex model with a large number of small hyperboxes, the contraction process to remove overlaps causes loss of included data. Therefore, Reyes-Galaviz and Pedrycz (2015) proposed a novel technique for the construction of hyperboxes, where they deployed conditional Fuzzy C-Means (FCM) (Pedrycz 1998) to establish direct associations among information granules, and then utilize them to construct hyperboxes. Experimental results indicated that the clustering performance of this approach increases when data have the large number of features and instances compared to the original fuzzy min-max clustering neural network.

Palmer-Brown and Jayne (2011) proposed a hyperbox neural network (HNN) algorithm for classification, in which each class is associated with only one hyperbox and a single neuron. Each hyperbox covers and presents the ranges of attribute values of samples belonging to the same class. If a new sample falls in only one hyperbox, it will be classified immediately to the class label of that hyperbox. In contrast, if the pattern is located in the overlapping region among hyperboxes, the neurons of these hyperboxes are used to classify the sample. Eastwood and Jayne (2014) extended the hyperbox neural network by proposing a piece-wise sigmoid adaptive activation function to substitute the piece-wise linear function in Palmer-Brown and Jayne (2011). They also introduced a combination of the supervised hyperbox and neurons with unsupervised clustering. However, the use of only one

neuron per each class has one potential weakness that it might only express class distribution within a hyperbox varying in only one single direction (Eastwood and Jayne 2014). Therefore, Eastwood and Jayne (2014) proposed a change to the above method, in which they performed clustering on the data belonging to each hyperbox to form data clusters within each hyperbox. A single sigmoid neuron is then associated with each cluster and is trained on only the data in that cluster. Building more than one neuron for each hyperbox based on clusters allows to handle data sets where the different classes have many overlapping areas, and can model the class distribution within a hyperbox in more than one direction.

Park et al. (2014) suggested the development of a hyperbox classifier with hierarchical two-level structure on domino extension from seed hyperboxes. This classifier comprises a core structure built on a basis of a set of hyperboxes and secondary structure constructed on a basis of fuzzy sets. The core structure is formed by the domino extension of seed hyperboxes. As for the domino method, the feature space is discretized, and then a hyperbox chosen as a seed in the new space grows up to formulating a cluster. The secondary structure is built by implementing a Hausdorff distance (Olson 1998) between a group of hyperboxes and a sample.

The topological information is usually ignored in conventional clustering methods. This shortcoming is effectively tackled by using clustering techniques based on hyperboxes to capture the distribution of data on the feature space. Hence, Ramos et al. (2009) proposed a hyperbox clustering with ant colony algorithm (HACO) for clustering the unlabeled data by near-optimally placing hyperboxes in the feature space. Hyperboxes, which are frequently smaller than the number of patterns, are sought for by using the ACO algorithm (Dorigo and Stutzle 2004) and then clustered utilizing the nearest-neighbor approach. The obtained result from HACO is a vector of hyperboxes for clustering data which can be used as machine learning models. To refine this model, Ramos et al. (2008) proposed a second version of hyperbox-based machine learning algorithm with ACO (HACO2) by deploying the ant colony algorithm to evolve the geometry of hyperboxes in the feature space to better cover the data in the class. The output of HACO can be used as input of

HACO2 to reshape the hyperboxes. Therefore, HACO can be considered as an appropriate initialization step instead of random initialization of hyperboxes so that HACO2 refines the structure.

2.6 Discussion and Research Directions

2.6.1 Discussion

Hyperbox fuzzy sets possess many attractive characteristics to build highly efficient predictive models. This chapter gave an overview of properties and characteristics of existing hyperbox-based machine learning algorithms.

The existing studies taking advantage of the hyperbox representations for the construction of predictive models were categorized into three groups. The first group is the original FMNN and its variants. The second one is the combination of the FMNNs and tree-based learning algorithms or the formulation of an ensemble model. The final group includes models that only use pure hyperboxes to cover training samples without forming network or tree architectures.

The idea of using hyperbox fuzzy sets for machine learning models was proposed in the ART neural networks (Carpenter et al. 1991), and then it was improved to form the FMNN in Simpson's work. The FMNN has many strong points such as online learning ability, soft and hard decisions, construction of nonlinear decision boundaries, *etc.* However, this type of neural network still faces many drawbacks such as expansion and contraction problems, the dependence on the data presentation order, parameter sensitivity problem, and issues related to the membership function.

As a result, several improvements have aimed to form new learning algorithms for hyperbox-based neural networks such as using new rules to select hyperboxes for expansion, adding or modifying hyperbox test rules. Building a unified framework for FMM clustering and classification neural networks as in GFMMNN provided a significant enhancement to the original version. Many researchers have targeted eliminating the hyperbox contraction process by implementing inclusion/exclusion

neurons, compensatory neurons, overlapping neurons, using a multi-level network structure or some heuristic rules to choose the suitable hyperbox for samples located in the overlapping region among hyperboxes. To expand the hyperbox-based machine learning models for dealing with categorical data, several studies have taken categorical variables into consideration when building new membership functions and network architectures.

In addition to different types of hyperbox-based neural networks, many studies have taken advantage of online learning ability of FMNN to combine with rule extractors such as decision tree, clustering tree, and classification and regression trees. These types of hybrid models are capable of generating explanatory rules for their outputs. Some other works have concentrated on evolving and optimizing the positions of hyperboxes from the initial set of hyperboxes by using nature-inspired algorithms such as differential evolution, GA or ACO.

2.6.2 Research Directions

In spite of many advancements in existing literature, there are still a number of open problems having not been fully tackled in current studies on hyperbox-based machine learning algorithms, and ones which require more efforts on resolving them in the future work. Some potential research directions which are tackled in this thesis can be shown as follows. The remainder of potential research directions is presented in Section 9.2 in Chapter 9.

- *Improving the robustness of online learning algorithms.* As discussed in subsection 2.3.2, the contraction process in the learning algorithms of fuzzy min-max neural networks can lead to the disturbance of existing hyperboxes, and thus the performance of learning algorithms can dramatically decrease, especially in the case of building large-sized hyperboxes. Therefore, Objective 2 in this thesis is proposed to build more robust online learning algorithms for the GFMMNN. The details of the proposed method are presented in Chapter 4.

- *Learning from mixed-attribute data.* In the literature, there have been only two studies aiming to make the FMNNs able to learn from the mixed-attribute

data. Therefore, Objective 2 in this thesis focuses on developing the effective learning algorithms for the FMNNs to deal with this issue. It will present a novel learning algorithm for the GFMMNN to tackle a classification problem on mixed-attribute data as shown in Chapter 5.

- *Accelerating learning algorithms.* As will be deeply analyzed in Chapter 3, current learning algorithms of (general) fuzzy min-max neural networks have high complexity. Hence, training time of these algorithm is long. Objective 3 of this thesis aims to deal with this issue. Two solutions to speed-up learning algorithms for the GFMMNN will be proposed in this study based on reformulation of learning steps for the execution of GPUs and using smart ways to reduce the number of hyperbox candidates based on the mathematical lemmas in Chapter 6.

- *Building simple interpretable hyperbox-based learning algorithms.* Transparent learning models are highly desirable for real work applications, especially in high-stake areas such as medical diagnostic, trading, and criminal justice. Although the FMNN and its variants are interpretable models, their transparency and interpretability will be significantly reduced when the complexity of models increases in the case of using the small value of the maximum hyperbox size. As a result, it is desired to propose different solutions to keep the learning models as simple as possible while still maintaining their high performance. Objective 4 in this thesis will deal partly with this issue by proposing a method to build learning models from different levels of granularity as presented in Chapter 7.

- *Generation of ensemble models.* It is really difficult to achieve acceptable classification results for big data sets with massive dimensions. Therefore, one may use hyperbox representations to select valuable features and train classifiers on different training subsets. The results of base classifiers or the hyperbox components in base predictors are then able to be combined following a certain method to output the final result or model. An effective way of generating ensemble models of hyperbox-based classifiers is highly desired,

which leads to Objective 5 of this thesis. This study will introduce a random hyperboxes model in Chapter 8, which is a simple and powerful technique to build the ensemble models from base learners using hyperbox representations.

2.7 Summary

This chapter partly addressed the proposed Objective 1 in Section 1.2 regarding conducting a comprehensive survey of existing studies on hyperbox-based machine learning algorithms. One of the interesting features of hyperbox-based models is their online learning capability, which meets the urgent demand for constructing effective machine learning approaches for real big data analytics. The hyperbox representation is especially suitable for big data applications such as image processing because the use of hyperboxes assists the processing and inference in the form of granules instead of handling individual points. This solution can significantly decrease computational cost in large systems. The real-world applications usually exhibit uncertain behaviors, so the input data of the machine learning algorithms are likely to contain both granular and crisp points or missing values. Hence, effective algorithms must be able to cope with uncertain and ambiguous data. The hyperbox representation method in the general fuzzy min-max neural network allows to handle both fuzzy and crisp input samples as well as inputs with missing values. An interesting characteristic of the general fuzzy min-max neural network is that it unifies both classification and clustering problems in the same framework. This feature endows the model with the capability of learning from unlabeled, semi-labeled, and labeled data. Another attribute of the practical systems is the continuous changing over time, so they require machine learning models capable of evolving to fulfill the operations without retraining. The hyperbox creation, extension, and contraction mechanisms using fast Boolean algebra help hyperbox-based machine learning algorithms evolve and adjust to adapt rapidly to the changes in the data. Another advantage of hyperboxes is that they can be used to build the rule sets for a given problem. This ability contributes to the increase in the reliability of the predictive results as well as making the algorithms friendly and transparent to end-users.

Chapter 3

General Fuzzy Min-Max Neural Network: Practices and Existing Issues

This chapter presents a comprehensive empirical study of performance influencing factors, advantages, and drawbacks of the general fuzzy min-max neural network on pattern classification problems. The subjects of this study include (1) the impact of maximum hyperbox size, (2) the influence of the similarity threshold and measures on the agglomerative learning algorithm, (3) the effect of data presentation order, (4) comparative performance evaluation of the GFMMNN with other types of fuzzy min-max neural networks and prevalent machine learning algorithms. The main content of this chapter is taken from the following paper (Khuat and Gabrys 2020):

- **Thanh Tung Khuat**, and Bogdan Gabrys, "A comparative study of general fuzzy min-max neural networks for pattern classification problems," *Neurocomputing*, vol. 386, pp. 110-125, 2020.

3.1 Introduction

Pattern classification, which belongs to the class of supervised learning, aims to discover information and knowledge under data through taking advantage of the power of learning algorithms (Olivas et al. 2009). It plays a crucial role in many real world applications ranging from medical diagnostic (Burger et al. 2017), electronic devices (Alibart et al. 2013) to tourism (Li et al. 2015) and energy (Jokar et al. 2016).

Multi-dimensional hyperbox fuzzy sets can be used to deal with the pattern classification problems effectively by partitioning the pattern space and assigning a class label associated with a degree of certainty for each region. Each fuzzy min-max hyperbox is represented by minimum and maximum points along with

a fuzzy membership function. The membership function is employed to compute the degree-of-fit of each input sample to a given hyperbox. Meanwhile, the hyperbox is continuously adjusted during the training process to cover the input patterns. Among the hyperbox-based learning algorithms, the GFMMNN is an effective learning model, which can accept both fuzzy and crisp patterns for the input data. This characteristic supports the GFMMNN to manage uncertainty in the input samples explicitly. Another significant modification of the GFMMNN compared to the original FMNN is the ability to process both classification and clustering in a single model. Therefore, the GFMMNN can be deployed to handle many types of real-world applications, especially problems with uncertain data and the input samples in the form of intervals.

Learning algorithms of the GFMMNN have a number of user-defined hyper-parameters, which can have a significant impact on their performance. Hence, a comparative study which illustrates the influence of hyper-parameters on the predictive accuracy is crucial for researchers to consider the applicability of the GFMMNN to practical problems. In addition, the study on the influence of factors on the performance of the GFMMNN opens the research directions towards optimizing the parameters and hyperparameters in an automatic manner. This comparative research includes assessments of the roles of configuration parameters on the predictive results of the classifiers, clarifying the efficiency and effectiveness as well as drawbacks of the GFMMNN in addressing the pattern classification problems, and reviewing the classification accuracy of the GFMMNN in comparison to other techniques using robust evaluation approaches. The main contributions in this chapter can be summarized as follows:

- A comparative study of the GFMMNN for pattern classification problems has been conducted, making clear the advantages and disadvantages of each training algorithm and identifying factors influencing the performance of the GFMMNN. The implementations of learning algorithms for the fuzzy min-max neural networks as well as benchmark datasets are publicly available at https://github.com/UTS-AAi/comparative-gfmm

- The GFMMNN has been empirically evaluated in comparison to other types of fuzzy min-max neural networks using the hyperbox expansion/contraction mechanism in the learning process as well as popular machine learning algorithms on the benchmark datasets using robust evaluation techniques, i.e., density-preserving sampling (DPS) (Budka and Gabrys 2013), parameter tuning by the grid-search method and cross-validation, as well as statistical hypothesis tests.

- Existing issues of the GFMMNN have been identified and discussed in depth.

3.2 General Fuzzy Min-Max Neural Network and Learning Algorithms

3.2.1 An Overall Architecture

General fuzzy min-max neural network contains three layers as shown in Figure 3.1. The input layer includes $2n$ nodes, where n is the number of dimensions of the input pattern. The first n nodes are the lower bounds of the input, while the remaining n nodes correspond to the upper bounds. These input nodes are connected to m hyperbox nodes in the hidden layer, in which the connection weights of the lower bound input nodes are stored in a matrix $\mathcal{V}$ and the connection weights between upper bound input nodes and hyperbox nodes are saved in a matrix $\mathcal{W}$. These weights correspond to minimum points and maximum points of hyperboxes, and their values are adjusted during the training process. Each hyperbox node B_i is associated with an activation function, which is also known as the membership function defined by Eq. (3.1).

$$b_i(X, V_i, W_i) = \min_{j=1}^{n}(\min([1 - f(x_j^u - w_j, \gamma_j)], [1 - f(v_j - x_j^l, \gamma_j)])) \tag{3.1}$$

where $f(z, \gamma) = \begin{cases} 1, & \text{if } z\gamma > 1 \\ z\gamma, & \text{if } 0 \leq z\gamma \leq 1 \\ 0, & \text{if } z\gamma < 0 \end{cases}$ is the ramp function, $\gamma = [\gamma_1, \ldots, \gamma_n]$ is a sensitivity parameter describing the speed of decreasing of the membership function,

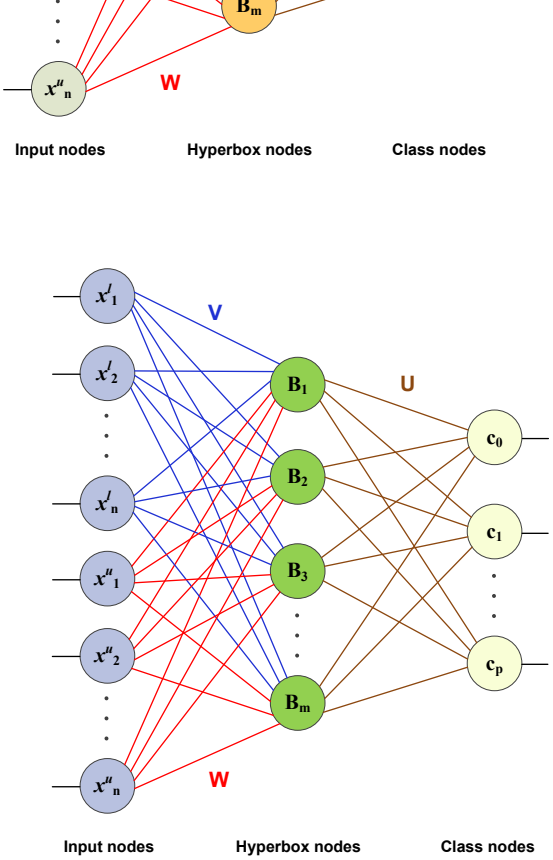


Figure 3.1 : An architecture of general fuzzy min-max neural network

and $X = [X^l, X^u]$ is an input pattern with lower bounds X^l and upper bounds X^u, which are vectors with values limited in the n-dimensional unit hyper-cube $[0, 1]^n$.

Each hyperbox B_i in the hidden layer is connected to each output (class) node c_j by a binary-valued parameter u_{ij} computed using Eq. (3.2). There are $p + 1$ output nodes corresponding to p classes, where node c_0 is linked to all unlabeled hyperboxes in the hidden layer. The transfer function of c_j is determined by the maximum membership value of all hyperboxes with same class as c_j and is shown in Eq. (3.3).

$$u_{ij} = \begin{cases} 1, & \text{if hyperbox } B_i \text{ represents class } c_j \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

$$c_j = \max_{i=1}^{m} b_i \cdot u_{ij} \tag{3.3}$$

where m is the number of hyperboxes in the middle layer. The output of each class node can be a fuzzy value calculated directly from Eq. (3.3) or a crisp value if the node associated with the highest membership value gets the value of one, and the others are assigned zero values (Gabrys and Bargiela 2000).

Although the GFMMNN can be applied for labeled and unlabeled datasets, this

chapter (also this thesis) focuses only on the classification problems. Therefore, the learning algorithms in the next sections are presented for labeled training data.

3.2.2 Online Learning Algorithm

The incremental (online) learning algorithm, proposed in Gabrys and Bargiela (2000), adjusts the size of existing hyperboxes or create new hyperboxes to accommodate new coming input patterns. There are three main steps in the algorithms including hyperbox expansion/creation, hyperbox overlap test, and hyperbox contraction. The pseudo code of the original online learning algorithm is given in Algorithm 3.1.

Assuming that each input pattern is represented in the form of $X = [X^l, X^u, c_X]$, where c_X is a class label and X^l and X^u are lower and upper bounds. The online learning algorithm first selects all existing hyperboxes with the same class as c_X. After that, the algorithm performs the computation of the membership values between these selected hyperboxes and the input pattern X, and then these membership values are sorted in a descending order (*lines 8-9*). Next, the algorithm traverses in turn each hyperbox B_i in the list of selected hyperboxes starting from the hyperbox with the maximum membership value to choose a hyperbox candidate aiming to expand and cover the input pattern. This process terminates when there is a hyperbox satisfying the expansion condition or the membership value is one (*lines 12-28*). Otherwise, a new hyperbox will be created with the same co-ordinates and label as the input pattern (*lines 29-31*). The expansion condition relates to the maximum hyperbox size threshold in each dimension as shown in Eq. (3.4).

$$\max(w_{ij}, x_j^u) - \min(v_{ij}, x_j^l) \leq \theta, \quad \forall j \in [1, n] \tag{3.4}$$

If this criterion is met, the hyperbox B_i is extended to accommodate the input pattern X using Eqs. (3.5) and (3.6).

$$v_{ij}^{new} = \min(v_{ij}^{old}, x_j^l) \tag{3.5}$$

$$w_{ij}^{new} = \max(w_{ij}^{old}, x_j^u), \quad \forall j \in [1, n] \tag{3.6}$$

Algorithm 3.1 The original online learning algorithm

Input:

- θ: The maximum hyperbox size threshold
- γ: The speed of decreasing of the membership function

Output:

A list $\mathcal{H}$ of hyperboxes with minimum-maximum values and classes

Initialize an empty list of hyperboxes: min-max values $\mathcal{V} = \mathcal{W} = \varnothing$, hyperbox classes: $\mathcal{L} = \varnothing$
for each input pattern $X = [X^l, X^u, c_X]$ **do**
  $n \leftarrow$ The number of dimensions of X
  if $\mathcal{V} = \varnothing$ **then**
    $\mathcal{V} \leftarrow X^l; \quad \mathcal{W} \leftarrow X^u; \quad \mathcal{L} \leftarrow c_X$
  else
    $\mathcal{H}_1 = [\mathcal{V}_1, \mathcal{W}_1, \mathcal{L}_1] \leftarrow$ Find hyperboxes in $\mathcal{H} = [\mathcal{V}, \mathcal{W}, \mathcal{L}]$ representing the same class as c_X
    $\mathcal{M} \leftarrow$ **ComputeMembershipValue**$(X, \mathcal{V}_1, \mathcal{W}_1, \mathcal{L}_1)$
    $\mathcal{H}_d \leftarrow$ **SortByDescending**$(\mathcal{H}_1, \mathcal{M}(\mathcal{H}_1))$
    Set $\overline{\mathcal{H}_1} \leftarrow \mathcal{H} \setminus \mathcal{H}_1$
    $flag \leftarrow false$
    for each $h = [V_h, W_h, c_h] \in \mathcal{H}_d$ **do**
      if $\mathcal{M}(h) = 1$ **then**
        $flag = true$
        break
      end if
      if $\max(w_{hj}, x_j^u) - \min(v_{hj}, x_j^l) \leq \theta, \forall j \in [1, n]$ **then**
        $W_h^t \leftarrow \max(W_h, X^u); \quad V_h^t \leftarrow \min(V_h, X^l)$
        for each hyperbox $[V_i, W_i, c_i] \in \overline{H_1}$ **do**
          $isOver \leftarrow$ **OverlapTest**(V_h^t, W_h^t, V_i, W_i)
          if $isOver = true$ **then**
            DoContraction(V_h^t, W_h^t, V_i, W_i)
          end if
        end for
        $flag = true$
        break
      end if
    end for
    if $flag = false$ **then**
      $\mathcal{V} \leftarrow \mathcal{V} \cup X^l; \quad \mathcal{W} \leftarrow \mathcal{W} \cup X^u; \quad \mathcal{L} \leftarrow L \cup c_X$
    end if
  end if
end for
return $\mathcal{H} = [\mathcal{V}, \mathcal{W}, \mathcal{L}]$

If the expansion step of the hyperbox candidate is carried out, the extended hyperbox B_i is verified for the overlap with the hyperboxes B_k representing other

classes. For each dimension j, four following conditions are checked for the overlap test operation (initially $\delta^{old} = 1$):

- $v_{ij} \leq v_{kj} < w_{ij} \leq w_{kj} : \delta^{new} = \min(w_{ij} - v_{kj}, \delta^{old})$
- $v_{kj} \leq v_{ij} < w_{kj} \leq w_{ij} : \delta^{new} = \min(w_{kj} - v_{ij}, \delta^{old})$
- $v_{ij} < v_{kj} \leq w_{kj} < w_{ij} : \delta^{new} = \min(\min(w_{kj} - v_{ij}, w_{ij} - v_{kj}), \delta^{old})$
- $v_{kj} < v_{ij} \leq w_{ij} < w_{kj} : \delta^{new} = \min(\min(w_{ij} - v_{kj}, w_{kj} - v_{ij}), \delta^{old})$

If $\delta^{new} < \delta^{old}$, then it is set $\Delta = j$ and $\delta^{old} = \delta^{new}$ to show an overlapping area on the Δth dimension, and the testing procedure is repeated for the next dimension. In contrast, there is no overlap region between two considered hyperboxes, and the hyperbox contraction step will not be performed ($\Delta = -1$). If $\Delta \neq -1$, the contraction procedure is applied on the Δth dimension to remove the overlapping area between two hyperboxes. The overlapping region is eliminated by tuning the value of the dimension with the smallest overlap. If $\Delta > 0$, this dimension is adjusted according to the four following cases:

Case 1: $v_{i\Delta} \leq v_{k\Delta} < w_{i\Delta} \leq w_{k\Delta} : w_{i\Delta}^{new} = v_{k\Delta}^{new} = (w_{i\Delta}^{old} + v_{k\Delta}^{old})/2$

Case 2: $v_{k\Delta} \leq v_{i\Delta} < w_{k\Delta} \leq w_{i\Delta} : w_{k\Delta}^{new} = v_{j\Delta}^{new} = (w_{k\Delta}^{old} + v_{j\Delta}^{old})/2$

Case 3: $v_{i\Delta} < v_{k\Delta} \leq w_{k\Delta} < w_{i\Delta}$:

$$v_{i\Delta}^{new} = w_{k\Delta}^{old}, \quad \text{if } w_{k\Delta} - v_{i\Delta} \leq w_{i\Delta} - v_{k\Delta}$$
$$w_{i\Delta}^{new} = v_{k\Delta}^{old}, \quad \text{if } w_{k\Delta} - v_{i\Delta} > w_{i\Delta} - v_{k\Delta}$$

Case 4: $v_{k\Delta} < v_{i\Delta} \leq w_{i\Delta} < w_{k\Delta}$:

$$w_{k\Delta}^{new} = v_{i\Delta}^{old}, \quad \text{if } w_{k\Delta} - v_{i\Delta} \leq w_{i\Delta} - v_{k\Delta}$$
$$v_{k\Delta}^{new} = w_{i\Delta}^{old}, \quad \text{if } w_{k\Delta} - v_{i\Delta} > w_{i\Delta} - v_{k\Delta}$$

In addition to setting up a fixed value of θ at the beginning of the learning algorithm and keeping it unchanged during the training process, another implementation using adaptive values θ was also introduced in Gabrys and Bargiela (2000).

In this way, the algorithm starts with a large value of θ, and then this value is decreased during the presentation of training data. The value of θ is updated after each iteration as Eq. (3.7):

$$\theta^{new} = \varphi \cdot \theta^{old} \tag{3.7}$$

where the coefficient φ ($0 \leq \varphi \leq 1$) controls the pace of decrease of θ. The learning process stops when no training patterns are misclassified or the minimum user-defined value of θ_{min} has been reached. This chapter will also compare the GFMMNN with the fixed and adaptive values of the parameter θ.

Time complexity of the Onln-GFMM algorithm. In terms of time complexity, assuming that there are N training samples with n features, the algorithm will first traverse each input sample and find a list of $\mathcal{K}$ hyperboxes with the same class as the input sample. The time complexity for this operation is constant if the hashtable technique is used. The membership computation must check all n dimensions of $\mathcal{K}$ hyperboxes, so the time complexity is $\mathcal{O}(\mathcal{K}n)$. $\mathcal{K}$ membership values for each input are obtained, therefore, the time complexity of the sorting operation is $\mathcal{O}(\mathcal{K}\log\mathcal{K})$. Let $\mathcal{R}$ be the number of hyperboxes representing classes different from the input class in the current iteration, it requires $\mathcal{O}(\mathcal{R})$ to collect these $\mathcal{R}$ hyperboxes. In the worst-case, all $\mathcal{K}$ selected hyperboxes are traversed to find the expandable hyperbox (*line 12*). For each hyperbox candidate, the checking of the expansion condition through n dimensions requires $\mathcal{O}(n)$. The overlap test between the hyperbox candidate and $\mathcal{R}$ hyperboxes belonging to other classes requires $\mathcal{O}(\mathcal{R}n)$ for time complexity. Only constant time is required for the process of contraction or generating a new hyperbox. Hence, the time complexity from *line 12* to *line 28* in the worst-case is $\mathcal{O}(\mathcal{K}\mathcal{R}n)$. Finally, let $\overline{\mathcal{K}}$ be the average number of hyperbox candidates for each iteration, $\overline{\mathcal{R}}$ be the average number of hyperboxes representing classes different from the input class over N training samples, the time complexity of learning algorithm 3.1 in the worst-case is $\mathcal{O}(N \cdot \overline{\mathcal{K}} \cdot \overline{\mathcal{R}} \cdot n)$.

3.2.3 Agglomerative Learning Algorithm

The online learning algorithm creates or adjusts the size of hyperboxes whenever an input sample comes in the network. Therefore, its performance depends on the data presentation order. Gabrys (2002a) proposed an agglomerative learning algorithm based on the full similarity matrix (AGGLO-SM) to reduce the impact of the data presentation order on the accuracy of the learning algorithm. In contrast to the online learning algorithm, the AGGLO-SM algorithm for the classification problems starts with all of the training samples. The idea is to merge hyperboxes with the same class, possessing the similarity values larger than a given threshold, and not generating the overlapping areas with existing hyperboxes representing other classes. The main steps of the AGGLO-SM algorithm are shown in Algorithm 3.2.

Firstly, the algorithm initializes a matrix $\mathcal{V}$ of minimum points and a matrix $\mathcal{W}$ of maximum points using the lower bounds $\mathbf{X}^l$ and upper bounds $\mathbf{X}^u$ of all training samples. Next, the algorithm performs a repeated training process of aggregating hyperboxes starting from the computation of a similarity matrix of hyperboxes for each class. There are three measures possible to be used to find the similarity value of each pair of hyperboxes B_i and B_k as follows:

- The first similarity measure is based on maximum points or minimum points of two hyperboxes. For simplifying, this measure is called as "middle distance" in this thesis, though the similarity measures are not distance measures:
  $s_{ik} = s(B_i, B_k) = \min_{j=1}^{n}(\min(1 - f(w_{kj} - w_{ij}, \gamma_j), 1 - f(v_{ij} - v_{kj}, \gamma_j)))$
  It can be seen that $s_{ik} \neq s_{ki}$, thus the similarity value between B_i and B_k may receive the minimum or maximum value between s_{ik} and s_{ki}. If the maximum value is used, this measure is called "mid-max distance"; otherwise, the name "mid-min distance" is used.

- The second similarity measure employs the smallest gap between two hyperboxes B_i and B_k, namely "shortest distance" in this thesis:

$$\widetilde{s}_{ik} = \widetilde{s}(B_i, B_k) = \min_{j=1}^{n}(\min(1 - f(v_{kj} - w_{ij}, \gamma_j), 1 - f(v_{ij} - w_{kj}, \gamma_j))) \quad (3.8)$$

Algorithm 3.2 Agglomerative algorithm with full similarity matrix - AGGLO-SM

Input:

- $\mathbf{X} = [\mathbf{X}^l, \mathbf{X}^u]$: A list of training features
- $\mathcal{C}$: A vector of pattern classes
- θ: The maximum hyperbox size threshold
- γ: The speed of decreasing of the membership function

Output:

A list $\mathcal{H}$ of hyperboxes with minimum-maximum values and classes

Initialize a list of hyperboxes: min-max values $\mathcal{V} = \mathbf{X}^l, \mathcal{W} = \mathbf{X}^u$, hyperbox classes: $\mathcal{L} = \mathcal{C}$
$loop \leftarrow true$; $n \leftarrow$ the number of features of $\mathbf{X}$
$S \leftarrow$ **ComputeSimilarityValPairWithinEachClass**$(\mathcal{V}, \mathcal{W}, \mathcal{L})$
while $loop = true$ **do**
    $loop \leftarrow false$
    $S \leftarrow S \setminus \{s \in S | s < \sigma\}$
    $I, K, S \leftarrow$ **SortByDescending**$(S, \mathcal{V}, \mathcal{W}, \mathcal{L})$
    for each $[i, k, s] \in [I, K, S]$ **do**
        if $\max(w_{ij}, w_{kj}) - \min(v_{ij}, v_{kj}) \leq \theta, \forall j \in [1, n]$ **then**
            $W_t \leftarrow \max(W_i, W_k)$; $V_t \leftarrow \min(V_i, V_k)$
            $\overline{H_1} \leftarrow$ A list of hyperboxes with classes different from $c_i \in \mathcal{L}$
            $isOver \leftarrow$ **IsOverlap**$(V_t, W_t, \overline{\mathcal{H}_1})$
            if $isOver = false$ **then**
                $loop \leftarrow true$
                $V_i \leftarrow V_t$; $W_i \leftarrow W_t$
                $\mathcal{V} \leftarrow \mathcal{V} \setminus V_k$; $\mathcal{W} \leftarrow \mathcal{W} \setminus W_k$; $\mathcal{L} \leftarrow \mathcal{L} \setminus L_k$
                $S \leftarrow$ **UpdateSimilarityMatrix**$(\mathcal{V}, \mathcal{W}, \mathcal{L})$
                break
            end if
        end if
    end for
end while
return $\mathcal{H} = [\mathcal{V}, \mathcal{W}, \mathcal{L}]$

- The third similarity measure is based on the longest possible distance between two hyperboxes B_i and B_k, called "longest distance" for short, defined as follows:
  $\widehat{s}_{ik} = \widehat{s}(B_i, B_k) = \min_{j=1}^{n}(\min(1 - f(w_{kj} - v_{ij}, \gamma_j), 1 - f(w_{ij} - v_{kj}, \gamma_j)))$
  it can be observed that both $\widetilde{s}_{ik}$ and $\widehat{s}_{ik}$ possess the symmetrical property.

From the similarity matrix of hyperboxes with the same class, the algorithm will merge sequentially hyperboxes by seeking for a pair of hyperboxes with the maximum

similarity value. It is noted that the algorithm only considers pairs of hyperboxes with similarity values larger than or equal to a minimum similarity threshold (σ): $s_{ih} \geq \sigma$ (*line 6* - Algorithm 3.2). Assuming that these two hyperboxes are B_i and B_k, the following conditions will be checked before aggregating:

(a) Maximum hyperbox size:
$\max(w_{ij}, w_{kj}) - \min(v_{ij}, v_{kj}) \leq \theta, \quad \forall j \in [1, n]$

(b) Overlap test. Newly aggregated hyperbox from B_i and B_k does not overlap with any existing hyperboxes belonging to other classes. The overlap checking conditions between two hyperboxes are shown in subsection 3.2.2. If any overlapping area exists, another pair of hyperboxes is selected.

If all above constraints are met, the hyperbox aggregation process is carried out as follows:

(a) Updating the coordinates of B_i so that B_i represents the coordinates of the merged hyperbox (*line 15*).

(b) Removing B_k from the current set of hyperboxes (*line 16*) and update the similarity matrix (*line 17*).

This training process is iterated until there are no pairs of hyperboxes to aggregate.

Time complexity of the AGGLO-SM algorithm. The time complexity of the computation of similarity values at *line 3* in Algorithm 3.2 is $\mathcal{O}(N^2 n)$, where N is the number of training samples and n is the number of features. With N training samples, a maximum of $N(N-1)/2$ hyperbox pairs is obtained. In the worst-case, all pairs of hyperboxes have to be looped through. At each iteration, let $\mathcal{Y}$ be the number of existing hyperboxes, the filtering step of hyperbox pairs satisfying the minimum similarity value (*line 6*) requires $\mathcal{O}(\mathcal{Y}^2)$ for time complexity. Assuming $\mathcal{Z}$ pairs of hyperbox candidates for the aggregation process are obtained, the complexity of the sorting step is $\mathcal{O}(\mathcal{Z} \log \mathcal{Z})$. In the worst-case, all of these

$\mathcal{Z}$ candidate pairs need to be checked for the aggregation process. For each pair, the checking for the maximum hyperbox size condition requires $\mathcal{O}(n)$. The process of collecting hyperboxes representing classes different from the newly aggregated hyperbox takes $\mathcal{O}(\mathcal{Y})$. The overlap test between the newly aggregated hyperbox and existing hyperboxes requires $\mathcal{O}(\mathcal{Y}n)$. The update step of the similarity matrix takes $\mathcal{O}(\mathcal{Y}n)$. Therefore, in the worst-case, the process of aggregating a pair of hyperboxes (*lines 6-21* in Algorithm 3.2) requires $\mathcal{O}(\mathcal{Z}\mathcal{Y}n+\mathcal{Y}^2)$ for time complexity. As a result, let $\overline{\mathcal{Z}}$ be the average number of pairs of hyperbox candidates considered during the training process and $\overline{\mathcal{Y}}$ be the number of existing hyperboxes in each iteration, the time complexity for the AGGLO-SM algorithm is $\mathcal{O}(N^2 \cdot (\overline{\mathcal{Z}} \cdot \overline{\mathcal{Y}} \cdot n + \overline{\mathcal{Y}}^2))$ in the worst-case.

3.2.4 Accelerated Agglomerative Learning Algorithm

Training process of the AGGLO-SM algorithm takes a very long time to complete, especially for massive datasets, due to the fact that the similarity matrix for all pairs of hyperboxes needs to be computed and sorted. To lower the training time of the AGGLO-SM algorithm, Gabrys (2002a) also introduced the second agglomerative algorithm (AGGLO-2) removing the usage of the full similarity matrix when selecting and merging hyperboxes. The main steps of the AGGLO-2 are shown in Algorithm 3.3.

The AGGLO-2 algorithm traverses and selects in turn each hyperbox in the current list of hyperboxes to perform the hyperbox merging process. For the first selected hyperbox candidate B_i, the similarity values of B_i and the remaining hyperboxes with the same class as B_i are computed and sorted (*lines 7-9* in Algorithm 3.3). The hyperbox B_k with the highest similarity value is chosen as the second candidate for the aggregation. The aggregation constraints of B_i and B_k are the same as in the AGGLO-SM algorithm. If the aggregation conditions are met, the coordinates of B_i are updated so that it shows the aggregated hyperbox and remove B_k of the list of current hyperboxes. If B_i and B_k do not meet the constraints, the hyperbox with the second largest similarity value is selected, and the above checking and merging steps are repeated until the agglomeration happens. The learning algo-

rithm stops when no pair of hyperboxes can be aggregated (the variable $loop = false$ in Algorithm 3.3).

Algorithm 3.3 The agglomerative algorithm version two - AGGLO-2

Input:

- $\mathbf{X} = [\mathbf{X}^l, \mathbf{X}^u]$: A list of training features
- $\mathcal{C}$: A vector of pattern classes
- θ: The maximum hyperbox size threshold
- γ: The speed of decreasing of the membership function

Output:

A list $\mathcal{H}$ of hyperboxes with minimum-maximum values and classes

Initialize a list of hyperboxes: min-max values $\mathcal{V} = \mathbf{X}^l, \mathcal{W} = \mathbf{X}^u$, hyperbox classes: $\mathcal{L} = \mathcal{C}$
$loop \leftarrow true$; $n \leftarrow$ the number of features of $\mathbf{X}$
while $loop = true$ **do**
    $loop \leftarrow false$; $i \leftarrow 1$
    while $i \leq |\mathcal{L}|$ **do**
        $\mathcal{H}_1 = [\mathcal{V}_1, \mathcal{W}_1, \mathcal{L}_1] \leftarrow$ Find hyperboxes in $[\mathcal{V}, \mathcal{W}, \mathcal{L}]$ representing the same class as $c_i \in \mathcal{L}$
        $S \leftarrow$ **ComputeSimilarityValPair**$(V_i, W_i, \mathcal{H}_1)$
        $S \leftarrow S \setminus \{s \in S | s < \sigma\}$
        $K, S \leftarrow$ **SortByDescending**$(S, \mathcal{V}_1, \mathcal{W}_1, \mathcal{L}_1)$
        for each $[k, s] \in [K, S]$ **do**
            if $\max(w_{ij}, w_{kj}) - \min(v_{ij}, v_{kj}) \leq \theta, \forall j \in [1, n]$ **then**
                $W_t \leftarrow \max(W_i, W_k)$; $V_t \leftarrow \min(V_i, V_k)$
                $\overline{H_1} \leftarrow$ A list of hyperboxes with classes different from l_i
                $isOver \leftarrow$ **IsOverlap**$(V_t, W_t, \overline{\mathcal{H}_1})$
                if $isOver = false$ **then**
                    $loop \leftarrow true$
                    $V_i \leftarrow V_t$; $W_i \leftarrow W_t$
                    $\mathcal{V} \leftarrow \mathcal{V} \setminus V_k$; $\mathcal{W} \leftarrow \mathcal{W} \setminus W_k$; $\mathcal{L} \leftarrow \mathcal{L} \setminus L_k$
                    if $i > k$ **then**
                        $i \leftarrow i - 1$
                    end if
                    break
                end if
            end if
        end for
        $i \leftarrow i + 1$
    end while
end while
return $\mathcal{H} = [\mathcal{V}, \mathcal{W}, \mathcal{L}]$

Time complexity of the AGGLO-2 algorithm. Assuming that it is given N training samples with n features, in the worst-case, the training process has to loop through all N initial hyperboxes. In each iteration, the process of finding the hyperboxes representing the same class as the considered hyperbox takes constant time if using the hashtable. Let $\mathcal{K}$ be the number of hyperboxes with the same class as the considered hyperbox in the current iteration, the computation step of similarity values takes $\mathcal{O}(\mathcal{K}n)$. The filtering step of hyperbox candidates satisfying the minimum similarity value takes $\mathcal{O}(\mathcal{K})$. Let $\mathcal{Z}$ be the number of pairs of hyperbox candidate for the aggregation process, the sorting step requires $\mathcal{O}(\mathcal{Z}\log\mathcal{Z})$. In the worst-case, the aggregation process needs to loop through all $\mathcal{Z}$ pairs of hyperbox candidates. For each candidate pair, the checking step of the maximum hyperbox size condition requires $\mathcal{O}(n)$. Let $\mathcal{R}$ be the number of hyperboxes representing classes different from the considered pair of hyperbox candidate, it takes $\mathcal{O}(\mathcal{R})$ to find these hyperboxes. The overlap test step (*line 14* in Algorithm 3.3) requires $\mathcal{O}(\mathcal{R}n)$. The operation of removing a merged hyperbox requires $\mathcal{O}(\mathcal{K})$ but this step only occurs once among $\mathcal{Z}$ pairs of hyperbox candidates. As a result, the time complexity for the aggregation process (*lines 6-26*) is $\mathcal{O}(\mathcal{Z}\mathcal{R}n)$. In summary, let $\overline{\mathcal{Z}}$ be the average number of pairs of hyperbox candidates and $\overline{\mathcal{R}}$ be the average number of hyperboxes belonging to classes different from the class of the considered pair of hyperbox, the time complexity of the AGGLO-2 algorithm is $\mathcal{O}(N \cdot \overline{\mathcal{Z}} \cdot \overline{\mathcal{R}} \cdot n)$ in the worst-case.

3.3 Existing Problems and Motivations

Fuzzy min-max neural networks are universal approximators, which can tackle both linear and non-linear classification problems. However, these classifiers depend on the selection of hyper-parameters, such as the maximum hyperbox size. If the hyper-parameters are set well, the trained model will achieve good performance on unseen data. Nonetheless, this is a challenging task because of the huge searching space of parameters. This study is not to optimize the hyper-parameters in an automatic manner. Instead, the impact of hyper-parameters on the performance of the models for each dataset is assessed. Based on these evaluations, conclusions related to the important role of the selection of hyper-parameters with regard to predic-

tive accuracy of models on each training dataset can be drawn. As a result, when comparing various learning algorithms, the best settings in the range of potential parameters were chosen based on the performance of classifiers on validation sets, which are formed by K-fold cross-validation and the density-preserving sampling method.

To generate a hyperbox-based classifier with good generalization error, besides independent learning schemes such as cross-validation and resampling approaches (Gabrys 2004), it also needs to integrate the explicit overfitting prevention mechanisms, i.e., pruning procedures, to learning algorithms. Taking decision trees as an example, if the training process constructs a full tree structure, the model will overfit the training set. Therefore, to ensure a good generalization error, one usually applies early stopping and pruning methods. Similarly, if the maximum hyperbox size is set to a small value, there are many generated hyperboxes for each hyperbox-based learner. These hyperbox fuzzy sets are more likely to overfit the training data. An example is shown in Figure 3.2 for *Iris* dataset with 112 training samples and two out of its four features. The model is trained using a small value of maximum hyperbox size ($\theta = 0.06$). It can be seen that the model contains 79 hyperboxes, and many hyperboxes include only one sample, which is unnecessarily complex.

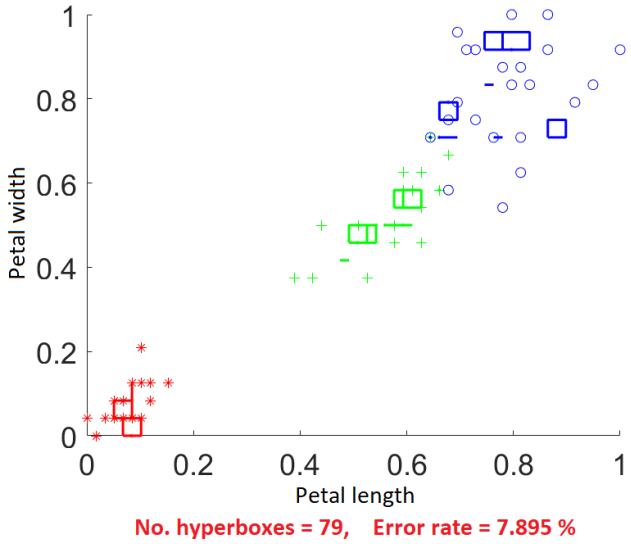


Figure 3.2 : A hyperbox-based model is trained on the *Iris* dataset

To cope with this problem, the training dataset can be split into disjoint training and validation sets using the DPS method (75 training samples and 37 validation patterns). The model trained on the training set is shown in Figure 3.3. The number of generated hyperboxes is lower than in the previous case because a smaller number of training samples were used, but the accuracy is still the same. This result also confirms that the DPS method can generate a representative training set from the original data. After training, the validation set is employed to remove low-quality hyperboxes, which have predictive accuracy less than 50%. The final classifier is presented in Figure 3.4. It can be easily observed that both the number of generated hyperboxes and error rate have been significantly reduced.

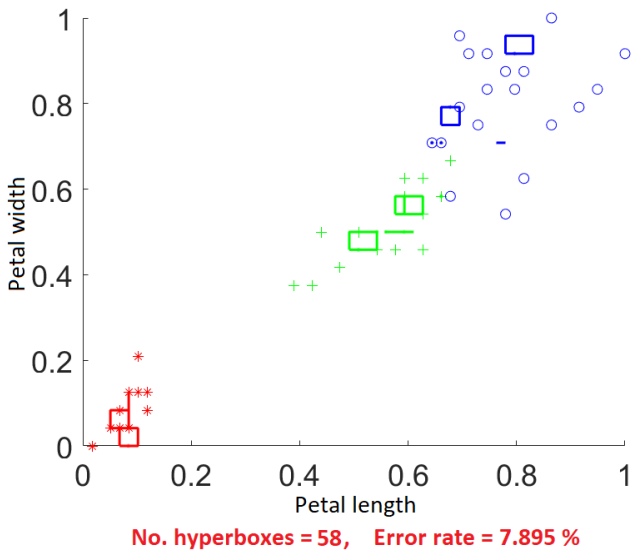


Figure 3.3 : A hyperbox-based model is trained on the *Iris* dataset

The removal of hyperboxes can lead to loss of important information because this operation is based on only the misclassification error on the validation set. If the selection of hyper-parameters results in a nearly optimal decision boundary after the training process, the pruning procedure may increase the error rates since it will break the optimal structure of the trained model. The experiments in the next section focus on clarifying the role of the pruning process if the classifier has been built using the best hyper-parameters. It can be also desirable to find the answer to the question of whether the impact of noisy data can be reduced through parameter

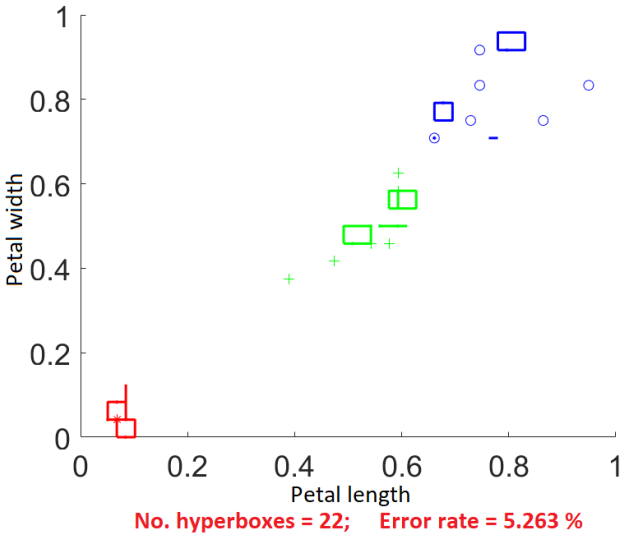


Figure 3.4 : A hyperbox-based model is trained on the *Iris* dataset

settings rather than identifying and removing them through pruning or data editing (Gabrys 2001).

3.4 Experiments and Results

The experiments in this chapter used 16 relatively small-sized datasets from the UCI repository (Dua and Graff 2019), which are shown in Table A.1 in Appendix A. Each dataset was separated into four folds using the density-persevering sampling technique (Budka and Gabrys 2013), which is a robust and efficient method competitive to cross-validation for error estimation. Three folds were used as training data, while the remaining fold was selected as a testing set. In common, for each dataset, experiments were repeated four times with each fold used as testing data in turn and reported results were average of results on each testing fold.

3.4.1 The Influence of the Maximum Hyperbox Size on the Performance of the Online Learning based GFMMNN

This experiment is to assess the impact of the maximum hyperbox size parameter, θ, on the performance of the GFMMNN using the incremental learning algorithm. Three out of four folds were used for training the network and one

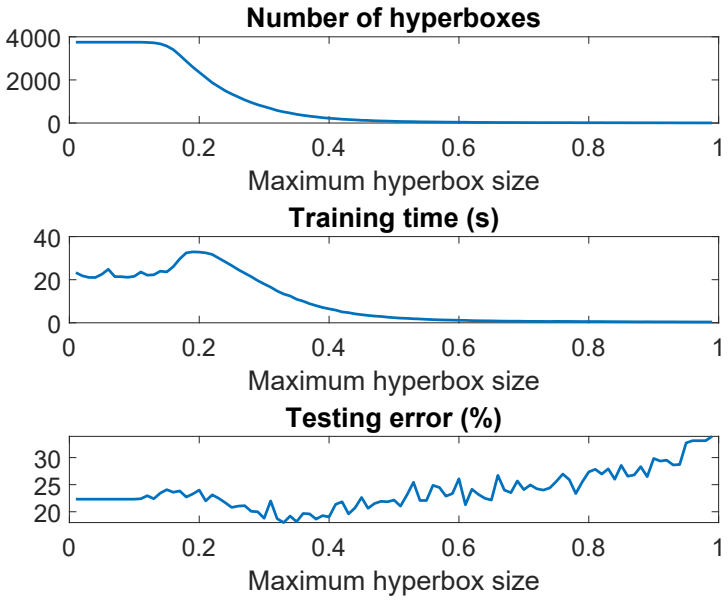


Figure 3.5 : The change in the number of hyperboxes, training time, and testing error of the *Waveform* dataset

remaining fold for the testing process. The value of θ was increased from 0.01 to 0.99 with the step being 0.01 and the incremental learning was used with the fixed hyperbox size for each dataset. Entire figures showing the change in the number of hyperboxes, training time, and testing error of all considered datasets can be found at https://github.com/UTS-AAi/comparative-gfmm/blob/master/experiment/hyperbox-size-changing.pdf. A representative example of changing trend in the number of generated hyperboxes, training time, and testing error is presented in Figure 3.5 for the *Waveform* dataset.

It can be seen that the larger value of θ, the fewer the number of hyperboxes in the model is generated. Generally, the training time also reduces when increasing the value of θ, and the training time is usually fast and decreases in a stable manner if the maximum hyperbox size is larger than 0.5. Furthermore, training time frequently fluctuates and stands at a high value when the value of θ is less than 0.2. Regarding the testing error, there is no general rule for all datasets when the value of θ gets larger, but the error rates are frequently high if the θ thresholds are larger than 0.8, except *Zelnik6*, *Thyroid*, *Iris*, and *Wine* datasets. It is easily observed from the

images that the prediction results of the GFMMNN using an incremental learning algorithm are substantially influenced by the selection of values of θ. It is not straightforward to choose an optimal value of θ to gain the best performance for each dataset. Several optimization algorithms can be deployed to find the optimal value of θ in an automatic manner.

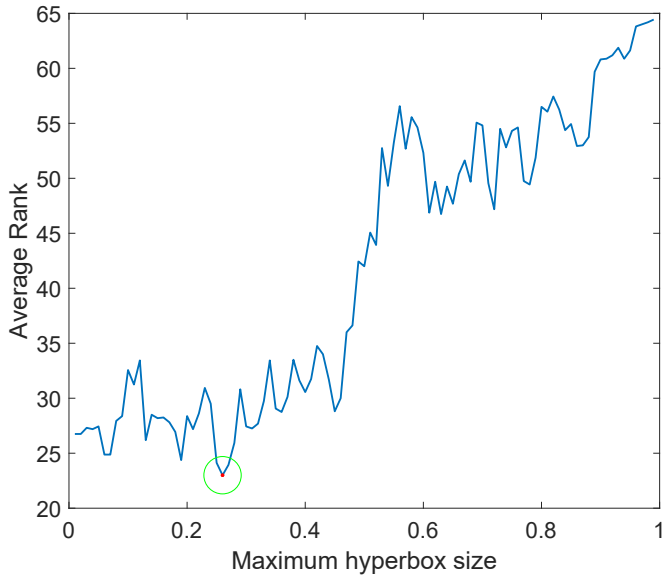


Figure 3.6 : Average rank of performance on 16 datasets using different values of θ

To remedy the impact of the maximum hyperbox size, the incremental learning algorithm using the adaptive value of θ was developed as described at the end of subsection 3.2.2. To compare the performance of GFMMNN using the adaptive values of θ with the one using the fixed value of θ, $\theta = 0.26$ as selected as an initial value, and the learning algorithm was repeated until the minimum value of θ being 0.01 was reached out ($\varphi = 0.9$) in the case of using the adaptive incremental learning algorithm. The value $\theta = 0.26$ was selected because it gave the lowest average rank of prediction errors over 16 datasets in comparison to other fixed values of θ as shown in the above experiment. The average rank of the performance of general fuzzy min-max neural network using different fixed values of θ over 16 datasets is given in Figure 3.6.

In this experiment, each dataset was also split into four folds, and each execu-

tion used a fold for testing and three remaining folds were deployed for training. For each training dataset, ten runs were performed, and each iteration shuffled training data randomly. The obtained value for each testing fold is an average of ten executions. Table 3.1 reports the averaged experimental results concerning the number of generated hyperboxes, training time, and testing error rate for two strategies of employing the value of θ on four folds over different datasets.

Table 3.1 : Comparison of fixed and adaptive maximum hyperbox size parameters ($\theta = 0.26$)

ID	**Dataset**	**Fixed value**			**Adaptive value** ($\theta_{min} = 0.01$)		
		No. hyperboxes	**Training time (s)**	**Testing error (%)**	**No. hyperboxes**	**Training time (s)**	**Testing error (%)**
1	Circle	29.950	0.092	5.240	71.175	3.092	3.530
2	Complex 9	28.275	0.272	1.755	38.350	10.913	0.267
3	Diagnostic Breast Cancer	113.550	0.302	4.586	118.400	0.740	4.516
4	Glass	42.675	0.060	39.286	75.425	1.220	40.597
5	ionosphere	144.675	0.178	12.229	144.675	0.230	12.229
6	Iris	16.775	0.016	4.683	18.975	0.393	4.491
7	Ringnorm	1411.525	31.666	26.468	2260.450	164.892	27.886
8	Segmentation	230.275	2.970	4.588	246.750	25.998	4.567
9	Spherical_5_2	13.600	0.020	1.274	13.600	0.040	1.274
10	Spiral	26.95	0.102	7.810	42.450	2.902	0.650
11	Thyroid	22.475	0.025	4.268	30.400	0.576	3.988
12	Twonorm	1862.950	44.715	4.932	1926.500	57.923	4.928
13	Waveform	1185.700	24.529	20.688	1622.375	55.546	20.638
14	Wine	75.375	0.056	4.229	75.375	0.074	4.229
15	Yeast	128.900	0.992	67.832	1456.750	137.667	72.062
16	Zelnik6	12.600	0.015	0.212	12.600	0.031	0.212

In several datasets such as *Circle*, *Complex 9*, and *Spiral*, the testing errors fell sharply when using the adaptive mechanism for θ. Meanwhile, the error rate in some datasets like *Glass*, *Ringnorm*, and *Yeast* increased slightly in the case of implementing adaptive values of θ. A reason for this fact is the overfitting in the trained model. It can be seen this phenomenon in Figure 3.7 for the *Ringnorm* dataset, where a large number of hyperboxes were generated and the testing errors at fixed values of $\theta < 0.26$ are relatively high. In the remaining cases, the error rates

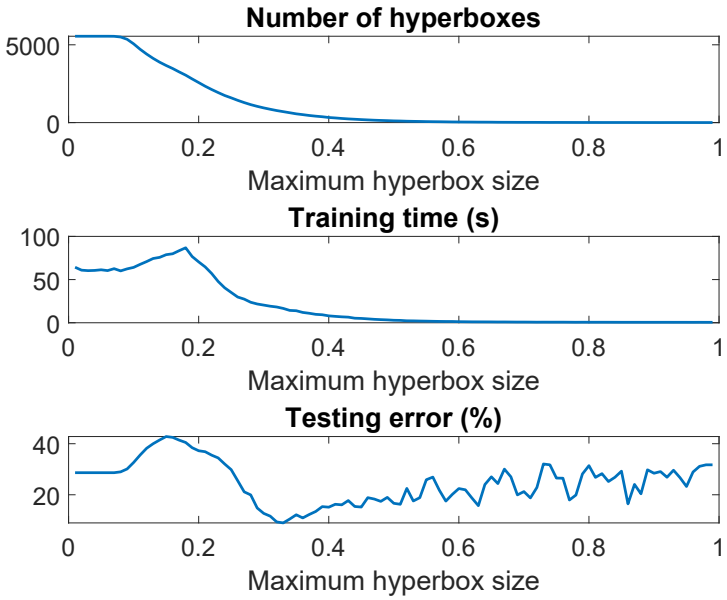


Figure 3.7 : The change in the number of hyperboxes, training time, and testing error of the *Ringnorm* dataset

of the GFMMNN using adaptive values of θ are slightly lower or the same as those employing the fixed values of the maximum hyperbox size. It can be concluded that the adaptive hyperbox size based GFMMNN has limited impact in case of using the starting value of θ being the best value for many datasets. To further evaluate the performance of the GFMMNN using the adaptive values of hyperbox size, another starting value of θ away far from the optimal value was chosen. $\theta = 0.56$ was selected because it leads to the large changing in the average rank of GFMMNN as shown in Figure 3.6. The outcomes of GFMMNN using fixed value of $\theta = 0.56$ and adaptive values starting from $\theta = 0.56$ are shown in Table 3.2.

It is easily observed that in most of the datasets the testing errors using adaptive values of θ are significantly enhanced compared to the cases using the fixed values of θ. In several datasets such as *Yeast*, *Thyroid*, *Segmentation*, and *Ionosphere*, the accuracy of predictive results decreases slightly. In general, the accuracy of GFMMNN using adaptive values of θ starting from $\theta = 0.56$ is superior to that employing the fixed value $\theta = 0.56$. However, the number of created hyperboxes and training time of the algorithm using the adaptive values of θ increased considerably,

Table 3.2 : Comparison of fixed and adaptive maximum hyperbox size parameters ($\theta = 0.56$)

ID	Dataset	Fixed value			Adaptive value ($\theta_{min} = 0.01$)		
		No. hyperboxes	**Training time (s)**	**Testing error (%)**	**No. hyperboxes**	**Training time (s)**	**Testing error (%)**
1	Circle	9.65	0.059	15.22	73.275	3.937	3.48
2	Complex 9	11.775	0.234	11.943	37.65	13.256	0.432
3	Diagnostic Breast Cancer	22.35	0.065	5.733	84.85	1.571	4.994
4	Glass	17.225	0.04	47.983	105.375	1.793	46.062
5	ionosphere	80.825	0.112	13.62	81.625	2.38	13.733
6	Iris	6.875	0.008	6.01	13.125	0.587	3.558
7	Ringnorm	59.25	1.74	21.77	2151.5	593.817	4.768
8	Segmentation	47.725	0.486	17.349	442.9	34.322	17.882
9	Spherical_5_2	5	0.014	0.794	5	0.032	0.794
10	Spiral	8.975	0.084	41.94	52.225	4.068	1.38
11	Thyroid	8.05	0.015	5.196	30.875	0.84	5.206
12	Twonorm	51.55	1.874	13.205	3539.95	561.18	5.27
13	Waveform	47.95	1.508	23.054	3265.75	858.192	19.416
14	Wine	17.7	0.025	3.586	17.775	0.037	3.586
15	Yeast	34.775	0.704	92.507	1933.275	626.437	93.713
16	Zelnik6	7	0.012	6.895	8.475	0.394	1.013

especially in large-sized datasets such as *Ringnorm*, *Twonorm*, *Waveform*, and *Yeast* datasets. In addition, the accuracy of GFMMNN in this experiment is lower than that using adaptive values of the maximum hyperbox size starting from $\theta = 0.26$. In many datasets, it can be seen that the error rates of GFMMNN using the adaptive values from $\theta = 0.56$ are higher than those utilizing fixed value $\theta = 0.26$. These results indicate the impacts of choosing the suitable values of maximum hyperbox size on the accuracy of predictive results. They also confirm that the incremental learning algorithm using the adaptive values of the maximum hyperbox size has not yet been an effective method to tackle the dependence of classification performance on the selection of the maximum hyperbox size parameter. Hence, to compare the performance of GFMMNN with other methods, the fixed value of θ that leads to the minimum error on the validation set in the range of given values for each dataset was used rather than using the same value of θ for all considered datasets.

3.4.2 The Influence of the Similarity Threshold on the Performance of the Agglomerative Learning based GFMMNN using Different Similarity Measures

This experiment is to evaluate the influence of the similarity threshold on the performance of AGGLO-2 and AGGLO-SM algorithms using different similarity measures. For each dataset, a fold was selected for testing data, while three other folds were used as training data. The maximum hyperbox size $\theta = 0.26$ was used in this experiment. The minimum similarity threshold values (σ) were moved from 0.02 to 0.98 with the step being 0.02. The graphs showing the change in the number of hyperboxes and the testing error through several typical datasets can be found at https://github.com/UTS-AAi/comparative-gfmm/blob/master/experiment/similarity-threshold-changing.pdf. An example is presented in Figure 3.8.

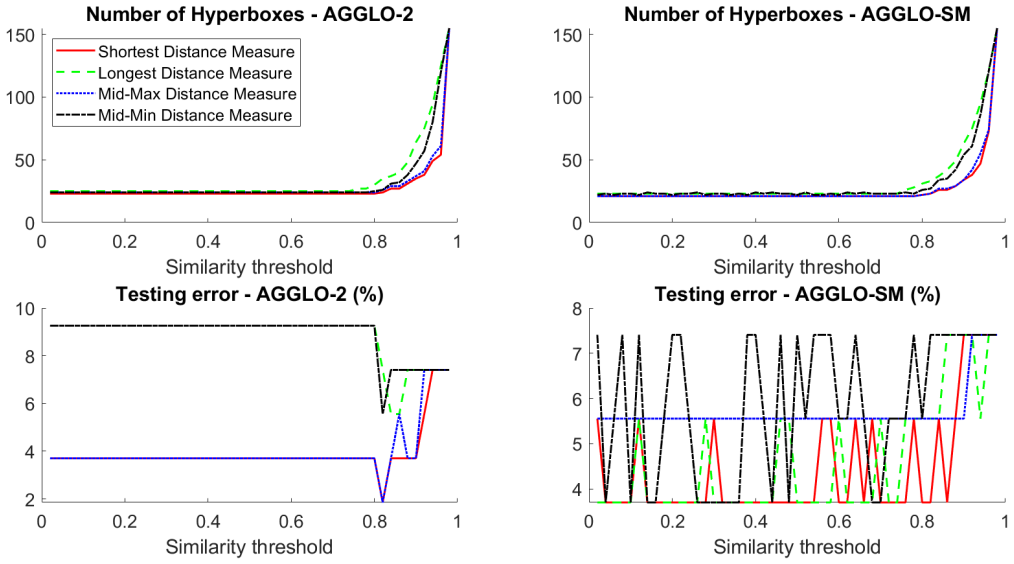


Figure 3.8 : The influence of similarity threshold on the number of hyperboxes and testing errors for GFMMNN using agglomerative learning on the *Thyroid* dataset

It can be seen from the figures that the numbers of hyperboxes of both algorithms on all similarity measures regularly increase when the similarity threshold moves to one. Especially, they sharply rise when the threshold is larger than 0.8, and they

oscillate a little if the similarity value is less than 0.7. It can be seen that the number of generated hyperboxes in the case of using the shortest distance measure to compute the similarity degree is lowest, whereas the use of the longest distance measure results in the highest number of generated hyperboxes among four measures.

For the AGGLO-SM algorithm, the selection of the similarity threshold considerably affects the testing error. Its testing error rates oscillate not following a general rule. For the AGGLO-2, the testing error fluctuates only if the value of the similarity threshold is larger than 0.8. Therefore, experiments in the rest of this chapter employed a similarity threshold $\sigma \leq 0.8$ for the agglomerative learning algorithms. It can be observed that the best performance of the AGGLO-2 algorithm is frequently achieved in the case of using the shortest distance measure. It can recognized that the classification performance of the GFMMNN using the agglomerative learning algorithms depends on the choice of the similarity measures for each dataset. Of four similarity measures, there is no measure giving the best results on all datasets. Hence, the similarity measure, similarity threshold, and maximum hyperbox size are three hyper-parameters that need to be optimized for each dataset to achieve the best predictive accuracy.

3.4.3 Comparison of Different Agglomerative Learning Versions of GFMMNN

This part compares the full similarity matrix based agglomerative learning and accelerated agglomerative learning algorithms. Each dataset was split into four folds using the density-preserving sampling method (Budka and Gabrys 2013). Each fold was used in turn as testing data, while the remaining folds were employed as the training set. The obtained result of each model is the average result of four testing folds. For a given training set, experiments were repeated ten times to determine the average training time. The similarity threshold $\sigma = 0.8$ and the maximum hyperbox size $\theta = 0.26$ were established for both algorithms on all datasets. Table 3.3 shows the mean values of the number of produced hyperboxes, training time, and testing error rate of each algorithm through typical datasets.

Table 3.3 : The comparison of the full similarity matrix based agglomerative learning and accelerated agglomerative learning, $\theta = 0.26, \sigma = 0.8$

ID	Dataset	AGGLO-2			AGGLO-SM		
		No. hyperboxes	**Training time (s)**	**Testing Error (%)**	**No. hyperboxes**	**Training time (s)**	**Testing error (%)**
1	Circle	40.750	0.196	3.200	41	21.998	3.300
2	Complex 9	31.750	0.932	0.165	30.500	229.952	0.231
3	Diagnostic Breast Cancer	133.500	0.579	5.6252	133.250	20.515	5.622
4	Glass	47.500	0.043	35.500	47.750	0.564	41.125
5	ionosphere	151.750	0.179	11.406	152.250	3.164	11.974
6	Iris	18.250	0.023	4.623	17.500	0.173	4.623
7	Segmentation	243.750	2.237	4.285	240.750	171.512	3.982
8	Spherical_5_2	13.750	0.029	1.197	12.750	0.639	0.397
9	Spiral	28.500	0.169	0.100	24.500	12.132	0
10	Thyroid	26	0.037	5.573	24.500	0.599	4.167
11	Wine	89.250	0.061	5.076	91	0.391	5.076
12	Yeast	144.250	1.295	68.661	139.750	97.463	70.348
13	Zelnik6	12.750	0.031	0.424	12.500	0.681	0

As indicated in the table, the AGGLO-2 algorithm is from one to two orders of magnitude faster than the AGGLO-SM in almost all datasets. However, the average number of hyperboxes generated in the AGGLO-2 is slightly higher than that of hyperboxes created by the AGGLO-SM algorithm. The average testing error values of the GFMMNN using the AGGLO-2 are slightly higher than those using the AGGLO-SM algorithm on many datasets except *Circle*, *Complex 9*, *Glass*, *Ionosphere*, and *Yeast*. In general, the predictive results using the GFMMNN trained by the AGGLO-2 is relatively the same as those implementing the AGGLO-SM while the training time is much faster. As a result, the AGGLO-2 algorithm significantly improves the performance of the full similarity matrix based agglomerative learning algorithm. It is noted that the training time of the AGGLO-SM algorithm for large-sized training datasets such as *Ringnorm*, *Twonorm*, and *Waveform* is extremely long (more than two days for each iteration), so they were not reported in this chapter. The computational expense of the AGGLO-SM is costly because of its cubic time complexity. This fact prevents the applicability of the AGGLO-SM in tackling

Table 3.4 : Standard deviation on results of diffirent versions of GFMMNNs due to the impact of presentation order

ID	Dataset	Online		AGGLO-2		AGGLO-SM	
		No. hyperboxes	**Testing error (%)**	**No. hyperboxes**	**Testing error (%)**	**No hyperboxes**	**Testing error (%)**
1	Circle	1.687	0.844	1.059	0.627	0	0
2	Complex 9	1.287	0.377	0.994	0.056	0	0
3	Diagnostic Breast Cancer	3.011	1.188	2.944	0.996	1.054	0.371
4	Glass	1.059	5.031	0.667	3.334	0	0
5	ionosphere	1.430	1.290	1.265	0.804	0	0
6	Iris	0.632	1.140	1.370	1.396	0.949	0
7	Segmentation	5.446	0.364	3.736	0.390	0.516	0
8	Spherical_5_2	1.174	0.502	0.707	0.837	0	0
9	Spiral	1.764	2.892	1.491	0	0	0
10	Thyroid	1.197	1.991	0.816	1.295	0	0.895
11	Wine	1.829	1.174	1.633	0	0	0
12	Yeast	2.058	2.107	2.406	1.222	1.337	0.475
13	Zelnik6	0.667	0	0.422	0	0	0

large-sized datasets. In the rest of this chapter, the AGGLO-2 was implemented for the next experiments to compare to other classification algorithms.

3.4.4 The Influence of Data Presentation Order on the Performance of GFMMNN

This experiment is to assess the impact of data presentation order to the classification performance of incremental learning and agglomerative learning algorithms of the GFMMNN. For each dataset, one fold was chosen as the testing set, and three remaining folds were training data. Each experiment was executed ten times, and each time randomly shuffled the order of samples in the same training set, and three learning algorithms were trained on the same dataset. The similarity threshold $\sigma = 0.8$ using the shortest distance measure was set for the similarity computation and the maximum hyperbox size parameter $\theta = 0.26$. Table 3.4 reports the standard deviation (std) of the number of hyperboxes and testing errors of different algorithms applied to 13 datasets.

It is seen that the standard deviation values of the testing errors of the GFMMNN trained by the AGGLO-SM algorithm are zero on almost all datasets, except *Yeast*, *Diagnostic Breast Cancer*, and *Thyroid*. Even on these three datasets, the standard deviation of testing error values is very small ($< 0.9\%$). These figures indicate that the full similarity matrix based learning algorithm is almost unaffected by the input data presentation order. In contrast, of three learning algorithms, the incremental learning version is most affected by the data presentation order since hyperboxes are adjusted for each input pattern. The AGGLO-2 is also influenced by the data presentation order because it selects, in turn, each hyperbox to calculate the similarity value with the other ones, but the standard deviation values of testing errors are quite tiny. This experiment confirms that agglomerative learning algorithms are stable against the change of presentation order within training data.

3.4.5 Comparison of GFMMNN and Other Types of Fuzzy Min-Max Neural Networks

This experiment aims to compare the performance of the GFMMNNs to other types of fuzzy min-max neural networks using the expansion and contraction phases in the learning algorithm such as the original fuzzy min-max neural network (Simpson 1992), the enhanced fuzzy min-max neural network (Mohammed and Lim 2015), and the enhanced fuzzy min-max neural network with the K-nearest hyperbox selection rule (Mohammed and Lim 2017a).

Through experimental results mentioned above, it can be observed that the performance of fuzzy min-max neural networks depends on the value of maximum hyperbox size for each dataset. Therefore, the grid search method and 3-fold cross-validation were used for tuning the maximum hyperbox size of the classification model on validation sets among values within the list of $\theta \in \{0.06, 0.1, 0.16, 0.2, 0.26, \ldots, 0.8\}$. In addition to the maximum hyperbox size, the KNEFMNN model also depends on the number of selected hyperboxes (K) for the hyperbox expansion process. The searching range of K in the range of $[2, 10]$ was set. As for the AGGLO-2 version, the longest-distance measure was used and the similarity threshold $\sigma = 0$ was set so that the GFMM model using this agglomerative learning algorithm is only

dependent on the value of θ. It is not feasible to exhaustively explore all the possible values for the maximum hyperbox size value, and the purpose of this chapter is to compare the performance of the fuzzy min-max classifiers, not on the fine-tuning approaches, the number of values was limited for each parameter.

Each dataset was split into four folds using the density-preserving sampling method (Budka and Gabrys 2013). Each fold was selected as testing set in turn, while three remaining folds were employed as the training and validation data. Assuming that F_1, F_2, and F_3 are three folds used for parameter-tuned process, F_1 and F_2 are employed as training data to construct the fuzzy min-max classifiers for each value of θ. Then, the error rate on the validation fold F_3 is computed. This process is repeated for F_1 and F_2 used as the validation set. The value of θ leading to the lowest averaged prediction error on three folds is selected to build the final fuzzy min-max classifier on the training set containing all F_1, F_2 and F_3 folds.

The mean values of the number of generated hyperboxes, training time, parameter-tuned time, and testing error for each learning algorithm on four testing folds using different datasets were shown in Table A.2 in Appdendix A.

Regarding training time, it is seen that Simpson's learning algorithm in the FMNN is fastest, while the AGGLO-2 is slowest. The online version of the GFMMNN executes more rapidly compared to improved versions of the FMNN such as the EFMNN or KNEFMNN. It can be seen that the EFMNN using the K-nearest hyperbox selection runs faster than the EFMNN in some cases, but in general it is slower than the EFMNN with optimized parameters. In terms of parameter-tuned time, the KNEFMNN is slowest in most cases, but on medium-sized datasets such as *Ringnorm*, *Twonorm*, and *Waveform*, the time to find the best parameters of AGGLO-2 is longest. Therefore, the current version of AGGLO-2 algorithm should not be used for tuning parameters in an automatic manner in cases of large-sized training datasets.

The number of hyperboxes generated by the learning algorithms of the GFMM is fewest in general, while the EFMNNN and the original FMNN produce the largest number of hyperboxes. The use of K-nearest hyperbox selection rule in the

KNEFMNN also helps considerably reduce the number of hyperboxes created by the EFMNN. It can be observed that the GFMMNN and KNEFMNN generate quite fewer hyperboxes compared to the FMNN or EFMNN since they consider many current hyperboxes for the expansion conditions before creating new hyperboxes. K hyperboxes are taken into account in the KNEFMNN, and as many hyperboxes as possible are considered in the GFMMNN, whereas the FMNN and EFMNN produce a new hyperbox when the winner hyperbox does not meet the expansion constraints.

Generally, the KNEFMNN reduces the number of generated hyperboxes and increases the accuracy of the EFMNN on the considered datasets. The best classification performance belongs to the KNEFMNN, and the online version of GFMMNN and the EFMNN achieve the worst classification results. It can be observed that, on average, only AGGLO-2 and KNEFMNN refine the accuracy of the original FMNN using optimal parameter configurations, but their training time increases substantially. Althout the AGGLO-2 is a promising learning algorithm, its running time is still long on the large datasets. Therefore, many research efforts should be put on improving this algorithm.

It can be easily observed that the number of generated hyperboxes in fuzzy min-max classifiers is large because the best performance of models is achieved for a small value of θ. As shown in the example in Section 3.3, small values of the maximum hyperbox size result in complex models, which are more likely to overfit the training data. Therefore, to assess the efficiency of hyper-parameters selected using density-preserving and cross-validation methods, the models were trained using the same best parameters returned by grid-search procedure on only two DPS folds instead of three DPS folds as in the above experiments. The remaining fold was used as a validation set to conduct the hyperbox pruning. The hyperboxes with the predictive accuracy on the validation set less than a user-defined threshold (0.5 in this work) were removed. It is noted that there are several hyperboxes that do not take part in the pruning process as they have not been used to classify any validation samples (i.e., they have not been the "winners"). Therefore, there is no information about their potential predictive accuracy, and they can be pruned or retained. The decision

of removing or keeping such hyperboxes depends on the misclassification error of the final model on the validation set. If the removal of these hyperboxes leads to the lower error rates on the validation set, they will be pruned, and vice versa.

Table A.3 in Appendix A shows results before and after applying the pruning procedure. The model trained on two DPS folds was verified on the same testing sets as in the previous experiment. It can be seen that the number of hyperboxes after performing the pruning operation is significantly reduced. The pruning procedure contributes to small reduction of the classification errors on four datasets, keeping the same errors on four datasets, and slightly increasing error rates ($< 2\%$) on eight datasets. These outcomes show that the learning algorithms using the best hyper-parameters and training sets generated by the density-preserving sampling method produced the nearly optimal decision boundaries. In such cases, it has been observed that the pruning process can have a small negative effect and can lead to the increase of the testing errors. However, the validation set is also representative of the underlying data distribution, so the error only grows a little. Only for the *Glass* dataset, the error rate increases by around 5% after conducting the pruning operation. This case can be explained by the unrepresentative of the validation set. This dataset has a small number of patterns, while it has a high number of features and classes. Therefore, the samples are sparsely distributed in the input space, and the DPS method may not find the representative subsets. In general, the error rates of models trained on two DPS folds are slightly higher than those of classifiers trained on three DPS folds. These results confirm that the DPS method generated representative subsets for small datasets to assist the learning algorithms. The obtained results also indicate that the overfitting phenomenon on the training set does not always result in the bad predictive performance on unseen data if the training data are representative patterns of the underlying data distribution.

To better understand the performance of fuzzy min-max neural networks, a rigorous statistical significance test procedure will be employed to interpret the obtained results on the considered datasets. Statistical testing was only performed for results of classifiers trained on whole training sets. The null hypothesis is:

H_0: *There is no difference in the performance of different types of fuzzy min-max neural networks on 16 different experimental datasets*

To reject this hypothesis, a "multiple testing" procedure will be used. Two methods regularly used to test the significant differences among multiple samples are a parametric analysis of variance (ANOVA) and its non-parametric counterparts such as the Friedman test. In a survey on the theoretical work of statistical tests, Demsar (2006) recommended that the Friedman test with a relevant posthoc test should be utilized in the case of the comparisons conducted on more than two objects. This chapter employs the Friedman rank-sum test (Eisinga et al. 2017) to evaluate the classification performance statistically because the testing error values of predictors do not follow any symmetric distribution. Firstly, the Friedman rank-sum test ranks the performance of classification algorithms with the best classifier assigned the first rank, and the second best ranked two, etc. Then, the Friedman test performs comparisons on the average ranks of classifiers. Table 3.5 shows testing error ranks over five learning algorithms of different types of fuzzy min-max neural networks as well as the average rank on 16 datasets.

Let r_i^j be the rank of the j-th model in k models on the i-th dataset of N datasets, where k is equal to 5 and N is 16 in this experiment. A null hypothesis as mentioned above states that all algorithms perform similarly, so their average ranks R_j should be equal, and the Friedman statistic

$$\chi_F^2 = \frac{12 \cdot N}{k \cdot (k+1)} \left[\sum_j R_j^2 - \frac{k \cdot (k+1)^2}{4} \right] \tag{3.9}$$

is distributed according to χ_F^2 with $k-1$ degrees of freedom when N and k are big enough, i.e., $N \geq 10$ and $k \geq 5$. Nonetheless, Iman and Davenport (1980) claimed that Friedman's χ_F^2 is undesirably conservative, and they introduced a better new statistic:

$$F_F = \frac{(N-1) \cdot \chi_F^2}{N \cdot (k-1) - \chi_F^2} \tag{3.10}$$

This metric is distributed according to the F-distribution with $k-1$ and $(k-1) \cdot (N-1)$ degrees of freedom. If the null hypothesis is rejected, i.e., the performances of fuzzy min-max neural networks are statistically different, a posthoc test needs

to be carried out to find the critical difference among the average ranks of those models.

Table 3.5 : Testing error ranking of the different FMNN variants

ID	Dataset	Onln-GFMM	AGGLO-2	FMNN	EFMNN	KNEFMNN
1	Circle	2	3	5	1	4
2	Complex9	2.5	2.5	5	2.5	2.5
3	Diagnostic Breast Cancer	5	1	2	4	3
4	Glass	5	4	2	3	1
5	Ionsphere	4	5	3	1.5	1.5
6	Iris	2.5	2.5	1	4.5	4.5
7	Ringnorm	2	1	3	5	4
8	Segmentation	5	4	3	1	2
9	Spherical_5_2	5	1	2	3.5	3.5
10	Spiral	3	3	3	3	3
11	Thyroid	1	5	3	4	2
12	Twonorm	3	2	4	5	1
13	Waveform	2	1	5	4	3
14	Wine	3	4	1	5	2
15	Yeast	4	3	5	2	1
16	Zelnik6	3	3	3	3	3
Average rank		3.25	2.8125	3.125	3.25	2.5625

This chapter uses the 95% confidence interval ($\alpha = 0.05$) as a threshold to identify the statistic significance of fuzzy min-max neural networks. Firstly, the Friedman test calculates the F-distribution:

$$\chi_F^2 = \frac{12 \cdot 16}{5 \cdot (5+1)} \left[(3.25^2 + 2.8125^2 + 3.125^2 + 3.25^2 + 2.5625^2) - \frac{5 \cdot (5+1)^2}{4} \right] = 2.35$$

$$F_F = \frac{(16-1) \cdot \chi_F^2}{16 \cdot (5-1) - \chi_F^2} = \frac{(16-1) \cdot 2.35}{16 \cdot (5-1) - 2.35} = 0.5718$$

With 16 datasets and five classifiers, F_F is distributed according to the F-distribution with $5 - 1 = 4$ and $(5 - 1) \cdot (16 - 1) = 60$ degrees of freedom. The critical value of $F(4, 60)$ for the significance level $\alpha = 0.05$ is 2.5252. It is observed that $F_F < F(4, 60)$, so the null hypothesis is not rejected. It means that there is no statically significant difference in the performance between the general fuzzy min-max neural network and other types of fuzzy min-max neural networks on the considered datasets.

3.4.6 Comparison of GFMMNN and Other Machine Learning Algorithms

This experiment is to compare the classification performance of the GFMMNN with other prevalent machine algorithms such as Naive Bayes, K-nearest neighbors (KNN), Support vector machines (SVM), and Decision trees. These algorithms were implemented by using the scikit-learn toolbox (Pedregosa et al. 2011a) in Python. Similarly to the above experiments, each dataset was also split into four folds using the density-preserving sample technique. Experiments were conducted on each fold as the testing set in turn and three training and validation folds. The validation fold was used to select the parameters leading to the best performance among a range of setting values for each dataset. This process was mentioned in subsection 3.4.5. The configuration parameters for the GFMMNN using incremental and AGGLO-2 learning algorithms were remained unchanged as shown in subsection 3.4.5. As for the value K of the KNN classifier, it was attempted to find the best value in the range of $[3, 30]$. In terms of decision tree models, the tree depth parameter (max_depth) was adjusted ranging from 3 to 30 and unlimited values. For support vector machines, a Radial Basis function (RBF) kernel was used. There are two parameters needing to adjust for RBF kernel ,i.e., the penalty parameter (C) and the parameter gamma (γ). As shown in Hsu et al. (2003), $C \in \{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ and $\gamma \in \{2^{-15}, 2^{-13}, \ldots, 2^{3}\}$ were set. The Gaussian Naive Bayes model has no hyperparameters, so its default settings in the scikit-learn library were used.

Table 3.6 shows the average values of the testing error of different algorithms on four testing folds using the best parameter configurations for each learning model, while Table 3.7 reports the ranks among algorithms.

As indicated in Table 3.7, the best algorithm is SVM, followed by KNN. The highest testing error values belong to the decision trees. The AGGLO-2 algorithm outperforms Gaussian Naive Bayes, decision trees, and the incremental learning algorithm, but it cannot overcome the performances of KNN and SVM in general. These results show that the GFMMNN is competitive to other popular learning models. However, the training and parameter-tuned time of the online and agglomerative

Table 3.6 : Comparison of the average testing errors of the GFMMNN with other machine learning algorithms

ID	Dataset	Onln-GFMM	AGGLO-2	KNN	SVM	Decision tree	Naive Bayes
1	Circle	3.4	3.6	2.8	1.1	4.1	5.7
2	Complex9	0	0	0	0	0.5613	5.279
3	Diagnostic Breast Cancer	4.7463	2.987	2.2848	2.11025	8.6083	6.5018
4	Glass	30.3985	30.3895	28.5028	24.7643	31.3068	52.3933
5	Ionsphere	12.2585	14.2435	12.2485	4.271	10.8088	11.1025
6	Iris	5.299	5.299	3.325	2.6495	5.3343	4.641
7	Ringnorm	13.0405	9.311	23.2298	1.2703	11.2298	1.3378
8	Segmentation	4.1558	3.9825	3.4628	2.4675	3.3768	20.173
9	Spherical_5_2	1.2033	0.8	2.0033	1.6003	0.3968	1.5875
10	Spiral	0	0	0	0	0.1	34.6
11	Thyroid	2.315	3.7215	4.1758	3.7128	5.1103	2.7868
12	Twonorm	4.527	4.33775	2.3918	2.189	15.1215	2.108
13	Waveform	17.88	17.76	13.9	12.74	23.24	18.96
14	Wine	3.952	4.50725	3.38375	1.12375	10.07575	1.69175
15	Yeast	49.3938	49.25875	40.027	37.938	43.8005	88.342
16	Zelnik6	0.4238	0.4238	1.688	0	0.8405	0

Table 3.7 : Testing error ranking of GFMMNN and other machine learning algorithms

ID	Dataset	Onln-GFMM	AGGLO-2	KNN	SVM	Decision tree	Naive Bayes
1	Circle	3	4	2	1	5	6
2	Complex9	2.5	2.5	2.5	2.5	5	6
3	Diagnostic Breast Cancer	4	3	2	1	6	5
4	Glass	4	3	2	1	5	6
5	Ionsphere	5	6	4	1	2	3
6	Iris	4.5	4.5	2	1	6	3
7	Ringnorm	5	3	6	1	4	2
8	Segmentation	5	4	3	1	2	6
9	Spherical_5_2	3	2	6	5	1	4
10	Spiral	2.5	2.5	2.5	2.5	5	6
11	Thyroid	1	4	5	3	6	2
12	Twonorm	5	4	3	2	6	1
13	Waveform	4	3	2	1	6	5
14	Wine	4	5	3	1	6	2
15	Yeast	5	4	2	1	3	6
16	Zelnik6	3.5	3.5	6	1.5	5	1.5
Average rank		3.8125	3.625	3.3125	**1.6563**	4.5625	4.0313

learning algorithms of the GFMM classifier is costly compared to other machine learning algorithms. Therefore, the learning algorithms of the GFMM model need to be enhanced in many aspects to deal with the massive datasets.

Although the average performance ranks of the AGGLO-2 and incremental learning algorithms are not the best ones among learning models, it is desirable to assess the level of differences among obtained results in terms of statistical significance. Similarly to statistical hypothesis tests mentioned above, a null hypothesis in this experiment can be stated as follows:

H_0: *There is no difference in the performance of the general fuzzy min-max neural network and popular machine learning algorithms on 16 different experimental datasets*

The value of F-distribution can be computed as follows:

$$\chi_F^2 = 22.6722$$

$$F_F = \frac{(16-1)\cdot\chi_F^2}{16\cdot(6-1)-\chi_F^2} = \frac{(16-1)\cdot 22.6722}{16\cdot(6-1)-22.6722} = 5.9323$$

With 16 datasets and six classification algorithms, F_F is distributed according to the F-distribution with $6-1=5$ and $(6-1)\cdot(16-1)=75$ degrees of freedom. The critical value of $F(5,75)$ for the significance level $\alpha = 0.05$ is 2.3366. It is observed that $F_F > F(5,75)$, so the null hypothesis is rejected at a high level of significance. Based on these outcomes, it may be stated that there are statistical differences in the performance of the general fuzzy min-max neural network and popular machine learning algorithms.

A post-hoc test is implemented to verify the significant differences of the incremental and agglomerative learning algorithms and other machine learning models. The post-hoc test used in this study is a step down Holm procedure (Holm 1979). The Holm procedure tunes the value of significance level (α) according to a step-down method. Let $p_1, p_2, ..., p_{k-1}$ be the ordered p-values such that $p_1 \leq p_2 \leq ... \leq p_{k-1}$ and $H_1, H_2, ..., H_{k-1}$ be the respective null hypotheses, the Holm procedure rejects null hypotheses H_1 to H_{i-1} if i is the smallest integer such that $p_i > \frac{\alpha}{k-i}$

($\alpha = 0.05$ in this chapter). To find the value of p_i for each pair of predictors, the values of z_i in Eq. (3.11) has to be identified.

$$z_i = \frac{R_i - R_j}{\sqrt{\frac{k \cdot (k+1)}{6 \cdot N}}} \tag{3.11}$$

where i is the control classifier (AGGLO-2 or Onln-GFMM), and j is the another classifier used in the comparisons, R_i and R_j are the average ranks of learners i and j respectively. The probability value of p_i is computed from the corresponding value of z_i following the normal distribution N(0, 1). The calculating outcomes of the Holm procedure are shown in Table 3.8 for the AGGLO-2 algorithm and in Table 3.9 for incremental learning based GFMMNN.

Table 3.8 : Outcomes of Holm post-hoc test for AGGLO-2

i	AGGLO-2 vs.	z_i	p_i	$\frac{\alpha}{k-i}$
1	SVM	2.9764	0.0029	0.01
2	Decision tree	-1.4174	0.1564	0.0125
3	Naive Bayes	-0.6143	0.5390	0.0167
4	KNN	0.4725	0.6366	0.025
5	Onln-GFMM	-0.2835	0.7768	0.05

Table 3.9 : Outcomes of Holm post-hoc test for incremental learning based GFMMNN

i	Onln-GFMM vs.	z_i	p_i	$\frac{\alpha}{k-i}$
1	SVM	3.2599	0.0011	0.01
2	Decision tree	-1.1339	0.2568	0.0125
3	KNN	0.7559	0.4497	0.0167
4	Naive Bayes	-0.3308	0.7408	0.025
5	AGGLO-2	0.2835	0.7768	0.05

From Tables 3.8 and 3.9, it can be observed that $i = 2$ is the smallest integer such that $p_i > \frac{\alpha}{k-i}$. Therefore, H_1 is rejected, while null hypotheses H_2, H_3, H_4, and H_5 are retained. Therefore, AGGLO-2 and incremental learning based GFMMNN

are significantly different from SVM, but there are no statistically significant differences among AGGLO-2, decision tree, Naive Bayes, KNN, and the online version of GFMM at an alpha level of 0.05. These outcomes also indicate that SVM using optimal parameter settings is the best model among considered classifiers. Apart from SVM, learning algorithms of GFMMNN are competitive to popular machine learning models.

3.5 Discussion

This part highlights several notable issues when conducting a comparative study as follows:

- **The impact of hyper-parameters**: Similarly to other machine learning algorithms, the performance of the hyperbox-based classifiers is also dependent on the selection of hyper-parameters, e.g., maximum hyperbox size, etc. Each training dataset needs specific parameters, and a fixed setting should not be used for all datasets. The selection of suitable hyper-parameters should be conducted by combining k-fold cross-validation and sampling methods. The quality of selected hyper-parameters depends mainly on the quality of the training and validation sets. In general, the DPS method helps to preserve the data density and the classes shapes, so the performance of the model trained on small number of DPS folds is not significantly different in comparison to one trained on all DPS folds.

- **Selection of training and validation sets**: Experimental results confirmed the crucial roles of the choice of training and validation data. If a training set which is representative of the overall data distribution can be built for a given problem, a model which overfits on the training sets still performs well on the testing set. The use of the density-preserving sampling method contributes to forming such representative training samples with nearly the same distribution as the whole dataset. The average testing error rates through different density-preserving sampling folds can be used as the generalization error of the model.

Therefore, the hyper-parameters which lead to the lowest error rates on different DPS validation folds may form a trained hyperbox-based classifier with nearly optimal decision boundaries. It is also noted that a model trained on many representative patterns usually achieves higher accuracy than the model trained on a lower number of representative samples. However, if the training sets do not reflect the data density distribution accurately or the constructed model is too complicated, one needs to use overfitting prevention methods.

- **Overfitting prevention mechanisms**: Training model with more relevant and clean data is one of the approaches to restrict the negative impact of overfitting. In practice, however, it is difficult to gather many clean training samples. For a small number of training patterns such as datasets in this chapter, cross-validation and density-preserving sampling, which are the most appropriate methods, allow us to select the best set of hyper-parameters. In some cases, the best hyper-parameters can lead to complex models and make generalization error increase because of its overfitting on the training set. Therefore, several overfitting prevention techniques such as pruning should be used to eliminate low-quality hyperboxes. However, this method does not always work for all cases. If the training set is representative of underlying data distribution and the best-selected hyper-parameters form a nearly optimal decision boundary, the pruning operation is more likely to cause the loss of some critical information and increase testing error. In addition, the efficiency of the pruning procedure mainly depends on the quality of validation sets. In the case of sparse data with high dimensionality, a high number of classes, and a low number of samples, the DPS method cannot return the representative datasets, so the pruning operation can result in considerable increase of the testing error rates.

3.6 Summary

This chapter partly addressed Objective 1 proposed in Section 1.2 regarding the assessment of the advantages and drawbacks of the GFMMNN through empir-

ical results in many benchmark datasets. The impact of setting parameters on the classification problems was also presented. Experimental results indicated the competitive performance of the GFMMNN compared to other fuzzy min-max systems as well as popular machine learning algorithms using the best parameter settings for each algorithm. Nevertheless, the training time of the GFMMNN is a factor preventing the applicability of this type of neural network for the massive datasets in real-world applications.

Chapter 4

Improved Online Learning Algorithm for General Fuzzy Min-Max Neural Network

This chapter presents an improved version of the current online learning algorithm for the GFMMNN to tackle existing issues concerning expansion and contraction steps as well as the way of dealing with unseen data located on decision boundaries. These drawbacks lower its classification performance, so an improved algorithm is proposed in this chapter to address the above limitations. The proposed approach does not use the contraction process for overlapping hyperboxes, which is more likely to increase the error rate as shown in the literature. In order to reduce the sensitivity to the training samples presentation order of this new online learning algorithm, an simple ensemble method is also proposed. The main content of this chapter is taken from the following paper (Khuat et al. 2020):

- **Thanh Tung Khuat**, Fang Chen, and Bogdan Gabrys, "An improved online learning algorithm for general fuzzy min-max neural network," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1-9, 2020.

4.1 Introduction

The use of a contraction process affects the accuracy performance of the GFMMNN, as analyzed in Chapter 2 and in Khuat et al. (2021b); Bargiela et al. (2004) as well as being illustrated in Figure 4.1. As can be seen from Figure 4.1, some data points such as $\boldsymbol{E}$ and $\boldsymbol{F}$ belong to wrong hyperboxes after performing the contraction procedure. Other points such as $\boldsymbol{P}$ and $\boldsymbol{Q}$ are moved to outside of the hyperboxes, and they are more likely to be misclassified. For example, sample $\boldsymbol{P}$ covered by hyperbox $\mathbf{B_2}$ before conducting the contraction procedure is now near to hyperbox $\mathbf{B_1}$.

Therefore, it will be classified to a red class represented by hyperbox $\mathbf{B_1}$ because the membership degree of $\boldsymbol{P}$ in $\mathbf{B_1}$ is higher than that of $\boldsymbol{P}$ in $\mathbf{B_2}$.

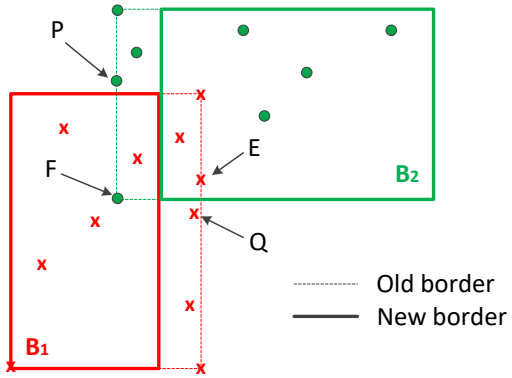


Figure 4.1 : A drawback of the contraction procedure.

Due to the observed undesired effects of the contraction step, other types of fuzzy min-max classifiers such as the inclusion/exclusion fuzzy hyperbox classifier (Bargiela et al. 2004), hyperbox classifier with compensatory neurons (Nandedkar and Biswas 2007a), data-core-based fuzzy min-max classifier (Zhang et al. 2011), and multi-level fuzzy min-max neural network (Davtalab et al. 2014) did not use the contraction phase to resolve the overlapping regions among hyperboxes. Instead, they deployed specialized neurons to handle the overlapping regions. However, these mechanisms make the model architecture complex and increase training time. Therefore, they have been less likely to be applied to large-sized datasets. In this study, rather than using a special neuron for each overlapping region, the expansion of hyperboxes has been prevented if this operation leads to the appearance of overlapping zones. This principle was very successfully used in the agglomerative learning algorithms in Gabrys (2002a), and it is adopted here in the proposed online learning algorithm. Although the agglomerative learning algorithms are efficient and they do not face the limitations of the overlap resolving step, they use all of the training samples to generate and aggregate hyperboxes repeatedly, so their training time is long. Meanwhile, the online learning algorithm uses a single pass learning mode through the training data, thus its training time is much shorter. This chapter proposes to use the learning principle of the agglomerative algorithms in a single pass

learning mode to construct an improved online learning algorithm for GFMMNN, denoted by IOL-GFMM. A strong point of the proposed method is demonstrated through an example presented in Figure 4.2.

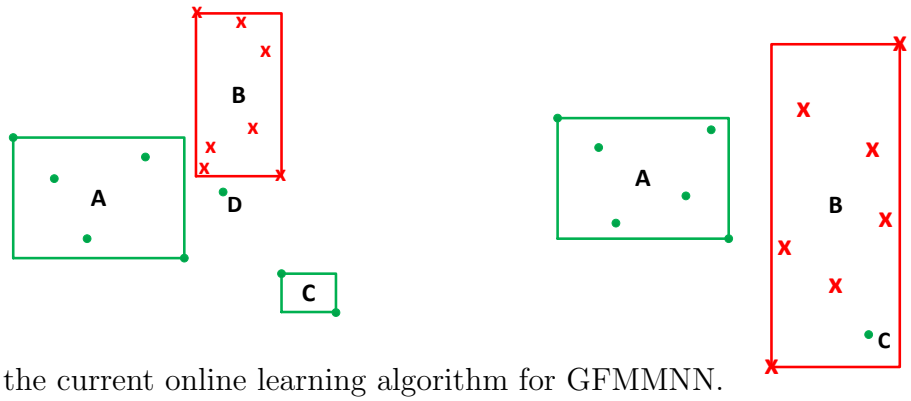


Figure 4.2 : A drawback of the current online learning algorithm for GFMMNN.

In this figure, the original online learning version of GFMMNN (Onln-GFMM) will expand hyperbox $\boldsymbol{A}$ to cover the new input patter $\boldsymbol{D}$, and then the contraction process is performed to resolve the overlapping region with hyperbox $\boldsymbol{B}$. However, if the IOL-GFMM is used, hyperbox $\boldsymbol{C}$ can join the expansion step to cover the input sample $\boldsymbol{D}$ because the expansion of hyperbox $\boldsymbol{A}$ causes the overlap with $\boldsymbol{B}$ representing other class. This process will not lead to any disturbance of the model due to the contraction phase.

As mentioned in Boucheron et al. (2005), a classifier is considered stable if its performance is not varied too much with the small perturbations, e.g. noise, in the training samples. Fuzzy min-max classifiers using online learning algorithms are sensitive to noise, especially when the value of maximum hyperbox size is large. Taking the case shown in Figure 4.3 as an example, with a large value of maximum hyperbox size, hyperbox $\boldsymbol{A}$ will be extended to cover the noisy input patterns $\boldsymbol{C}$ and $\boldsymbol{D}$. Then, the contraction process conducted can cause the negative disturbance in the learned classifier. In the case of using the IOL-GFMM, hyperbox $\boldsymbol{A}$ cannot expand and the patterns $\boldsymbol{C}$ and $\boldsymbol{D}$ are considered as new hyperboxes without any disturbance. The robustness of the IOL-GFMM to the noisy data will be verified in the experimental part.

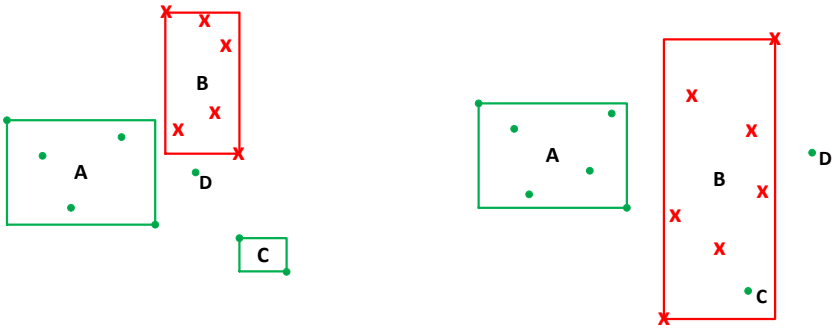
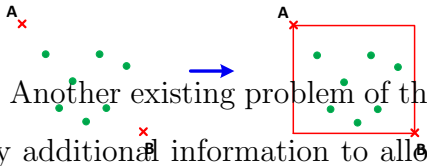


Figure 4.3 : An example of hyperbox-based model and noise.



Another existing problem of the current Onln-GFMM is that it does not provide any additional information to allocate the class for unseen data points locating on the decision boundary. In this case, there may exist several winner hyperboxes with the same membership value representing different classes. To cope with this issue, a solution is the use of a Manhattan measure to compute the distance from the input pattern to the central point of each winner hyperbox as introduced in Upasani and Om (2019), and then assigning the input pattern to the winner hyperbox with the smallest distance. However, the central point is the average value of the minimum and maximum coordinates of the hyperbox, so it is also sensitive to noise. This chapter uses a probability equation generated from cardinality information of each winner hyperbox, as shown in Gabrys (2002c), to decide the suitable class for each input pattern.

With all of the above reasons, the IOL-GFMM is proposed. The main distinct point of the proposed method and the original version is that the proposed method does not use the contraction process. The selected hyperbox is only extended to accommodate a new input pattern if this operation does not introduce any overlapping regions with hyperboxes representing other classes. For the classification phase, cardinality information is used to classify unseen data points in the case of existing many winner hyperboxes with the same maximum membership value and belonging to different classes. However, the IOL-GFMM still depends on the order of training data presentation but to a lesser extent than the original Onln-GFMM. With more data seen, the space is becoming more constrained, and new incoming

patterns located inside the previously created hyperboxes of other classes cannot be expanded to form hyperboxes. This limitation will be also analyzed in the experimental section and if all training data are stored, a simple ensemble method to tackle this drawback is proposed.

The rest of this chapter is organized as follows. Section 4.2 shows the improved online learning algorithm to address existing downsides of the current learning approach. Section 4.3 describes empirical results and some discussion of the proposed method. Section 4.4 summarizes several key findings in this study.

4.2 Improved Online Learning Algorithm

To address the identified drawbacks in the original online learning algorithm of the GFMM model mentioned in the introduction section, i.e., the problems of overlap resolving and equal membership degree, this chapter proposes an improved version of the original training algorithm. The enhanced algorithm eliminates the contraction process during the training phase, simultaneously using the cardinality information of each hyperbox to support the classification phase.

4.2.1 Training Phase

Similarly to the agglomerative learning algorithm in Gabrys (2002a), the improved learning algorithm of GFMMNN does not allow the overlap to occur between the expanded hyperbox and any hyperboxes belonging to other classes, so it does not need to use the contraction procedure. The learning process contains two main steps, i.e., expansion/creation of hyperboxes and overlap checking. The details of the proposed method are shown in Algorithm 4.1. The key difference of this proposed algorithm with the original online learning algorithm shown in Algorithm 3.1 is highlighted in red color. Two main procedures of the algorithm for each input pattern are described as follows:

Expansion of hyperboxes For each input pattern X, all existing hyperboxes with the same class as X or being unlabeled are sought. After that, the membership

values of the input pattern to all of these hyperboxes are computed (*lines 7-8*). Then, all selected hyperboxes are sorted in descending order of the membership degrees (*line 9*). If there is any hyperbox of which the membership value is one, the expansion procedure is not carried out (*lines 13-16*). Otherwise, each hyperbox candidate is traversed in turns and verified the expansion conditions. If all expansion conditions are met, the size of the selected hyperbox and the number of samples contained in that hyperbox are updated , and then the expansion step continues with next input patterns (*lines 17-27*). If all hyperbox candidates are not satisfied with the conditions, a new hyperbox is created to cover the input pattern and add this hyperbox to the current list of hyperboxes (*lines 29-31*). Two conditions need to be verified, i.e., maximum hyperbox size in Eq. (3.4) and overlap. If the maximum hyperbox size requirement is met, the non-overlapping condition is then checked as follows:

Overlap test The overlap checking occurs between the newly expanded hyperbox and the remaining hyperboxes representing different classes. After expanding the selected hyperbox, if it overlaps with any hyperboxes of other classes, the next hyperbox candidate will be considered. If the extended hyperbox does not overlap with any hyperboxes belonging to other classes, the selected hyperbox will be updated with new size, and the learning process continues with another input patterns. The conditions for the overlap test are the same as in the original GFMMNN presented in subsection 3.2.2 (Chapter 3).

Note that the overlapping areas happening during the hyperbox extension process are not allowed, but if the input pattern is in the form of hyperbox, the overlap between it and existing hyperboxes representing other classes can still occur. It is due to the fact that it is directly added to the current list of hyperboxes without verifying the overlap. However, such hyperboxes cannot be expanded to cover other patterns because they do not meet the non-overlapping condition. As a result, these hyperboxes are more likely to be removed if a pruning step is used. If not, the classification step using additional cardinality information can still classify the unseen

samples correctly because these hyperboxes contain only one sample, which leads to the probability to cover the unseen patterns very small.

Algorithm 4.1 The improved learning algorithm of GFMMNN

Input:

- θ: The maximum hyperbox size
- γ: The speed of decreasing of the membership function

Output:

A list $\mathcal{H}$ of hyperbox fuzzy sets containing minimum-maximum values and classes

Initialize an empty list of hyperboxes: min-max values $\mathcal{V} = \mathcal{W} = \varnothing$, hyperbox classes: $\mathcal{L} = \varnothing$
for each input pattern $X = [X^l, X^u, c_X]$ **do**
  $n \leftarrow$ The number of dimensions of X
  if $\mathcal{V} = \varnothing$ **then**
    $\mathcal{V} \leftarrow X^l$; $\mathcal{W} \leftarrow X^u$; $\mathcal{L} \leftarrow c_X$
  else
    $\mathcal{H}_1 = [\mathcal{V}_1, \mathcal{W}_1, \mathcal{L}_1] \leftarrow$ Find hyperboxes in $\mathcal{H} = [\mathcal{V}, \mathcal{W}, \mathcal{L}]$ representing the same class as c_X or being unlabeled
    $\mathcal{M} \leftarrow$ **ComputeMembershipValue**$(X, \mathcal{V}_1, \mathcal{W}_1, \mathcal{L}_1)$
    $\mathcal{H}_d \leftarrow$ **SortByDescending**$(\mathcal{H}_1, \mathcal{M}(\mathcal{H}_1))$
    Set $\overline{\mathcal{H}_1} \leftarrow \mathcal{H} \setminus \mathcal{H}_1$
    $flag \leftarrow false$
    for each $h = [V_h, W_h, c_h] \in \mathcal{H}_d$ **do**
      if $\mathcal{M}(h) = 1$ **then**
        $flag = true$
        break
      end if
      if $\max(w_{hj}, x_j^u) - \min(v_{hj}, x_j^l) \leq \theta, \forall j \in [1, n]$ **then**
        $W_h^t \leftarrow \max(W_h, X^u)$; $V_h^t \leftarrow \min(V_h, X^l)$
        $isOver \leftarrow$ **IsOverlap**$(W_h^t, V_h^t, \overline{\mathcal{H}_1})$
        if $isOver = false$ **then**
          $V_h \leftarrow V_h^t$; $W_h \leftarrow W_h^t$
          $c_h \leftarrow c_X$ if c_h is unlabeled and c_X is labeled
          $flag \leftarrow true$
          Increase the number of samples contained in h
          break
        end if
      end if
    end for
    if $flag = false$ **then**
      $\mathcal{V} \leftarrow \mathcal{V} \cup X^l$; $\mathcal{W} \leftarrow \mathcal{W} \cup X^u$; $\mathcal{L} \leftarrow L \cup c_X$
    end if
  end if
end for
return $\mathcal{H} = [\mathcal{V}, \mathcal{W}, \mathcal{L}]$

4.2.2 Classification Phase

For an unseen input pattern X, the membership values between X and hyperboxes of the trained model are computed. Then, the input X is classified to the hyperbox with the highest membership value. If many hyperboxes representing K different classes have the same maximum membership value (b_{win}), an additional mechanism needs to be applied to find the suitable class. If $b_{win} = 1$ and $\exists i : n_i = 1$, then the prediction class of X is the class of B_i. Otherwise, an additional formula will be used to find the right class for the input pattern as shown in Eq. (4.1) (Gabrys 2002c). The final class of an input pattern is the class c_k with the highest value of $\mathcal{P}(c_k|X)$.

$$\mathcal{P}(c_k|X) = \frac{\sum_{j \in \mathcal{I}^k_{win}} n_j \cdot b_j}{\sum_{i \in \mathcal{I}_{win}} n_i \cdot b_i} \tag{4.1}$$

where $k \in [1, K]$ and $\mathcal{I}_{win} = \{i, \text{if } b_i = b_{win}\}$ is a set of indexes of all hyperbox with the same highest membership value, $\mathcal{I}^k_{win} = \{j, \text{if } class(B_j) = c_k \text{ and } b_j = b_{win}\}$ is a subset of $\mathcal{I}_{win}$ with indexes for the k-th class, and n_i is the number of samples contained in the hyperbox B_i.

4.2.3 Time Complexity of the IOL-GFMM Algorithm

It can be seen that the IOL-GFMM is different from the original online learning algorithm of the GFMMNN in the contraction step only. Therefore, the time complexity of the IOL-GFMM algorithm is the same as that of the Onln-GFMM algorithm in the worst-case, i.e., $\mathcal{O}(N \cdot \overline{\mathcal{K}} \cdot \overline{\mathcal{R}} \cdot n)$, where N is the number of training samples, n is the number of features, $\overline{\mathcal{K}}$ is the average number of expandable hyperbox candidates considered during the training process, and $\overline{\mathcal{R}}$ is the average number of hyperboxes representing classes different from the class of the current training sample.

4.3 Experiments

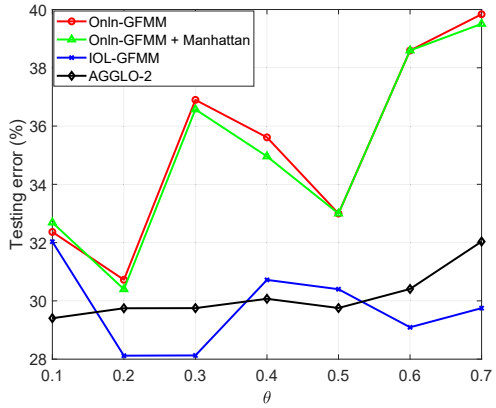
Experiments in this part were conducted on popular datasets stored in UCI repository (Dua and Graff 2019), which are shown in Table B.1 in Appendix B.

Because these datasets have a small or medium size, the density-preserving sampling (DPS) method (Budka and Gabrys 2013) was used to split the datasets aiming to preserve the data density and class shapes for folds employed in training and testing phases. Each dataset was partitioned into four folds using the DPS approach. Three folds were used as training data to build the learning model, and the remaining fold was deployed to evaluate the performance of the trained model. This process was repeated until all folds were used as testing data. The average testing error rates on four folds are reported in this chapter.

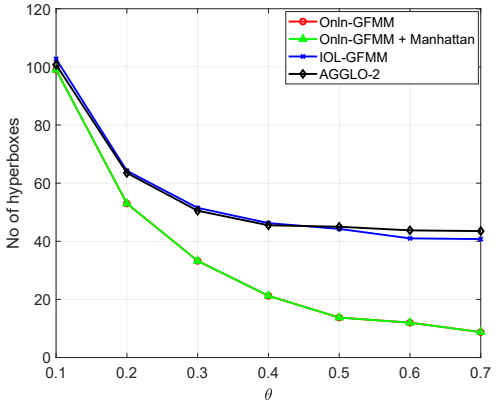
4.3.1 Comparison of the Performance and Complexity of the Proposed and Original Learning Algorithms of GFMMNN

This experiment is to assess the performance and complexity between the proposed method and existing learning algorithms of GFMMNN including online learning (Onln-GFMM) and accelerated agglomerative (batch) learning (AGGLO-2). In addition to the membership value, the original online learning algorithm does not use any additional information to support the classification step. Therefore, in the case that there are many hyperboxes with the same maximum membership degree, classes of the winner hyperboxes are randomly selected to return as the predicted class. To strengthen the comparison with the proposed method, in the original online learning algorithm, the hyperbox central point (the average value of maximum and minimum coordinates) and a Manhattan distance measure (Onln-GFMM + Manhattan) are used to handle the case of many winner hyperboxes as shown in Upasani and Om (2019). The Manhattan distances from the input pattern to central points of the winner hyperboxes are calculated. Then, the input pattern will be assigned the hyperbox with the minimum value of the Manhattan distance. For AGGLO-2, Eq. (4.1) is used to find the predictive class in the case of many winner hyperboxes.

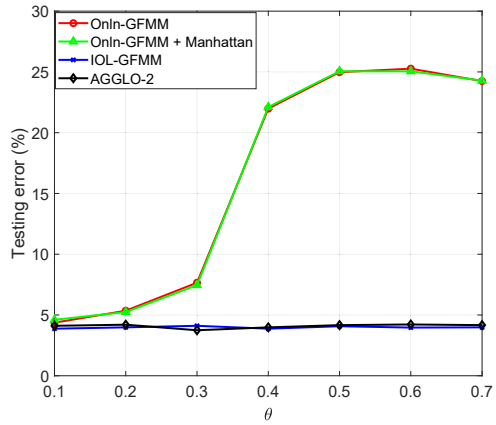
This experiment aims to assess the influences of the maximum hyperbox size parameter (θ) on the classification accuracy, model complexity, and training time of the proposed and original learning algorithms using different datasets. Three representative datasets, i.e., *Haberman*, *Page blocks*, and *Landsat Satellite*, representing



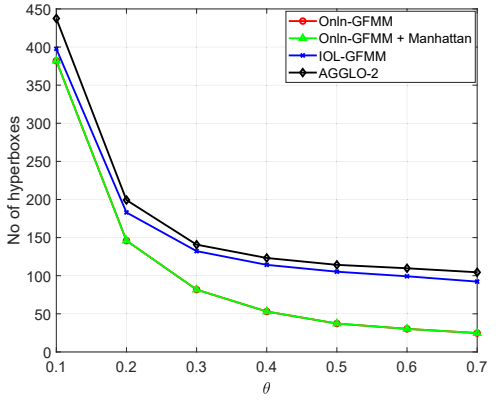
(a) Error rate on the *Haberman* dataset



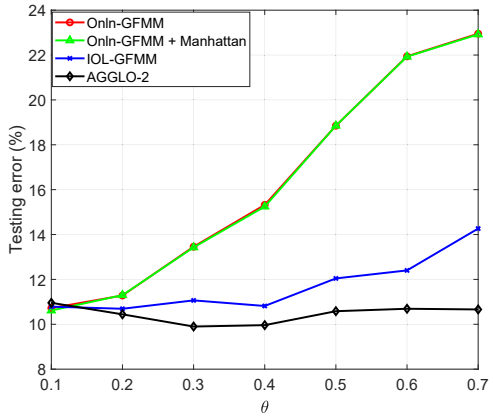
(b) Model complexity on the *Haberman* dataset



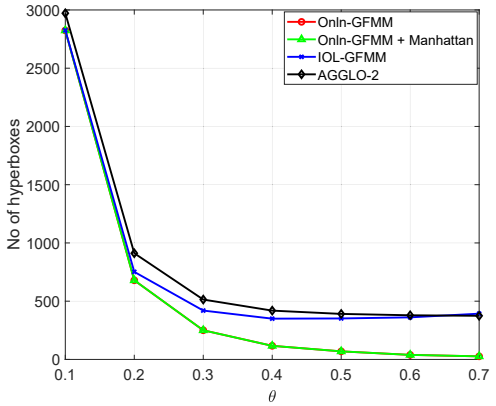
(c) Error rate on the *Page blocks* dataset



(d) Model complexity on the *Page blocks* dataset



(e) Error rate on the *Landsat Satellite* dataset



(f) Model complexity on the *Landsat Satellite* datase

Figure 4.4 : The performance and model complexity of the proposed and original online learning algorithm.

the diversity in the numbers of samples, features, and classes, were selected for this purpose. The values of θ were increased from 0.1 to 0.7, and at each threshold, the models were trained and assessed repeatedly four times on four different training and testing DPS folds. Figure 4.4 shows the average test error rates and complexity (the number of generated hyperboxes) of models on three datasets at different values of θ.

It can be observed that the GFMM classifier trained using the proposed online learning algorithm produces smaller average testing errors compared to the models using the original learning algorithms, especially in the case of employing high values of θ. The use of the Manhattan distance measure contributes to reducing the testing error rates of the model using the original online learning algorithm. While the performances of GFMMNN trained by the original online learning algorithm are severely affected by the increase of maximum hyperbox sizes, the proposed online learning algorithm only causes a slight increase in the testing error rates of model when using large values of θ. These facts can be explained by observing the number of generated hyperboxes at each threshold of θ. At small values of θ, the complexity of the model using the proposed algorithm is nearly equivalent to that of the classifier using the original approach. At higher settings of θ, the numbers of generated hyperboxes of three methods are reduced, but the complexity of the model trained by the improved algorithm is much higher than the models using the original method. As a result, knowledge encoded in the classifier using the improved method is still maintained, so the test accuracy rates are not reduced considerably. In other words, the proposed learning algorithm is efficient in lowering the misclassification errors when increasing the maximum hyperbox size. Compared to the batch learning algorithm AGGLO-2, error rates of the IOL-GFMM are only slightly higher in some thresholds, while its complexity is usually smaller.

Table 4.1 shows the average training time of the original and proposed learning algorithms on three datasets. In general, the training time of the IOL-GFMM is much faster than the original online learning algorithm when the maximum hyperbox size is small (usually ≤ 0.4). It is due to the fact that the proposed learning algorithm

Table 4.1 : Average training time (in seconds) of GFMMNNs using the original and improved learning algorithms (smaller values are better and highlighted in bold)

ID	Dataset	Model	$\theta = 0.1$	$\theta = 0.2$	$\theta = 0.3$	$\theta = 0.4$	$\theta = 0.5$	$\theta = 0.6$	$\theta = 0.7$
1	Page blocks	Onln-GFMM	3.8717	1.9758	1.3024	1.0879	1.1667	1.0385	1.0290
		IOL-GFMM	**0.8238**	**0.4376**	**0.3484**	**0.3432**	**0.3223**	**0.3306**	**0.3313**
		AGGLO-2	15.6012	6.3211	5.1393	5.2248	5.8189	6.2523	6.0002
2	Landsat Satellite	Onln-GFMM	67.1135	27.5141	7.4381	3.3578	**2.2326**	**1.5856**	**1.1288**
		IOL-GFMM	**8.1266**	**2.0505**	**1.7381**	**2.1205**	2.4473	2.7987	3.2385
		AGGLO-2	867.9369	68.8635	46.1044	54.4781	58.4633	66.5776	60.4559
3	Haberman	Onln-GFMM	0.0794	0.0649	0.0547	0.0343	**0.0272**	0.0371	**0.0248**
		IOL-GFMM	**0.0376**	**0.0290**	**0.0275**	**0.0339**	0.0313	**0.0309**	0.0346
		AGGLO-2	0.2493	0.1604	0.1219	0.1385	0.1571	0.1869	0.1812

reduces the number of expandable hyperbox candidates as well as without using the contraction process. In the case that the value of θ is high and the number of data points in the training set is large such as *Landsat Satelite* dataset, the training time of the proposed method is longer than that of the original learning algorithm. This fact is not surprising since the number of generated hyperboxes at high values of θ using the proposed method is much larger than that using the original learning approach (see Figure 4.4-(f)). Hence, the overlap test procedure in the proposed method has to run on much more hyperboxes, so it increases the training time. Furthermore, the training time of online learning algorithms is much faster than that of the agglomerative learning algorithm. These figures show that the IOL-GFMM algorithm is competitive to the batch learning algorithm while its training time is much faster.

4.3.2 Evaluation the Robustness of the Proposed Method to Noise

This experiment is to prove the robustness, as shown in the example in Figure 4.3, of the improved online learning algorithm for GFMM. To evaluate the effectiveness of the IOL-GFMM and the pruning operation in dealing with noise in the learning process with single pass through, three representative datasets, i.e., *Habermann*, *Landsat Satellite*, and *Page blocks* as shown in the subsection 4.3.1, were used. Each dataset was split into four folds using the DPS method; two folds were used

for building the models, one fold for pruning, and one fold for testing. The training and validation sets were corrupted with 5%, 10%, and 15% of the total number of samples changed their class labels randomly. The experiments were repeated four times for four different testing folds to obtain the testing errors before and after conducting the pruning step. This chapter uses a simple pruning operation, in which hyperboxes with predictive accuracy lower than 0.5 are eliminated (Khuat et al. 2021b).

Table 4.2 : Testing errors on noisy datasets in the case of using $\theta = 0.1$

%Noise	GFMM		GFMM-Manhattan		IOL-GFMM	
	B. Pr[a]	A. Pr[b]	B. Pr	A. Pr	B. Pr	A. Pr
Haberman						
0	28.460	27.474	30.421	30.712	28.132	31.707
5	30.404	28.777	29.755	29.401	28.448	27.781
10	32.045	32.702	32.698	31.694	31.066	31.037
15	34.330	31.374	35.646	34.3175	32.698	33.672
Landsat Satellite						
0	11.313	13.458	11.236	11.064	11.360	11.313
5	12.945	14.080	12.883	12.370	12.681	12.417
10	15.090	15.214	15.027	14.577	14.483	14.918
15	17.250	17.125	17.219	14.266	16.503	14.499
Page blocks						
0	4.404	4.824	4.422	6.760	3.874	7.326
5	14.142	13.959	14.306	8.643	4.952	7.875
10	20.263	19.880	20.501	11.565	6.085	8.186
15	27.608	26.768	27.828	15.183	8.240	7.528

[a]B. Pr: Before pruning; [b]A. Pr: After pruning

Experiments were conducted using a small value of $\theta = 0.1$ and a large value of $\theta = 0.7$. Table 4.2 presents the obtained results with $\theta = 0.1$, and Table 4.3 is testing error results with $\theta = 0.7$. It can be seen that the GFMM models using the original online learning algorithms with large-sized hyperboxes ($\theta = 0.7$) are sensitive to noise because the testing error increases significantly at the level of 15% noise on three considered datasets, while they are less sensitive to noise in the case of small-sized hyperboxes. Meanwhile, the IOL-GFMM is less affected by noise even in the case of employing a large value of θ as the error rates only increase $< 7\%$

Table 4.3 : Testing errors on noisy datasets in the case of using $\theta = 0.7$

%Noise	GFMM		GFMM-Manhattan		IOL-GFMM	
	B. Pr[a]	A. Pr[b]	B. Pr	A. Pr	B. Pr	A. Pr
Haberman						
0	28.439	28.439	28.439	39.807	28.777	44.412
5	28.435	28.435	28.435	39.807	30.071	37.239
10	33.002	33.002	33.002	41.554	33.356	38.901
15	47.374	47.374	47.374	37.838	34.638	37.615
Landsat Satellite						
0	24.335	24.351	24.195	35.429	14.437	21.322
5	42.269	42.284	42.269	50.210	14.918	21.259
10	56.736	56.643	56.674	59.346	17.124	21.072
15	59.053	58.913	58.960	66.994	20.357	25.409
Page blocks						
0	28.360	28.396	28.378	42.878	4.002	28.941
5	64.387	64.478	64.387	9.757	4.532	17.304
10	77.928	77.947	78.001	51.993	5.664	24.795
15	81.328	81.310	81.291	81.602	5.865	24.504

[a]B. Pr: Before pruning; [b]A. Pr: After pruning

when there are noisy samples in the training data. In general, the IOL-GFMM is more stable than the original Onln-GFMM, and this claim is demonstrated via the *Landsat Satellite* and *Page blocks* datasets. It can also be seen that the pruning procedure contributes to decreasing the error rates of models trained by the original online learning algorithms in some cases. Nevertheless, it is not effective for IOL-GFMM, especially when using a high maximum hyperbox size threshold because it leads to an increase in the testing error. In general, the pruning procedure is efficient on the model with many small-sized hyperboxes. With a small value of θ and high rate of noise, the pruning step is very useful as it contributes to a significant reduction of error rates. However, in the case of using large values of θ, the pruning step is ineffective for the IOL-GFMM model. This fact confirms that the generated hyperboxes using the proposed method are highly valuable ones, and the removal of these hyperboxes leads to the loss of learned knowledge for unusual patterns.

4.3.3 Comparison of the Performance of the IOL-GFMM to Other Fuzzy Min-Max Classifiers

This experiment is to compare the performance of the GFMM model using the improved online learning algorithm to other types of fuzzy min-max neural networks such as the GFMMNN using the Onln-GFMM (Gabrys and Bargiela 2000) and AGGLO-2 (Gabrys 2002a) algorithms, FMNN (Simpson 1992), EFMNN (Mohammed and Lim 2015), and KNEFMNN (Mohammed and Lim 2017a). To conduct a fair comparison, 3-fold DPS cross-validation and a grid-search procedure were used to find the best value of θ in the range of $[0.1, 0.15, \ldots, 0.65, 0.7]$ for each classifier. As for KNEFMNN, the value of K was searched in the range of [2, 10]. For each training dataset including three DPS folds, a fold was used as a validation set, and two remaining folds were deployed to train the model. Next, the performance of the trained classifier was assessed on the validation fold. This process was iterated three times for three different validation folds for each set of parameters. The parameters which resulted in the lowest average validation error rate were used to train the final classifier using three DPS folds. Finally, the final model was executed on the remaining DPS testing fold. This process was repeated four times for four different DPS testing folds, and the average testing error is reported in Table 4.4.

It can be observed that no method is always superior through all datasets, but the average testing error of GFMMNN using the proposed method is competitive among different methods, such as AGGLO-2 and KNEFMNN. In addition, the IOL-GFMM outperforms other online learning algorithms like Onln-GFMM, FMNN, and EFMNN. To facilitate the conclusion, the performance of approaches is ranked, in which the method outputs the lowest error rates on each dataset is ranked first, the second-best method ranks two, and so on. The average ranks through nine datasets of models are shown in Table 4.5. It can be easily observed that IOL-GFMM produces the best average performance compared to other types of online learning algorithms of the fuzzy min-max models, as well as being competitive to the batch learning algorithm, i.e., AGGLO-2.

Table 4.4 : The lowest average testing errors (%) of models on nine experimental datasets (smaller values are better and shown in bold)

ID	Dataset	Onln-GFMM	Onln-GFMM + Manhattan	IOL-GFMM	AGGLO-2	FMNN	EFMNN	KN EFMNN
1	Blood transfusion	34.358	33.289	**23.797**	24.599	30.883	33.289	32.353
2	Breast Cancer Coimbra	28.448	28.448	32.759	26.724	30.172	37.069	**25.862**
3	Haberman	31.703	32.032	**28.768**	31.374	39.833	34.010	30.379
4	Heart	23.326	19.244	22.218	22.597	**18.157**	19.99	18.882
5	Page blocks	4.550	4.605	**3.746**	3.910	6.998	4.915	4.440
6	Landsat Satellite	11.080	11.018	10.816	**10.303**	17.545	11.826	10.894
7	Waveform	18	18	18.7	**17.92**	22.2	21.22	20.78
8	Yeast	42.116	42.185	41.850	42.989	51.620	40.834	**39.962**
9	Spherical_5_2	1.203	1.203	1.203	1.203	**1.197**	2.407	1.600

Table 4.5 : The average rank of the performance of models through nine datasets (the best value is highlighted in bold)

ID	Model	Average rank
1	Onln-GFMM	4.50
2	Onln-GFMM + Manhattan	4.11
3	FMNN	5.0
4	EFMNN	5.5
5	KNEFMNN	3
6	AGGLO-2	**2.94**
7	IOL-GFMM	**2.94**

4.3.4 A Simple Ensemble Learning Approach to Tackle the Disadvantages of the IOL-GFMM

Although the IOL-GFMM has shown the effectiveness via experiments, especially in the case of using large values of θ, it is still sensitive to the presentation order of training samples like other online learning algorithms. A disadvantage of the IOL-GFMM algorithm is shown in Figure 4.5. In this case, when samples A and B with the same class label are presented to the GFMM classifier first, they will

form a hyperbox. Unfortunately, this hyperbox covers all remaining input patterns of another class. Therefore, when these remaining patterns, which are denoted by dots, come to the network, they do not satisfy the non-overlapping condition between hyperboxes representing different classes. Therefore, all these points form hyperboxes with only one sample. Consequently, all unseen data points with green label located inside the hyperbox with the red label will be incorrectly classified. To overcome this limitation, small-sized hyperboxes should be constructed before aggregating these hyperboxes with a larger size, as shown in Khuat et al. (2021b). In addition, if the training data are available, it can be seen an ensemble method with base learners trained on the same training set but with different data presentation orders in order to overcome the above limitation. By using different orders of training data to build an ensemble model, the limitation presented in Figure 4.5 is less likely to happen on all base learners, so the ensemble classifier is more robust than the single IOL-GFMM model.

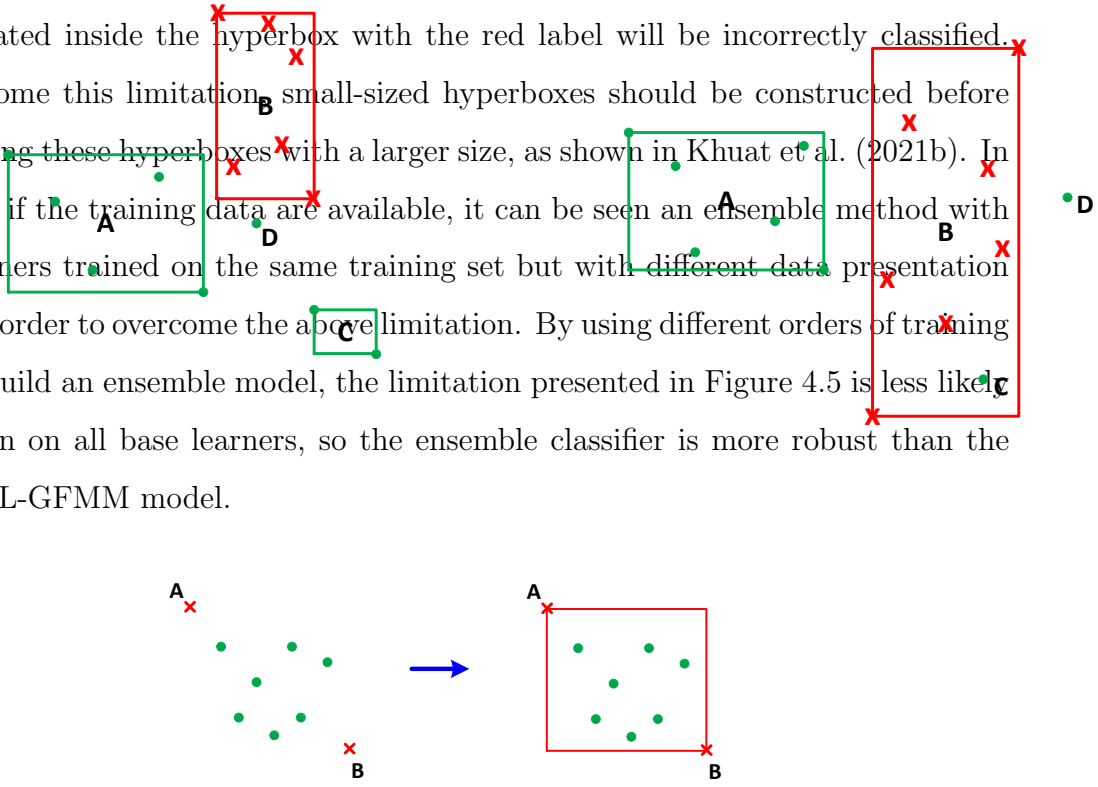


Figure 4.5 : A limitation of the IOL-GFMM algorithm

In this experiment, each dataset was split into four folds using the DPS method (Budka and Gabrys 2013). For each execution, three folds were used for training the model, and then the performance of the trained model was assessed on the remaining fold. The figures reported in this part are the average values of four testing folds. For each training set, eleven GFMM models were built using the original online learning and IOL-GFMM algorithms, each was trained by randomly shuffling the presentation orders of training data. Next, each model was tested on the remaining testing fold. The average standard deviation for testing errors of models using different learning algorithms on four testing folds is shown in Table 4.6. Generally, it can be observed that the standard deviation of models trained by

the IOL-GFMM is lower than those trained by the Onln-GFMM + Manhattan. The standard deviations for testing errors of models using IOL-GFMM are usually lower than 6% and more stable than those using the original online learning algorithm, especially for *Page blocks* and *Blood transfusion* datasets. However, the IOL-GFMM algorithm is still affected by data presentation orders, especially in the case of using high values of θ. In the case of using small values of θ, online learning algorithms of GFMMNN are less influenced by data presentation orders.

Table 4.6 : The average standard deviation (%) of algorithms

ID	**Datasets**	**Algorithm**	$\theta = 0.1$	$\theta = 0.2$	$\theta = 0.3$	$\theta = 0.4$	$\theta = 0.5$	$\theta = 0.6$	$\theta = 0.7$
1	Blood transfusion	Onln-GFMM	3.2647	5.4090	9.3228	10.4414	9.8742	8.1452	8.4970
		IOL-GFMM	2.7962	3.8013	4.3108	5.1811	5.0234	4.9881	4.5658
2	Breast Cancer Coimbra	Onln-GFMM	0	3.2960	6.3032	5.8891	6.4826	6.3821	6.4206
		IOL-GFMM	0	3.2960	5.1123	5.020	6.0098	5.3864	5.9864
3	Haberman	Onln-GFMM	2.4569	3.8980	5.0810	5.0356	7.1989	10.4423	12.4998
		IOL-GFMM	2.4369	3.3359	3.2321	3.5610	3.9824	4.4686	4.3484
4	Heart	Onln-GFMM	0	0	0	0.9722	1.9509	2.1942	1.9381
		IOL-GFMM	0	0	0	0.4791	1.5416	1.8697	1.620
5	Page blocks	Onln-GFMM	0.3658	0.5107	2.6319	10.6540	12.5095	15.3907	18.3566
		IOL-GFMM	0.4321	0.3701	0.4721	0.4981	0.5522	0.5154	0.5127
6	Landsat Satellite	Onln-GFMM	0.3221	0.5147	0.9505	1.0422	1.2635	2.3933	3.7008
		IOL-GFMM	0.2922	0.4713	0.4986	0.820	0.8033	0.7444	0.7862
7	Waveform	Onln-GFMM	0	0.9443	0.8290	0.7722	0.9021	1.4253	1.5462
		IOL-GFMM	0	0.9593	0.7593	0.8792	1.1903	0.9754	1.1182
8	Yeast	Onln-GFMM	1.3597	1.7120	2.5940	2.560	2.7802	3.9750	2.6742
		IOL-GFMM	1.4396	1.9833	1.9490	2.7558	2.1458	2.5104	2.6816
9	Spherical_5_2	Onln-GFMM	0.9206	0.8130	0.7029	0	0	0	0
		IOL-GFMM	0.8705	0.8130	0.7029	0	0	0	0

To reduce the impact of data presentation orders, it can be built a simple ensemble model of base learners trained on the same training set but with different training sample orders. After that, the predictive results of base learners are aggregated using a majority voting method. Table 4.7 shows the average testing errors of eleven base learners and the ensemble model of these eleven base models trained by the IOL-GFMM algorithm. In general, the performance of the ensemble model is better than the single models. With high values of θ, the standard deviation values of base models are usually high, but the ensemble model shows its superior effectiveness in

Table 4.7 : Average testing errors (%) of the IOL-GFMM and the ensemble method

ID	Datasets	Algorithm	$\theta = 0.1$	$\theta = 0.2$	$\theta = 0.3$	$\theta = 0.4$	$\theta = 0.5$	$\theta = 0.6$	$\theta = 0.7$
1	Blood transfusion	IOL-GFMM	27.8926	27.4307	28.3058	29.8128	29.1687	29.0107	29.0107
		Ensemble of IOL-GFMMs	24.3316	24.4652	24.3316	25.1337	23.5294	24.8663	25.4011
2	Breast Cancer Coimbra	IOL-GFMM	31.8966	28.4483	28.6834	28.3699	29.4671	29.3887	29.3887
		Ensemble of IOL-GFMMs	31.8966	28.4483	25	24.1380	24.9999	25.0000	25.8621
3	Haberman	IOL-GFMM	30.0632	30.0686	29.6282	29.5936	31.5032	31.2057	32.6539
		Ensemble of IOL-GFMMs	30.0666	27.1232	26.8113	25.487	28.439	27.1318	29.4173
4	Heart	IOL-GFMM	21.8503	22.2234	21.8558	21.5914	22.1232	21.7480	20.1932
		Ensemble of IOL-GFMMs	21.8503	22.2234	21.8558	21.4882	21.8503	20.3633	18.8707
5	Page blocks	IOL-GFMM	4.1244	4.0115	4.1061	4.0762	4.3320	4.2589	4.3570
		Ensemble of IOL-GFMMs	3.5447	3.3986	3.3803	3.3072	3.2523	3.5265	3.6727
6	Landsat Satellite	IOL-GFMM	10.9006	10.5911	11.1097	12.3019	12.9617	13.4406	13.6680
		Ensemble of IOL-GFMMs	10.3341	9.1064	9.5415	10.7070	11.4218	11.4063	11.7793
7	Waveform	IOL-GFMM	21.94	21.8091	18.9382	18.9327	19.2982	19.4727	19.4073
		Ensemble of IOL-GFMMs	21.94	18.24	15.26	15.04	15.46	15.44	15.7
8	Yeast	IOL-GFMM	40.6408	43.5354	46.2186	49.92493	53.7171	55.7447	58.0855
		Ensemble of IOL-GFMMs	39.0191	39.2839	41.9788	47.1685	52.4907	54.5810	56.0649
9	Spherical_5_2	IOL-GFMM	1.2341	0.9071	1.0502	0.7937	0.7937	0.7937	0.7937
		Ensemble of IOL-GFMMs	1.1967	0.3968	1.1969	0.7937	0.7937	0.7937	0.7937

comparison to base estimators. For example, with $\theta = 0.6$, the standard deviations of the base models on *Blood transfusion* and *Breast Cancer Coimbra* datasets are approximately 5%, whereas the testing errors of ensemble models are lower by about 4% compared to the average testing errors of the single models. This result confirms that the ensemble method is a suitable approach to overcome the limitations of the proposed method and contribute to building robust learning models.

4.4 Summary

This chapter partly addressed the thesis Objective 2 proposed in Section 1.2. It presented the improved online learning algorithm for the GFMMNN. The proposed method does not use the contraction step in the training process or reduce the expandable hyperbox candidates for the expansion step. Experimental results indicated that the performance of the proposed approach outperformed the original learning algorithm, especially in the case of using large values of maximum hyperbox size. The new method also showed the robustness to noise in the training sets.

One of the drawbacks of the proposed and original learning algorithms is that they do not handle categorical features effectively because the current membership function is designed for only continuous values. Therefore, Chapter 5 will address this problem in order to build more robust classifiers.

Chapter 5

Mixed-Attribute Data Classification using General Fuzzy Min-Max Neural Network

One of the downsides of the original learning algorithms for the GFMMNN is the inability to handle and learn from the mixed-attribute data. The popular approaches such as using categorical encoding techniques are not suitable for online learning algorithms working in the dynamically changing environments without ability to retrain or access full historical data, which are usually required for many real world applications. This chapter presents an extended online learning algorithm for the GFMMNN. The proposed method can handle the datasets with both continuous and categorical features. It uses the change in the entropy values of categorical features of the samples contained in a hyperbox to determine if the current hyperbox can be expanded to include the categorical values of a new training instance. An extended architecture of the original GFMMNN and its new membership function are introduced for mixed-attribute data. Important mathematical properties of the proposed learning algorithms are also presented and proved in this chapter. The main content shown in this chapter is taken from the two following papers (Khuat and Gabrys 2020; Khuat and Gabrys 2021b):

1. **Thanh Tung Khuat**, and Bogdan Gabrys, "An online learning algorithm for a neuro-fuzzy classifier with mixed-attribute data," Submitted to *IEEE Transactions on Fuzzy Systems* (Revised and Resubmit).
2. **Thanh Tung Khuat**, and Bogdan Gabrys, "An in-depth comparison of methods handling mixed-attribute data for general fuzzy min-max neural network," *Neurocomputing*, vol. 464, pp. 175-202, 2021.

5.1 Introduction

Classical batch learning algorithms usually require the complete availability of data at the training time. These algorithms do not constantly accommodate new information to the built models. Instead, the model needs to be reconstructed from scratch when the underlying data changes. This operation is time-consuming, especially in the case of massive data, and the constructed models are more likely to be outdated in dynamically changing environments. Taking an advertising recommendation system as an example, this system constructs a customer preference model based on the tracking information about the shopping and browsing behaviors of the users. The buying activities and preferences are temporary and continuously changing. For example, the pandemic events can dramatically change the online shopping behaviors of customers where people tend to purchase things they have never bought before. Therefore, the learning models trained on consumer behavior data prior to the pandemic have been deteriorated or crashed. As a result, these models need to be retrained on new (normal?) behavior data. In this context, and many others characterised by streaming data in changing environments, it is desirable or even necessary to have online learning algorithms that can learn constantly new information without retraining from scratch.

With the increase in the data volume and the rapid change of the environmental conditions nowadays, online learning algorithms are in high demand (Lakshminarayanan et al. 2014; Lughofer and Pratama 2018). These algorithms require smaller or no data storage as they only need one or few newest training samples at one time to rapidly update the constructed model. Hence, the online learning models are ideal candidates for the systems with frequently updating demands. The GFMMNN in this study is a potential candidate which perfectly fits with this requirement. This type of learning model combines the artificial neural networks with the fuzzy set theories to form a consolidated framework. The model creates new hyperboxes or adjusts the existing hyperboxes to cover new samples in its structure. Each hyperbox is defined by the minimum and maximum points in an n-dimensional space. The degree-of-fit of an input pattern to a hyperbox is identified by a mem-

bership function.

Chapters 3 and 4 in this thesis introduced the online learning versions for the GFMMNN. However, both the original online learning algorithm (Gabrys and Bargiela 2000) and the IOL-GFMM algorithm (Khuat et al. 2020) work well on the datasets with only numerical features. To perform classification for the datasets with mixed-type features, it would need to use the encoding methods to transform the categorical values into numerical values. As shown in a recent study (Khuat and Gabrys 2021b), each encoding method has its own drawbacks and except for the CatBoost (Prokhorenkova et al. 2018) and label encoding techniques, all of the remaining encoding approaches need to use the entire training set to encode the categorical features. Therefore, they are not appropriate for incremental learning algorithms, where the new values can appear during the operation time. In addition, according to the empirical results in Khuat and Gabrys (2021b), the classification performance of the online learning algorithms using the CatBoost or label encoding method for the GFMMNN is quite poor. It is because the label encoding method imposes an artificial distance metric for categorical groups, in which this distance is not correspondent to the correlation among original categorical values (Brouwer 2002). Not only this poses a serious problem but the CatBoost encoding method is sensitive to the presentation order of training samples and a shift in the encoded values between training and testing data as well as between training samples have been observed. For the same categorical value, its encoded value in the training data may be distinct from that in the testing data. Even in the training set, the same categorical value may be mapped into many different encoded values depending on the historical patterns prior to the current training pattern. The proposed method in this chapter avoids all of these issues by not using any encoding methods for categorical attributes in the first place.

Many real-world datasets are in the form of mixed-type features. The mixed-attribute data contain both continuous and categorical (or categorical) features. Nowadays, the mixed-attribute data are more and more popular in a wide range of applications from the credit approval data to medical diagnostic data (Huang et al.

2019). Hence, to apply the GFMMNN to such problems, it is necessary to extend its current learning algorithms so that they can deal effectively with mixed-attribute data. Although there are a large number of improved algorithms of the FMNN, only two existing studies have focused on expanding the learning algorithms for both categorical and numerical features as shown in a recent survey paper (Khuat et al. 2021b) and Chapter 2. The first study was proposed in Castillo and Cardenosa (2012) (denoted by Onln-GFMM-M1 in this chapter) using the correlation between the occurrence frequency of categorical values and classes to determine the similarity degrees among categorical values for each categorical feature. After that, the authors proposed to extend the original online learning algorithm (Gabrys and Bargiela 2000) for mixed-attribute data. The second idea of expanding the original online learning algorithm of the FMNN model for both numerical and categorical features was introduced in Shinde and Kulkarni (2016), called Onln-GFMM-M2 in this chapter. It uses the one-hot encoding method for the categorical features and logical operators such as AND and OR to operate on the categorical groups. However, the main weak point of both algorithms is the use of the entire training set to encode or compute the similarity degree between categorical values. If a new value occurs without being encountered during a training process before, these algorithms cannot handle such situation and produce a valid prediction. Different from these two approaches, this chapter proposes a new incremental learning algorithm for both continuous and categorical features. The proposed method does not use any encoding methods for categorical values. Instead, it uses a union operator of a set to add new categorical value to the current set of values in each categorical feature of a hyperbox. The decision on expanding a selected hyperbox to accommodate a new input pattern is based on the change in the entropy for each categorical feature. The membership function is also modified to handle both categorical and numerical attributes. The membership degree for all categorical features is computed from the average probability of categorical values in the input sample with regard to all of the existing categorical values stored in categorical features of the hyperbox.

In short, this study focuses on addressing a classification problem on the tabular

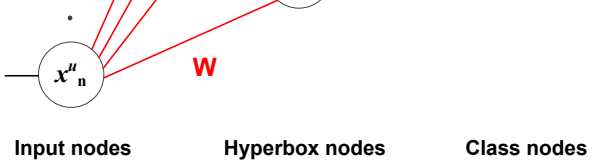
data with mixed continuous and categorical features, where an incremental learning ability of the classifiers is highly desired to avoid retraining these models on a large amount of historical data. The main contribution in this chapter can be summarized as follows:

- A novel online learning algorithm for the GFMMNN able to learn from mixed-attribute data is proposed. This is the first online learning algorithm for the family of fuzzy min-max neural networks which can handle both continuous and categorical features without using any encoding methods.
- Several properties of the proposed method with regard to the categorical attributes are presented and proved.
- Extensive experiments are conducted to prove the effectiveness of the proposed method in comparison to other relevant methods.
- The impact of hyper-parameters on the classification performance of the proposed method is assessed, and a simple method for the parameter estimation is proposed.

5.2 Extended Improved Online Learning Algorithm for Mixed-Attribute Data

5.2.1 Formal Description

Let $\mathcal{T}_N = \{(X_i, c_i)\}_{i=1}^N$, where $X_i = [X_i^l, X_i^u, X_i^d]$, be N training patterns, where c_i is the class of the *i-th* pattern, $X_i^l = (x_{i1}^l, \ldots, x_{in}^l)$ and $X_i^u = (x_{i1}^u, \ldots, x_{in}^u)$ are n continuous attributes (determined in a unit hyper-cube $[0, 1]^n$) of lower bound X_i^l and upper bound X_i^u for the *i-th* training sample, $X_i^d = (x_{i1}^d, \ldots, x_{ir}^d)$ represents r categorical attributes for the *i-th* training sample, x_{ij}^d is a categorical value of the *j-th* categorical feature (A_j^d) at the *i-th* training sample, $x_{ij}^d \in \text{DOM}(A_j^d) = \{a_{1j}, a_{2j}, \ldots, a_{n_j j}\}$, where $\text{DOM}(A_j^d)$ is a domain of categorical values for the categorical attribute A_j^d and n_j is the number of symbolic values of A_j^d. This chapter presents an online learning algorithm to train an efficient GFMM classifier from $\mathcal{T}_N$.



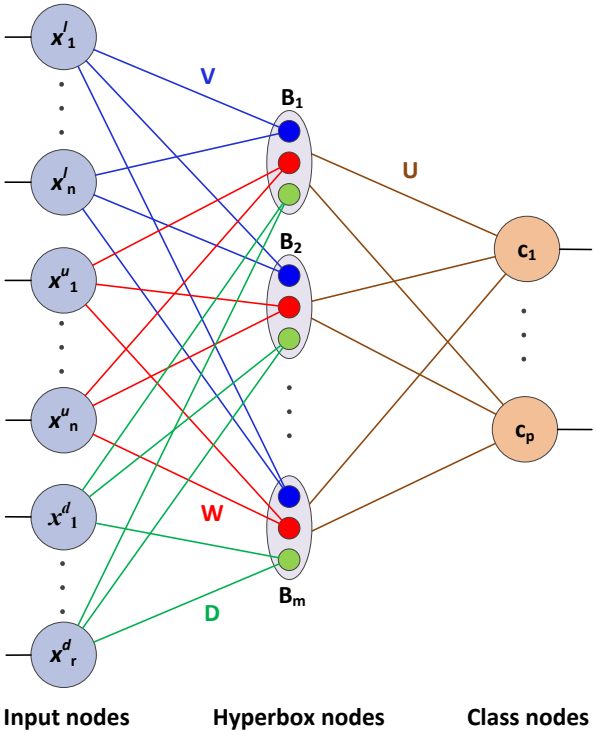


Figure 5.1 : The extended architecture of GFMMNN for mixed-attribute data

5.2.2 Architecture of the GFMMNN for Mixed-Attribute Data

First of all, it is necessary to expand the architecture of the GFMMNN for mixed-attribute data. Instead of using $2n$ input nodes as in the GFMMNN for continuous data, it is requested to have $2n + r$ nodes for the input layer. The first $2n$ nodes are lower bound and upper bound nodes for n numerical features, respectively. The last r remaining nodes correspond to r categorical features in each input pattern. These r input nodes are connected to m hyperboxes by connection weights stored in a matrix $\mathbf{D}$. New architecture of the GFMMNN is shown in Figure 5.1. Beside the minimum point V_i and the maximum point W_i, each hyperbox B_i in the hidden layer also contains a vector D_i storing r categorical-valued sets. Each element $d_{ij} \in D_i$ $(1 \leq j \leq r)$ is a set of categorical values with their cardinalities for the j-th categorical dimension of the hyperbox B_i. For example, $d_{i1} = \{apple : 5, orange : 1\}$ means that the first categorical feature of the hyperbox B_i contains 5 values of *apple* and 1 value of *orange*. The values of vectors V_i, W_i, and D_i for each hyperbox B_i are generated and adjusted during the learning process. The

membership function of each hyperbox B_i with regard to each input pattern with mixed-attribute $X = (X^l, X^u, X^d, c_X)$ is modified as follows:

$$b_i(X,B_i) = \alpha \cdot \min_{j=1}^{n}(\min([1 - f(x_j^u - w_{ij}, \gamma_j)], \\ [1 - f(v_{ij} - x_j^l, \gamma_j)])) + \frac{1-\alpha}{r} \cdot \sum_{j=1}^{r} \mathsf{P}_j(x_j^d \in d_{ij}) \tag{5.1}$$

where $\alpha(\alpha \in [0,1])$ is a trade-off factor regulating the contribution level of continuous numerical features part and categorical features part to the membership score, and $\mathsf{P}_j(x_j^d \in d_{ij})$ is a probability of encountering a categorical value x_j^d in the *j-th* categorical attribute of the hyperbox B_i. This probability is formally defined as follows:

$$\mathsf{P}_j(x_j^d \in d_{ij}) = \frac{|\{a \in d_{ij} | a = x_j^d\}|}{|d_{ij}|} \tag{5.2}$$

where $|\cdot|$ is the cardinality of a set. For the above example, $\mathsf{P}_j(\{orange\} \in d_{i1}) = 1/6$, $\mathsf{P}_j(\{apple\} \in d_{i1}) = 5/6$, and $\mathsf{P}_j(\{banana\} \in d_{i1}) = 0$ are obtained. Unlike the continuous numerical part in the membership function, an average operation for the categorical part is used to reduce sensitivity to the membership value. If the min operator is also used for the categorical part, the membership value for the categorical features will get the value of zero when there is only one categorical feature getting a new categorical value.

5.2.3 Extended Improved Online Learning Algorithm for Mixed-Attribute Data Classification

To create new hyperboxes or adjust existing hyperboxes towards learning mixed-attribute training samples in the GFMM model, it is necessary to expand the current improved learning algorithm presented in Chapter 4, denoted by EIOL-GFMM in this chapter. The proposed modifications include the expansion condition for categorical features, the way of accommodating a categorical value into the hyperbox, and the overlap test for categorical features.

For each training sample, $X = [X^l, X^u, X^d, c_X] \in \mathcal{T}_N$, the algorithm first filters all of the existing hyperboxes representing the same class as c_X. Then, the membership values of X in these selected hyperboxes are calculated and sorted in descending

order. After that, expandable hyperbox candidates are selected in turn starting from the hyperbox with the highest membership degree if the highest membership score is smaller than one. Assuming that B_i is the currently considered hyperbox, the numerical features of B_i are checked for the maximum hyperbox size condition as shown in Eq. (3.4). If the expansion condition for continuous features is satisfied, the algorithm continues to check the constraint for categorical features.

Entropy-based measures can be used to assess the heterogeneity of data in clusters, and they are appropriate for clustering of categorical data due to the lack of explicit distance measures between categorical values (Li et al. 2004). The change in the entropy value of categorical values contained in the hyperbox is used to decide whether the current hyperbox can be expanded to accommodate the categorical values of a new training sample. Given a categorical attribute j, let $H_j(B_i)$ be the current entropy of hyperbox B_i for the *j-th* categorical feature, computed from the probability of all current categorical values stored in the *j-th* attribute as follows:

$$H_j(B_i) = -\sum_{l=1}^{N_j} \mathsf{P}_j(a_l \in d_{ij}) \cdot \log_2 \mathsf{P}_j(a_l \in d_{ij}) \tag{5.3}$$

where N_j is the number of different categorical values (a_l) in the *j-th* attribute, and P_j is defined in (5.2). It is clear that if a new sample is added to the hyperbox for which most of the sample's categorical values existed in the categorical attributes of the hyperbox, the change in the entropy of that hyperbox is small. In contrast, if a sample is added into the hyperbox for which most of the sample's categorical values are new symbolic values to the set of existing categorical attributes of the hyperbox, the homogeneity of this hyperbox is significantly changed, and so the entropy will increase. As a result, the change in the entropy of the hyperbox can be used as an expansion condition for categorical attributes. This entropy changing value is defined in Eq. (5.4) for each categorical feature j of each hyperbox candidate B_i.

$$Z_j = H_j(B_i \cup \{X\}) - \frac{n_i}{n_i + 1} H_j(B_i) \tag{5.4}$$

where $H_j(B_i \cup \{X\})$ is the entropy of the hyperbox B_i in the *j-th* attribute after putting the input pattern X into this hyperbox B_i, computed using Eq. (5.3); n_i

is the number of samples contained in the hyperbox B_i (n_i is also equal to the summation of cardinalities of categorical values in the dimension j).

Based on Z_j, there are two ways to construct the expansion condition for categorical attributes:

- The first method is similar to the expansion condition for continuous features. The extended algorithm using this way is denoted by EIOL-GFMM-v1 in this chapter. It is required the change in the entropy for every categorical attribute smaller than a maximum entropy changing threshold δ for all categorical dimensions:

$$Z_j \leq \delta, \ \forall j \in [1, r] \tag{5.5}$$

- The second approach to build the expansion condition for categorical features uses a weaker condition compared to the first way. The proposed online learning algorithm adopting this condition is called EIOL-GFMM-v2 in this chapter. This method requires the average change in the entropy of all of the r categorical attributes smaller than a maximum average entropy changing threshold δ:

$$\frac{1}{r} \cdot \sum_{j=1}^{r} Z_j \leq \delta \tag{5.6}$$

If both conditions for categorical and numerical features are met for the hyperbox B_i, it will be temporarily expanded to new coordinates. The expansion of numerical features is performed using Eqs. (3.5) and (3.6). Each categorical feature d_{ij} of B_i is expanded as follows:

$$d_{ij}^{new} = \begin{cases} d_{ij}^{old} \cup \{x_j^d : 1\}, & \text{if } \nexists a_j \in d_{ij}^{old} : a_j = x_j^d \\ d_{ij}^{old}.\mathbf{update}(a_j), & \text{if } \exists a_j \in d_{ij}^{old} : a_j = x_j^d \end{cases} \tag{5.7}$$

where $\mathbf{update}(a_j)$ is a function to increase the number of elements of the categorical value a_j by 1. After that, an overlap checking procedure is performed for the newly expanded B_i to examine whether B_i overlaps with any hyperboxes belonging to other classes. In the improved online learning algorithm for numerical features,

only four overlap test cases are used as in the original online learning algorithm. However, these four cases are not sufficient to identify all potential overlap cases between two hyperboxes. Therefore, in this extended version, a similarity measure between two hyperboxes will be deployed based on their smallest gap introduced in Gabrys (2002a) to check the overlap for numerical features between B_i and other hyperboxes B_k representing different classes. This similarity measure s_{ik} is defined in Eq. (5.8).

$$s_{ik} = \min_{j=1}^{n}[\min(1 - f(v_{kj} - w_{ij}, 1), 1 - f(v_{ij} - w_{kj}, 1))] \tag{5.8}$$

where n is the number of continuous features, f is a ramp function given in Eq. (3.2). If B_i and B_k overlap with each other, $s_{ik} = 1$; otherwise, $s_{ik} < 1$. If B_i does not overlap with any B_k representing other classes in the numerical features, it is unnecessary to check the overlap conditions for their categorical features. Otherwise, the overlap has to be verified for the categorical features between B_i and hyperboxes B_k overlapping with B_i in the continuous features. Let Ω_{ij} and Ω_{kj} be the set of categorical values in the *j-th* categorical attribute of two hyperboxes B_i and B_k, respectively. B_i overlaps with B_k in the *j-th* categorical feature if and only if:

$$\begin{aligned} &\Omega_j = \Omega_{ij} \cap \Omega_{kj} \neq \varnothing \text{ and} \\ &\exists a_j \in \Omega_j : \mathsf{P}_j(a_j \in d_{ij}) = \mathsf{P}_j(a_j \in d_{kj}) \end{aligned} \tag{5.9}$$

where P_j is defined in (5.2), and Ω_j contains the common categorical values from two sets Ω_{ij} and Ω_{kj} for the *j-th* categorical feature. B_i overlaps with B_k in categorical attributes if the equation (5.9) is true for all of the r categorical features of these two hyperboxes.

If the hyperbox candidate B_i does not overlap with any hyperboxes B_k representing other classes in either categorical or continuous features, the new coordinates of B_i remain unchanged and the algorithm continues with the next training sample. Otherwise, the coordinates of B_i are reverted to the previous values and the hyperbox candidate with the next highest membership value is selected as an expandable hyperbox candidate, and the above steps are re-iterated.

If none of the hyperbox candidates can be extended to accommodate the input

pattern X, a new hyperbox B_i is generated as follows. For each numerical feature j, $v_{ij} = x_j^l, w_{ij} = x_j^u, \forall j \in [1, n]$ is set, and for each categorical feature j, $d_{ij} = \{x_j^d : 1\}, \forall j \in [1, r]$ is assigned.

The classification phase of the EIOL-GFMM algorithm remains unchanged as in the original IOL-GFMM algorithm. It can be seen that the way of working of the EIOL-GFMM algorithm itself can explain the reason leading to the classification results based on the selection of the hyperbox with the maximum membership degree.

5.2.4 Properties of the Change in Entropy of Categorical Features when Accommodating New Training Samples

This section presents several interesting properties related to the change of the entropy in each categorical attribute of a hyperbox B_i when accommodating a new training sample X. Their proofs can be found in the Appendix C.

Property 5.1. *When covering an input pattern, the change of the entropy in each categorical attribute j of B_i obtains its maximum value if and only if that attribute j includes a new categorical value which does not exist in the list of its current categorical values. Formally,*

$$d_{ij}^{new} = d_{ij}^{old} \cup \{x_j^d : 1\} \Rightarrow Z_j \mapsto \max \tag{5.10}$$

Property 5.2. *The upper bound of the change in the entropy for every categorical dimension j of B_i depends on the current number of samples included in B_i. That is:*

$$Z_j \leq \log_2(n_i + 1) - \frac{n_i}{n_i + 1} \log_2 n_i \tag{5.11}$$

Property 5.3. *The change of the entropy for each categorical dimension j always falls in the range of* [0, 1]*:*

$$0 \leq Z_j \leq 1; \quad \forall j \in [1, r]$$

Property 5.3 also confirms that $0 \leq \delta \leq 1$.

Property 5.4. *When the number of samples contained in B_i approaches infinity, the change of the entropy for every categorical dimension will be limited at 0. Formally,*

$$\lim_{n_i \to +\infty} Z_j = 0, \ \forall j \in [1, r] \tag{5.12}$$

Property 5.4 indicates that when the number of samples included in each hyperbox B_i increases, the expansion condition for categorical attributes of this hyperbox becomes easier to be satisfied.

5.2.5 Time Complexity Analysis for the EIOL-GFMM Algorithm

For a better understanding of its computational complexity, this section provides the time complexity analysis of the EIOL-GFMM algorithm. For each training input pattern, the algorithm first selects $\mathcal{K}$ hyperboxes from the existing set of hyperboxes belonging to the same class as the input pattern. The time complexity of this procedure is $\mathcal{O}(1)$ if a hash table is used to store indices of the hyperboxes corresponding to each class label. To compute the membership values, the algorithm needs to traverse n continuous features and r categorical features for the input pattern and each of the $\mathcal{K}$ selected hyperboxes. Because d_{ij} is a hash table, the complexity of the P_j computation in (5.2) is $\mathcal{O}(1)$. The time complexity of calculations in (3.1) for each continuous dimension is also $\mathcal{O}(1)$, thus the time complexity of the membership values calculation is $\mathcal{O}(\mathcal{K} \cdot (n + r))$. The sorting operation of $\mathcal{K}$ membership values has the time complexity of $\mathcal{O}(\mathcal{K} \cdot \log \mathcal{K})$. Let $\mathcal{R}$ be the number of hyperboxes with class labels different from the class of the input pattern in the current iteration; collecting these $\mathcal{R}$ hyperboxes has the time complexity of $\mathcal{O}(\mathcal{R})$. In the worst case, it is required to loop through $\mathcal{K}$ hyperbox candidates for the verification of expansion conditions and overlap test procedures with $\mathcal{R}$ hyperboxes representing other classes. For each expandable hyperbox B_i, the time complexity to check the expansion condition for n continuous features in Eqs. (3.5) and (3.6) is $\mathcal{O}(n)$. Let q be the maximum number of different categorical values for all categorical features j, i.e., $q = \max_j N_j$. The time complexity of the expansion condition checking for r categorical features using Eq. (5.4) is $\mathcal{O}(r \cdot q)$. Therefore, the time complexity of checking the expansion conditions is $\mathcal{O}(n + r \cdot q)$. The time complexity of

the hyperbox expansion procedure using Eqs. (3.5), (3.6) and (5.7) is $\mathcal{O}(n+r)$. Next, the newly expanded hyperbox B_i is tested for overlap with $\mathcal{R}$ hyperboxes representing other classes. To check the overlap for each pair of hyperboxes, the overlap condition for n continuous features using Eq. (5.8) has the time complexity of $\mathcal{O}(n)$. Because the set of categorical values for each categorical feature is stored using the hash table, the time complexity for the overlap checking operation for all r categorical features using Eq. (5.9) is $\mathcal{O}(r \cdot q)$. As a result, the overall time complexity of the expansion and overlap test processes for each training input pattern is $\mathcal{O}(\mathcal{K} \cdot (n + r \cdot q + \mathcal{R} \cdot (n + r \cdot q))) = \mathcal{O}(\mathcal{K} \cdot \mathcal{R} \cdot (n + r \cdot q))$. If none of the existing hyperboxes can be expanded, a new hyperbox is generated, and the time complexity of this operation is $\mathcal{O}(n+r)$. In summary, let $\overline{\mathcal{K}}$ be the average number of hyperbox candidates with the same class as the input pattern and $\overline{\mathcal{R}}$ be the average number of hyperboxes representing classes different from the class of the input pattern in each iteration, the time complexity of the proposed learning algorithm over N training patterns in the worst case is $\mathcal{O}(N \cdot \overline{\mathcal{K}} \cdot \overline{\mathcal{R}} \cdot (n + r \cdot q))$.

5.3 Experimental Results

The main purposes of the experiments in this section are to

- Analyze the critical roles of parameters α and δ on classification accuracy for the proposed method
- Compare the performance of the proposed method to relevant approaches of the GFMMNN for mixed-attribute data using fixed settings and tuning parameters
- Assess several different methods to estimate the values of α if there have sufficient samples at the training time.

These experiments were conducted on 14 datasets taken from the UCI machine learning repository (Dua and Graff 2019). These datasets were used in Khuat and Gabrys (2021b) to evaluate different methods to handle mixed-attribute data using

the GFMMNN. The same datasets were used to compare the proposed approach to the results presented in that research. The details of these datasets can also be found in Section C.5 in Appendix C. The datasets used in this experiment contain mixed-type features or only categorical features, and so they are different from those used in other Chapters. All of the experimental datasets in this chapter are class-imbalanced, so the class balanced accuracy (CBA) metric was used to assess the performance of classification algorithms. The superior facets of the CBA in comparison to other metrics for the class-imbalanced datasets were shown in Khuat and Gabrys (2021b); Mosley (2013).

5.3.1 Analyzing the Sensitivity of Parameters

There are three important parameters affecting the classification performance of the proposed method, i.e., the maximum hyperbox size for continuous attributes (θ), the trade-off factor (α) regarding the contribution levels of continuous and categorical attributes to the membership function, and the maximum entropy changing threshold (δ) for categorical attributes. The role of θ was analyzed in a recent study (Khuat and Gabrys 2020) and presented in Chapter 3, in which smaller values of θ usually result in better performance than the use of larger values of θ does. However, the smaller values of θ are, the more complex the final model is (i.e. the higher number of generated hyperboxes). This section only studies the influence of two new parameters introduced in the proposed method, i.e., δ and α.

Parameter α

To evaluate the impact of α on the performance of the EIOL-GFMM algorithms, the values of α were changed from 0 to 1 with step 0.1 and recorded the average CBA scores using 10 times repeated stratified 4-fold cross-validation for 11 mixed-attribute datasets. The impact of α is studied for two cases, i.e., large-sized hyperboxes and small-sized hyperboxes. To obtain the large-sized hyperboxes, the parameters $\theta = \delta = 1$ were established so that the expansion process of hyperboxes is not constrained. To achieve small-sized resulting hyperboxes, the small values were used for both θ and δ, i.e., $\theta = \delta = 0.1$ in this experiment. Figure 5.2 shows

the change in the CBA for different values of α in the case of large-sized hyperboxes. For $\delta = 1$, the behaviors of the EIOL-GFMM-v1 and EIOL-GFMM-v2 algorithms are identical. Figure 5.3 presents the change in the CBA results for different values of α in the case of small-sized hyperboxes for both EIOL-GFMM algorithms. Only the results for a representative *flag* dataset are presented in this section. The results of the remaining datasets are presented in Figures C.2, C.3, and C.4 in the Appendix C.

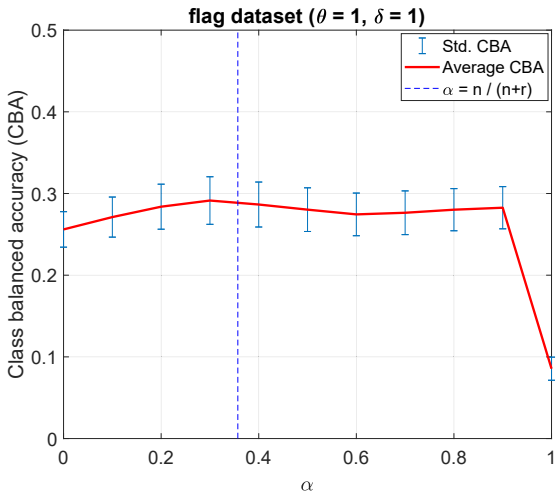
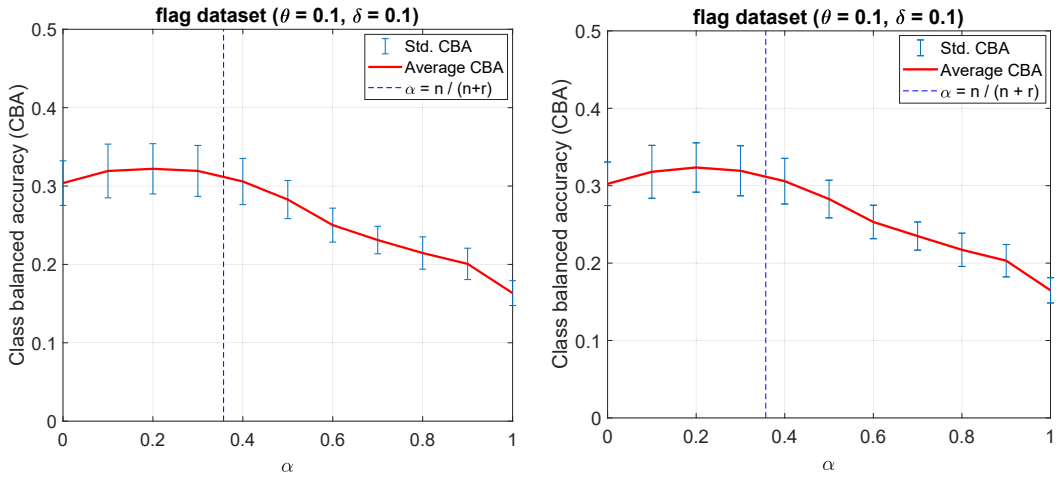


Figure 5.2 : The change in the class balanced accuracy according to the different values of α for the *flag* dataset ($\theta = 1, \delta = 1$).

In general, the CBA values of both proposed learning algorithms at $\alpha = 0$ (using categorical features only) and $\alpha = 1$ (using numerical features only) are usually smaller than the results of using both types of features. The impact of α on both EIOL-GFMM-v1 and EIOL-GFMM-v2 algorithms are similar for many datasets. It can be observed that the influence of α on the GFMM models with small-sized hyperboxes is significantly higher than that with large-sized hyperboxes. This is demonstrated by the degree of oscillation in classification accuracy among different values of α in Figure 5.2 and Figure 5.3. It is because the value of α affects the results of the membership function, and the membership value in turn impacts the selection of the final hyperbox for each unseen pattern. In the case of small-sized hyperboxes, the number of hyperboxes is high and a small change in the membership



(a) EIOL-GFMM-v1 (b) EIOL-GFMM-v2

Figure 5.3 : The change in the class balanced accuracy according to the different values of α for the *flag* dataset ($\theta = 0.1, \delta = 0.1$).

value can lead to a significant change in the selected hyperbox.

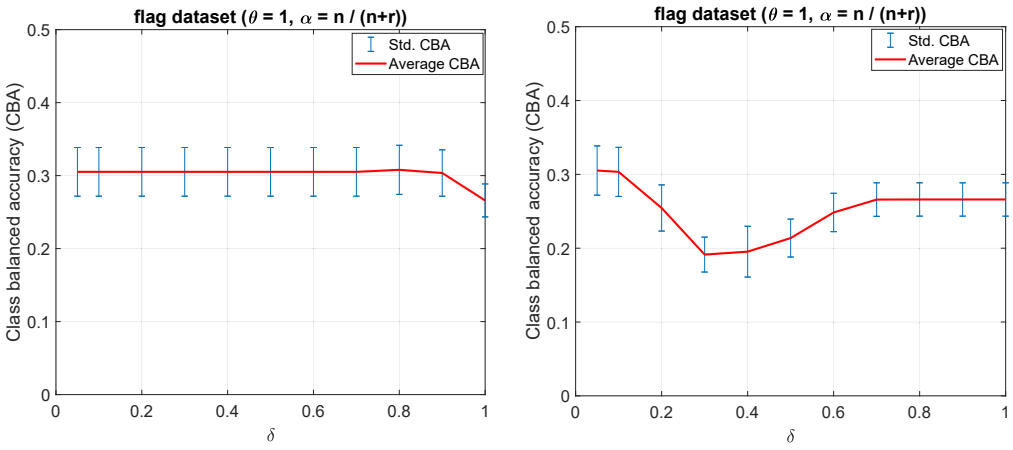
As can be seen, the selection of α can result in the change in the classification performance, thus this parameter needs to be tuned in the learning process. However, for a large number of training samples, performing a hyper-parameter tuning step for α is time-consuming. For the online learning process, one also faces another scenario, where there are not sufficient samples at the training time. In this case, the tuning process cannot be conducted using a cross-validation technique to select an appropriate α for the learning algorithms. Therefore, a fixed setting for α is usually used. From the empirical results in Figures 5.2, 5.3 and Figures C.2, C.3, and C.4 in the Appendix C, it is interesting to observe that the highest CBA results are usually obtained for the value of α near the threshold $n/(n+r)$. Therefore, in the case of using a fixed setting for α, $\alpha = n/(n+r)$ will be set. With this setting, each feature is treated as equally important in decision making.

Parameter δ

This subsection will assess the impact of the maximum entropy changing threshold (δ) for categorical attributes on the classification performance. To rule out the

influence of θ, $\theta = 1$ was set so that numerical features can be expanded without any limitation. From the above experimental results, $\alpha = n/(n+r)$ was used. Therefore, the performance of the learning algorithms depends on the selection of δ. The value of δ was changed from 0.05 to 0.1 and kept the change step of 0.1 up to 1. The impact of δ on the IOL-GFMM-v1 and IOL-GFMM-v2 algorithms is illustrated in Figure 5.4 for the *flag* dataset. The results for the remaining datasets can be found in Figures C.5 and C.6 in the Appendix C.

From these results, it can be observed that the change in the CBA results for the EIOL-GFMM-v1 using $\delta < 0.7$ is very small. For $\alpha \geq 0.7$, its impact on the classification performance of the EIOL-GFMM-v1 on a number of datasets such as *autralian*, *heart*, and *post operative* is significant, especially in the case of $\delta = 1$. In general, the classification error for $\delta \geq 0.7$ in the EIOL-GFMM-v1 is relatively high.



(a) EIOL-GFMM-v1 (b) EIOL-GFMM-v2

Figure 5.4 : The change in the class balanced accuracy according to the different values of δ for the $flag$ dataset $(\theta = 1, \alpha = n/(n+r))$.

For the EIOL-GFMM-v2, however, the change in CBA values is small for $\delta < 0.2$ and $\delta \geq 0.7$. In contrast, the classification performance has significantly changed for the values of δ from 0.2 to 0.7. It can be seen that the impact of δ on the performance of the EIOL-GFMM-v2 algorithm is higher than that on the EIOL-GFMM-v1 algorithm. It is because the hyperbox expansion procedure for categorical

features in the EIOL-GFMM-v2 algorithm can be performed much easier than that in the EIOL-GFMM-v1 algorithm. As a result, the number of generated hyperboxes in the EIOL-GFMM-v1 algorithm is higher than that of hyperboxes in the EIOL-GFMM-v2 algorithm, and so it can capture better the underlying data distribution. For the EIOL-GFMM-v1 algorithm, each categorical feature can only accommodate a new categorical value if the number of samples for the current categorical values is sufficiently large according to Properties 5.1 and 5.2. As a result the homogeneity for each categorical feature of hyperboxes in the EIOL-GFMM-v1 is higher compared to that in the EIOL-GFMM-v2. Therefore, the change in the classification performance among the different values of δ in the first version of the proposed method is smaller in comparison to the second version.

5.3.2 Comparing the Performance of the EIOL-GFMM Algorithms with Other Methods using the Fixed-Parameter Settings

As assuming that there will not be sufficient number of training samples up front in the considered online learning scenarios, as discussed earlier, a fixed setting will be used for certain hyper-parameters which cannot be reliably tuned/optimized using available data. This section is to assess the proposed method in comparison to other solutions to deal with mixed-attribute data for the GFMMNN shown in Khuat and Gabrys (2021b) using previously evaluated fixed values of hyper-parameters. In particular, the proposed method will be compared with two learning algorithms with the mixed-attribute handling ability for the GFMM model including the Onln-GFMM-M1 (Castillo and Cardenosa 2012) and the Onln-GFMM-M2 (Shinde and Kulkarni 2016). The proposed method will also be compared to the use of the original IOL-GFMM algorithm together with various encoding methods for categorical features.

Algorithms with Mixed-Attribute Learning Ability

In Khuat and Gabrys (2021b), the different learning methods were compared to each other using three different settings for θ, i.e., a small size $\theta = 0.1$, a large size $\theta = 0.7$, and an extreme case $\theta = 1$. To compare the proposed method to the

previous solutions, the same settings were also used for the θ parameter. For the γ parameter, $\gamma = 1$ was set as recommended in Abe (2001) for all algorithms. In addition to θ and γ, the existing learning algorithms for the GFMMNN with mixed-feature handling ability have their own hyper-parameters. The Onln-GFMM-M1 algorithm depends on the η parameter, which represents the maximum hyperbox size for categorical features. $\eta \in \{0.1, 0.7, 1\}$ was used as shown in Khuat and Gabrys (2021b). The Onln-GFMM-M2 algorithm has the β parameter to control the minimum number of categorical features matched between the selected hyperbox and the input pattern so that hyperbox can be expanded to cover the input pattern. Similarly to (Khuat and Gabrys 2021b), $\beta \in \{25\%, 50\%, 75\%\}$ of the total number of features for each dataset was used. To be fair in the comparison, the parameter $\delta \in \{0.1, 0.7, 1\}$ and $\alpha = n/(n + r)$ were set for the proposed learning algorithms.

The average CBA values over 10 times repeated stratified 4-fold cross-validation with different parameter settings are shown in Table C.2 in the Appendix C. It can be easily observed that in extreme cases (the largest values of parameters), the classification performance of the proposed method significantly outperforms the Onln-GFMM-M1 and Onln-GFMM-M2 algorithms. To facilitate the comparison of results, for each value of θ, the best results of the remaining parameter will be used to rank four algorithms over 14 datasets. For example, for $\theta = 0.7$, the Onln-GFMM-M1 usually obtains the best performance using $\eta = 0.1$, the Onln-GFMM-M2 achieves its best results with $\beta = 0.75r$, and two proposed methods attain their best results using $\delta = 0.1$. The average ranking of algorithms using their best settings is shown in Table 5.1.

It can be seen that the classification performance of the proposed methods is better than that of two existing algorithms with the mixed-attribute learning ability for three different thresholds of θ. The superior performance of the proposed method compared to the Onln-GFMM-M1 and Onln-GFMM-M2 using fixed parameter settings indicates that the new membership function and entropy-based hyperbox expansion condition, proposed in this chapter, are more suitable for handling categorical features than the mechanisms used in the other evaluated learn-

Table 5.1 : The average rank for the algorithms using their best settings

θ	**Method**	**Other parameters**	**Average rank**
0.1	Onln-GFMM-M1	$\eta = 0.1$	3
	Onln-GFMM-M2	$\beta = 0.75r$	3.214
	EIOL-GFMM-v1	$\delta = 0.1$	1.893
	EIOL-GFMM-v2	$\delta = 0.1$	1.893
0.7	Onln-GFMM-M1	$\eta = 0.1$	2.429
	Onln-GFMM-M2	$\beta = 0.75r$	3.857
	EIOL-GFMM-v1	$\delta = 0.1$	1.607
	EIOL-GFMM-v2	$\delta = 0.1$	2.107
1	Onln-GFMM-M1	$\eta = 0.1$	2.429
	Onln-GFMM-M2	$\beta = 0.75r$	3.857
	EIOL-GFMM-v1	$\delta = 0.1$	1.679
	EIOL-GFMM-v2	$\delta = 0.1$	2.036

ing algorithms. Another advantage of the proposed method is that it can learn in an incremental manner and can include any new categorical values, while the Onln-GFMM-M1 and Onln-GFMM-M2 algorithms need to have all training data to compute the distances between categorical values (as in the Onln-GFMM-M1) or to build fixed size binary one-hot encoding strings (as in the Onln-GFMM-M2). As a result, the Onln-GFMM-M1 and Onln-GFMM-M2 cannot accommodate new categorical values which have not appeared in the training set.

To conclude if there are statistically significant differences among algorithms, a non-parametric test procedure will be carried out as recommended in Demsar (2006) employing the Friedman rank-sum test with a confidence level of 95% (a significance level $\epsilon = 0.05$). The null hypothesis is "there are no statistical differences between learning algorithms", and if this hypothesis is rejected, then the Nemenyi post-hoc test is performed to determine the particular differences. For Z datasets and M algorithms, the Friedman statistic distribution is computed using average ranks $R_j (j \in [1, M])$ of each algorithm j as follows:

$$\chi_F^2 = \frac{12Z}{M \cdot (M+1)} \left[\sum_{j=1}^{M} R_j^2 - \frac{M \cdot (M+1)^2}{4} \right] \tag{5.13}$$

From χ_F^2, a F-distribution with $M-1$ and $(M-1) \cdot (N-1)$ degrees of freedom

can be calculated using (5.14).

$$F_F = \frac{(Z-1)\cdot\chi_F^2}{Z\cdot(M-1)-\chi_F^2} \tag{5.14}$$

The rejection of the null hypothesis occurs with the significance level ϵ if F_F is smaller than a critical value of $F(M-1,(M-1)\cdot(N-1),\epsilon)$. In this experiment, 14 datasets and four learning algorithms were used, so F_F is distributed according to the F-distribution with $4-1=3$ and $(4-1)\cdot(14-1)=39$ degrees of freedom. The critical value of $F(3,39)$ for $\epsilon = 0.05$ is 2.845.

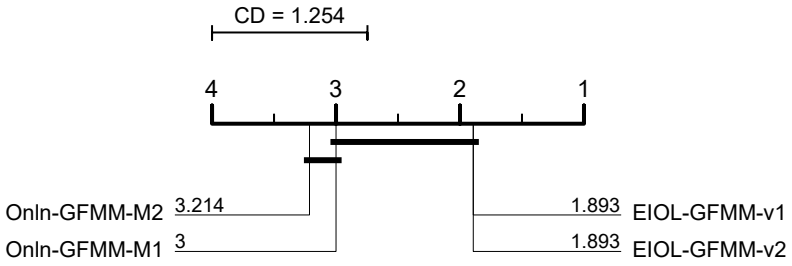


Figure 5.5 : Critical difference diagram for four learning algorithms ($\theta = 0.1$).

For $\theta = 0.1$, $F_F = 5.5539 > 2.845$ is obtained, and so the null hypothesis is rejected. This means that there are significant differences between the results of learning algorithms. Using the Nemenyi post-hoc test, a critical difference (CD) diagram is obtained in Figure 5.5. The groups of algorithms that are not significantly different from each other are connected by a solid line. It can be seen that the proposed methods are statistically better compared to the Onln-GFMM-M2 algorithm with the selected settings. However, there is no statistically significant difference in the classification performance between the proposed methods and the Onln-GFMM-M1 algorithm.

For $\theta = 0.7$, we have $F_F = 16.5238 > 2.845$, and so the null hypothesis is also rejected. Applying the Nemenyi post-hoc test, a CD diagram is constructed as in Figure 5.6. It can be observed that there are no statistically significant differences among the obtained empirical results in the groups of two proposed learning algorithms and the Onln-GFMM-M1 algorithm. However, the algorithms in this group significantly outperform the Onln-GFMM-M2 algorithm.

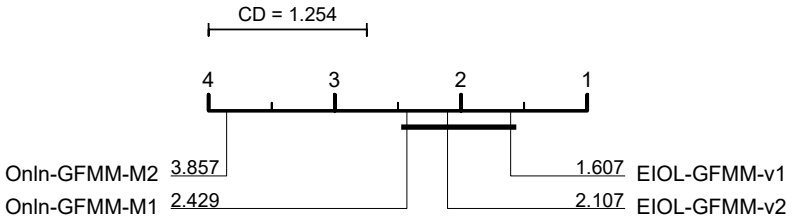


Figure 5.6 : Critical difference diagram for four learning algorithms ($\theta = 0.7$).

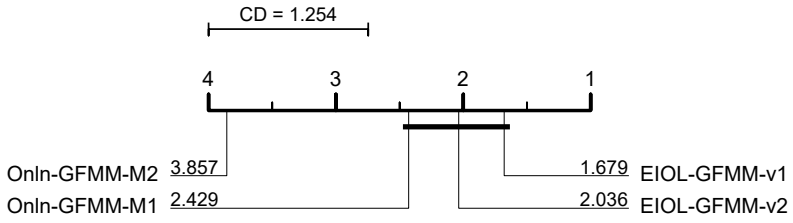


Figure 5.7 : Critical difference diagram for four learning algorithms ($\theta = 1$).

Similarly, for $\theta = 1$, $F_F = 15.7716 > 2.845$ is obtained, and so the null hypothesis is rejected as well. Figure 5.7 shows the CD diagram using the Nemenyi post-hoc test. In this case, the statistical difference in the classification performance among the four methods is the same as in the case of $\theta = 0.7$.

For the complexity of the resulting GFMM models using these learning algorithms, the average number of generated hyperboxes for each method is shown in Table C.3 in the Appendix C. It can be seen that in most of the cases, the complexity of the model using the Onln-GFMM-M2 algorithm is lowest, while the complexity of the GFMMNN using the EIOL-GFMM-v1 is highest. The number of generated hyperboxes using the EIOL-GFMM-v2 algorithm is usually smaller than that using the Onln-GFMM-M1 algorithm. The complexity of the models trained by the Onln-GFMM-M2 is low, because its expansion condition based on the total numbers of matching bits in the binary one-hot encoding strings of categorical values can be easily satisfied compared to the expansion conditions in other methods. In contrast, the expansion condition of the EIOL-GFMM-v1 using Eq. (5.5) is strong and harder to be met for all categorical features, and so the number of expanded hyperboxes is smaller compared to other algorithms. This means that the number of newly

generated hyperboxes in the EIOL-GFMM-v1 is high. The expansion condition of the EIOL-GFMM-v2 using Eq. (5.6) is weaker than that using Eq. (5.5), thus the complexity of the models trained by the EIOL-GFMM-v2 is lower than that trained by the EIOL-GFMM-v1.

Comparing the Proposed Method to the Original Learning Algorithm using Encoding Methods

This subsection will compare the EIOL-GFMM algorithms to the original IOL-GFMM algorithm using different encoding techniques. In Khuat and Gabrys (2021b), there are eight encoding methods used to transform the categorical features into numerical features, i.e., Leave-One-Out (LOO), CatBoost, Label, One-hot, Target, James-Stein, Helmert, and Sum encoding techniques. Similarly to the above experiment, three different thresholds for θ including 0.1, 0.7, and 1 will be considered. To be fair in the comparison, the value of δ was established equal to θ.

The average CBA values over 10 times repeated stratified 4-fold cross-validation of these approaches are shown in Table C.4 in the Appendix C. The average rank for these methods for different thresholds of θ is shown in Table 5.2, in which the best results are highlighted in bold. In general, it can be seen that the average performance of the proposed method is better than that using the original IOL-GFMM algorithm together with the encoding techniques for categorical attributes. One of the strong points of the proposed method is that it does not use any encoding method for categorical attributes. Hence, the proposed method can be applied flexibly to datasets with mixed categorical and continuous features in an incremental learning manner without using any pre-processing methods for categorical features. Meanwhile, the use of encoding methods need to have access to the entire training set at the training time to encode categorical values prior to learning steps. Additionally, each encoding method has its own drawbacks as analyzed in Khuat and Gabrys (2021b), and so the classification performance will be significantly affected if an unsuitable encoding method is deployed. These results indicate a great advantage of the proposed method compared to the use of encoding techniques for categorical features.

Table 5.2 : The average ranks for the proposed method and the original IOL-GFMM using different encoding techniques

Method	$\theta(=\delta)$		
	0.1	**0.7**	**1**
IOL-GFMM + CatBoost	5.357	5.071	5
IOL-GFMM + One-hot	8.179	8.036	7.536
IOL-GFMM + LOO	4.857	4.786	5.643
IOL-GFMM + Label	5.107	5.607	5.464
IOL-GFMM + Target	5.464	4.464	5.429
IOL-GFMM + James-Stein	5.250	4.536	5.214
IOL-GFMM + Helmert	7.893	7.750	7.179
IOL-GFMM + Sum	5.607	5.750	7.821
EIOL-GFMM-v1	3.679	**3.536**	**2.857**
EIOL-GFMM-v2	**3.607**	5.464	**2.857**

Similarly to the above experiments, a statistical test procedure will be used to analyze the statistical difference among the methods. The critical value of $F(9, 117)$ for 10 methods and 14 datasets at $\epsilon = 0.05$ is 1.9608.

For $\theta(=\delta) = 0.1$, $F_F = 4.289 > 1.9608$ is obtained, and so there are statistically significant differences among methods. Using the Nemenyi post-hoc test, a CD diagram is shown in Figure 5.8. It can be seen that the proposed method is significantly better than the original IOL-GFMM algorithm using the one-hot or Helmert encoding method. However, there are no statistical differences between the proposed methods and the IOL-GFMM algorithm using the remaining encoding techniques.

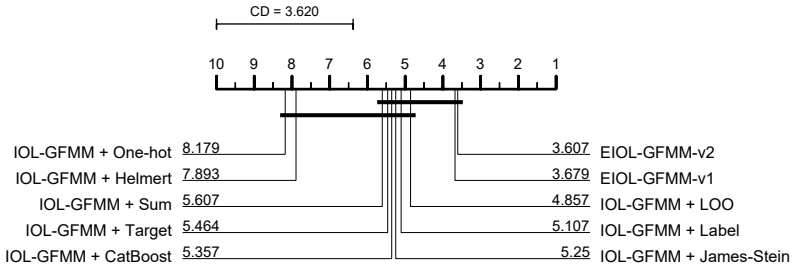


Figure 5.8 : Critical difference diagram for the proposed method and the original algorithm using encoding methods ($\theta = 0.1$).

For $\theta(=\delta) = 0.7$, $F_F = 3.6596 > 1.9608$ is obtained, and so there are also statistically significant differences among methods in this case. Employing the Nemenyi post-hoc test, it can be obtained a CD diagram in Figure 5.9. In this case, the EIOL-GFMM-v1 is significantly better than the original IOL-GFMM algorithm using the one-hot or Helmert encoding method as well, but it does not statistically outperform the original algorithm using the remaining encoding approaches. Moreover, in this case, there is not sufficient evidence to conclude that the EIOL-GFMM-v2 is statistically better than the original algorithm employing encoding methods.

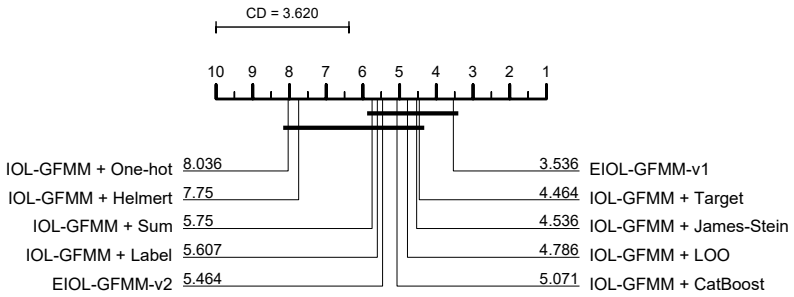


Figure 5.9 : Critical difference diagram for the proposed method and the original algorithm using encoding methods ($\theta = 0.7$).

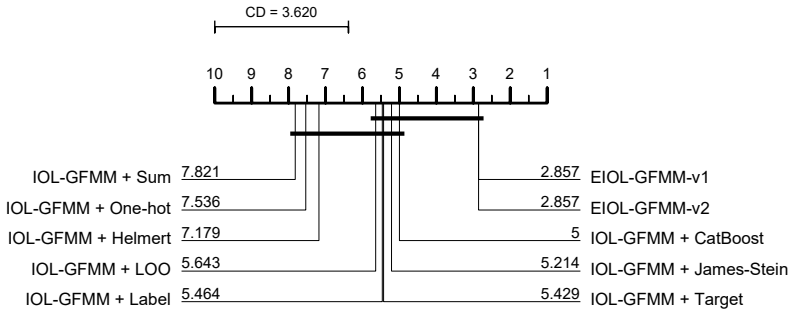


Figure 5.10 : Critical difference diagram for the proposed method and the original algorithm using encoding methods ($\theta = 1$).

For the extreme case $\theta(=\delta) = 1$, $F_F = 6.2138 > 1.9608$ is obtained, and so the null hypothesis is also rejected. Figure 5.10 shows the CD diagram, in this case, using the Nemenyi post-hoc test. It can be observed that there are no statistical differences among the original algorithms using encoding methods. The proposed

methods significantly outperform the original learning algorithm using the one-hot, sum, or Helmert encoding method. However, there are no statistical differences between the proposed methods and the original IOL-GFMM algorithm using the remaining encoding approaches.

5.3.3 Evaluating the Role of the Hyper-Parameter Tuning on the Performance of the EIOL-GFMM Algorithms

Hyper-Parameter Tuning and the Estimation of α

In the case that a large number of samples are given at the training time to build an initial model, a hyperparameter tuning step can be performed for α but this process is time-consuming. Meanwhile, the empirical results in subsection 5.3.1 indicate the relation between suitable values of α and the ratio of the number of continuous features over the total number of features. This subsection proposes a simple way to estimate the appropriate value of α in a data-driven manner. The estimation method does not loop through predefined values of α as in the tuning process, and so they will run faster than the hyper-parameter tuning step for α.

Each training fold T_i is split into three inner folds. To estimate the value of α for each training fold T_i, the learning process will be repeated three times. Each time two inner folds are used to build the GFMM model and the remaining inner fold is used as a validation set. Each inner training fold $T_{ij}(j \in [1, 3])$ is split into two separate parts, in which each part contains either continuous attributes or categorical attributes. Then, two separate GFMM models will be constructed using the EIOL-GFMM algorithm from these two training parts. After that, the CBA value for each trained model is computed using the inner validation fold V_{ij}. Let CBA_{ij1} and CBA_{ij2} be the CBA scores for the GFMM models trained on continuous features only and categorical features only, respectively. There are two ways to estimate the value of α for each training fold T_i. The first way uses both the CBA values and

the number of features, denoted Est-α-v1 in this chapter, as follows:

$$\hat{\alpha} = \frac{\sum_{j=1}^{3} CBA_{ij1} \cdot n}{\sum_{j=1}^{3} CBA_{ij1} \cdot n + \sum_{j=1}^{3} CBA_{ij2} \cdot r} \tag{5.15}$$

The second estimation way of α uses only the obtained CBA values, called Est-α-v2, as follows:

$$\hat{\alpha} = \frac{\sum_{j=1}^{3} CBA_{ij1}}{\sum_{j=1}^{3} CBA_{ij1} + \sum_{j=1}^{3} CBA_{ij2}} \tag{5.16}$$

Two ways of estimating α are summarized in Figure C.1 in the Appendix C. The obtained value of α is used to train a final model using the whole mixed-attribute training fold T_i and evaluate its performance using the *i-th* testing fold. The above process is repeated 40 times (10 times repeated stratified 4-fold cross-validation) to compare the average CBA values among different methods.

This section will compare the effectiveness of two above estimation methods to the fixed setting of $\alpha = n/(n + r)$ and the parameter tuning method for α. In the parameter tuning method, each training fold T_i is split into three inner training folds. Two inner training folds are used to build the GFMM model using the proposed EIOL-GFMM algorithm and the remaining fold is used as a validation fold. This process will be iterated three times to obtain three CBA values from three validation folds for each value $\alpha \in \{0, 0.1, \ldots, 0.9, 1\}$. The value of α resulting in the highest average CBA value over three inner validation folds is used to build the final GFMM model on the whole training fold T_i, and the trained model is assessed by the CBA value on the *i-th* testing fold. The whole process is repeated 40 times for different training folds T_i.

The average CBA results of 40 GFMM models trained using the proposed algorithms with two estimation methods of α, the parameter tuning method and the fixed setting of α for 11 datasets are shown in Table C.5 in the Appendix C. It is noted that the results are reported over 11 out of 14 experimental datasets because these datasets contain both continuous and categorical features while three

remaining datasets consist of only categorical features. The average rank over 11 datasets for different methods of finding the value of α for the GFMM model trained using the proposed algorithm is shown in Table 5.3. Similarly to subsection 5.3.1, the methods of finding α is compared in two cases, i.e., small-sized hyperboxes ($\theta = \delta = 0.1$) and large-sized hyperboxes ($\theta = \delta = 1$). The best rank in each row is highlighted in bold. In the case of $\theta = \delta = 1$, the behavior of both EIOL-GFMM-v1 and EIOL-GFMM-v2 is the same, and so they lead to the same results.

Table 5.3 : Average rank for different methods used to find values for parameter α

Algorithm	$\theta = \delta$	**Tuning** α	**Est-**α**-v1**	**Est-**α**-v2**	$\alpha = n/(n+r)$
EIOL-GFMM-v1	0.1	2.727	**2**	3	2.273
EIOL-GFMM-v2	0.1	2.545	**2.318**	2.727	2.409
Both	1	**2.273**	2.773	**2.273**	2.682

It can be observed that for small values of θ and δ, the estimation method using the CBA values from two separate models along with the number of features usually results in the best average CBA values in comparison to the second estimation method, the parameter tuning approach, and the fixed setting of α for both learning algorithms. However, the second estimation method without using the number of features often leads to the worst results. Interestingly, in this case, the fixed value of $\alpha = n/(n + r)$ shows slightly better results than the hyper-parameter tuning method. In the case of generating the largest hyperbox sizes, the best predictive results belong to the models using the hyper-parameter tuning method and the Est-α-v2 method. Meanwhile, the first estimation method usually leads to the worst classification performance.

To explain these facts, the distribution of the obtained values of α will be examined through 40 iterations and the change in the corresponding CBA values. Figure 5.11 shows the distribution of the obtained α values for different methods in the case of largest-sized hyperboxes for the *flag* dataset. The results in the case of $\theta = \delta = 0.1$ are presented in Figure 5.12. Similar results for all of the remaining datasets can be found in Figures C.7, C.8 and C.9 in the Appendix C.

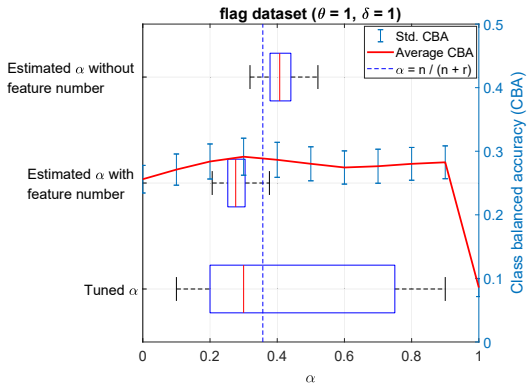
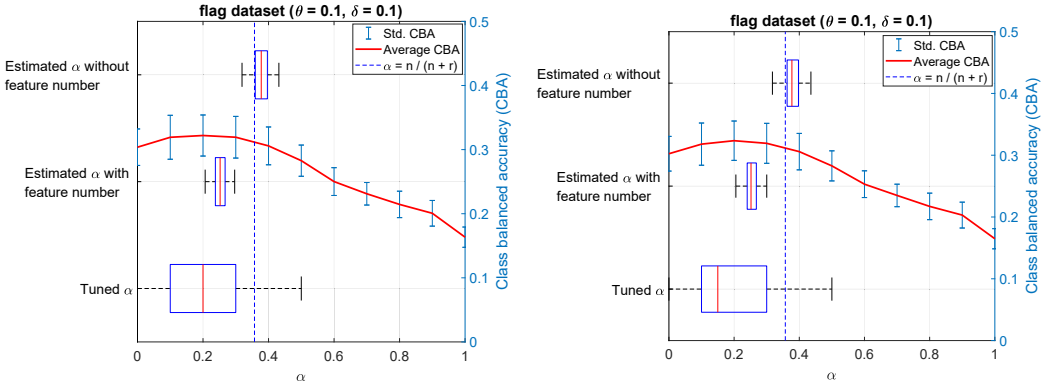


Figure 5.11 : The distribution of the obtained α values for different methods used to find α and the CBA values for the *flag* dataset ($\theta = 1, \delta = 1$).



(a) EIOL-GFMM-v1 (b) EIOL-GFMM-v2

Figure 5.12 : The distribution of the obtained α values for different methods used to find α and the CBA values for the *flag* dataset ($\theta = 0.1, \delta = 0.1$).

It can be seen that, in both cases, the use of the hyper-parameter tuning method returns a wide range of values for α, in which the obtained median value of α locates near the α value resulting in the best classification result. In the case of small-sized hyperboxes, it can be seen that the deviation in the classification results among adjacent values of α is high. Therefore, a wide range of α values usually leads to a low average classification result compared to the use of a narrow range of α values near the best results. As can be seen from Figure 5.12, the obtained α values employing two estimation methods are distributed in a narrower area than that using the hyper-parameter tuning approach. Also, the range of the obtained

α values of the Est-α-v2 is wider than that using the Est-α-v1 method. However, the range of the obtained α values using the Est-α-v1 is nearer the α value leading to the best classification performance than one using the Est-α-v2. Hence, in this case, the Est-α-v1 method usually gives the best classification results among the four methods.

In the case of largest-sized hyperboxes, the difference in the performance among different values of α is small. Therefore, the wide range of the obtained α values using the hyper-parameter tuning method regularly leads to a better average classification result compared to the outcomes employing other methods. As can be seen from Figure C.7 in the Appendix C that two estimation methods return a narrower range of the obtained α values in comparison to the use of the hyper-parameter tuning approach. However, in this case, the obtained α values using the Est-α-v2 usually locate nearer the α values leading to much better performance than those in the case of using the Est-α-v1 method. Therefore, the performance of the GFMM model using the Est-α-v2 outperforms that adopting the Est-α-v1 method.

In short, the second estimation method is appropriate for the model having a small number of hyperboxes, while the first estimation method should be used in the case when the resulting model has a large number of hyperboxes.

Comparing the EIOL-GFMM Algorithms to Other Algorithms with the Mixed-Attribute Learning Ability

In this experiment, the classification performance of the proposed method and two existing algorithms with the mixed-attribute learning ability will be compared using the hyper-parameter tuning procedure (grid-search) for important parameters in each learning algorithms. For the θ value in all learning algorithms, its best parameter value will be sought in the range of $\{0.1, 0.2, \ldots, 0.9, 1\}$ for each training fold. The η parameter for the Onln-GFMM-M1 algorithm is searched in the range of $\{0.1, 0.3, 0.5, 0.9, 1\}$. The searching range of the β parameter for the Onln-GFMM-M2 is $\{10\%, 30\%, 50\%, 70\%, 90\%, 100\%\}$ of the total number of categorical features. For the two proposed algorithms in this chapter, the δ parameter

is searched in the range of $\{0.1, 0.3, 0.5, 0.9, 1\}$, while the α value is sought in the range of $\{0, 0.1, \ldots, 0.9, 1\}$.

Each training fold T_i is split into three inner folds, in which two inner folds are used for training a GFMM model using learning algorithms. Then, the remaining fold is used to obtain the CBA value. This process is repeated three times for every inner validation fold. The combination of parameters resulting in the best average CBA values through three validation folds is used to train the final GFMM model using the whole training fold T_i. After that, this model is evaluated using the corresponding testing fold. This process is iterated 40 times (10 times repeated stratified 4-fold cross-validation) for each dataset. The average CBA results for four learning methods using the above hyper-parameter tuning approach are shown in Table C.6 in the Appendix C. The average rank for each method over 11 mixed-attribute datasets is shown in Table 5.4.

Table 5.4 : Average ranks for the learning algorithms using the hyper-parameter tuning approach

Algorithm	**Average rank**
Onln-GFMM-M1	3.182
Onln-GFMM-M2	2.909
EIOL-GFMM-v1	**1.5**
EIOL-GFMM-v2	2.409

It can be observed that the two proposed learning algorithms outperform two existing learning algorithms with the mixed-attribute handling ability, in which the best performance belongs to the EIOL-GFMM-v1 algorithm. For the experimental results in subsection 5.3.2, it can be seen that the Onln-GFMM-M1 algorithm is better than the Onln-GFMM-M2 algorithm using the fixed-parameter settings. However, by using the hyper-parameter tuning method, the Onln-GFMM-M2 algorithm overcomes the Onln-GFMM-M1. This is because of the difference in the distribution between the inner training set used to find the best combination of parameters and the training fold used to build the final model. The Onln-GFMM-M1 needs to use the entire training data to find the distance between categorical values

based on the relationship between the occurrence frequency of categorical values and classes. These distance values are deployed to build membership functions. Therefore, when the training data change, the best combination of parameters on the inner training folds no longer maintains the superior classification performance when used on the training fold T_i. The proposed methods do not use the training samples to build the similarity measure among categorical features, and so they still achieve the best performance as in the case of using the fixed parameter settings.

Interestingly, the classification performance of learning algorithms using the hyper-parameter tuning method in several datasets such as *cmc*, *cmc*, *zoo*, *australian*, and *japanese credit* is worse than those using fixed parameter settings presented in subsection 5.3.2. This is because the representativeness and distribution of the inner validation sets used to find the best combination of parameters are different from the training and testing folds. Therefore, the best parameters obtained from the inner validation folds may not lead to the best classification accuracy on the testing set. As a result, the hyper-parameter tuning method does not always result in better performance than the use of fixed parameters.

To verify the statistical difference in the performance among the learning algorithms, the above Friedman rank-sum test will be used. For 11 datasets and 4 learning algorithms, F_F is distributed according to the F-distribution with $4-1=3$ and $(4-1)\cdot(11-1)=30$ degrees of freedom. The critical value of $F(3,30)$ at a significant level $\epsilon = 0.05$ is 2.9223. In this case, $F_F = 4.884 > 2.9223$ is obtained. Therefore, there are statistically significant differences among the four considering algorithms. Using the Nemenyi post-hoc test, a CD diagram is achieved as in Figure 5.13.

It can be seen that there is a statistically significant difference in the classification performance between the EIOL-GFMM-v1 and the Onln-GFMM-M1 algorithms in this case. For $CD = 1.414$, it can also be concluded that the EIOL-GFMM-v1 algorithm significantly better than the Onln-GFMM-M2 algorithm. However, the EIOL-GFMM-v2 does not statistically outperform both existing learning algorithms with the mixed-attribute learning ability.

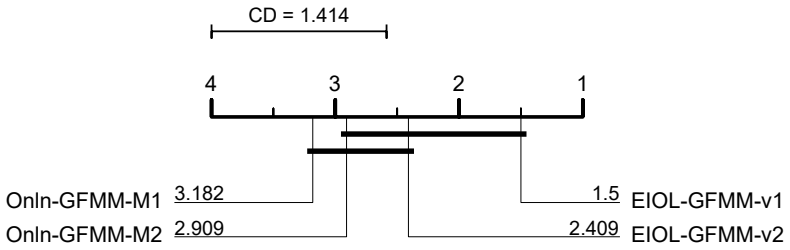


Figure 5.13 : A CD diagram of four learning algorithms using the hyper-parameter tuning method.

5.3.4 Discussion on Existing Issues of the Proposed Method

Although extensive experimental results confirmed very good performance of the proposed method in comparison to other learning algorithms, there are a number of outstanding issues that would need to be addressed in further studies. The first issue is related to the acceleration of the proposed method for data with mixed features by reducing the number of hyperbox candidates in the expansion process, i.e., the value of $\mathcal{K}$, also $\overline{\mathcal{K}}$, in the time complexity analysis. The mathematical lemmas proposed in (Khuat and Gabrys 2021a) to reduce the number of expandable hyperboxes only work on continuous features. Therefore, future studies need to expand these lemmas for categorical features. The second issue is related to the membership function. Similarly to the membership function presented in Eq. (3.1) for the original learning algorithm, the part handling continuous features in Eq. (5.1) only works appropriately when the continuous features are normalized to the range of [0, 1]. As a result, the membership function can be improved so that it can robustly work with any ranges of continuous features' values without need for normalization. For an online learning operation in the dynamically changing environments, the range of feature values can be different from ones in the training data, and so denormalization and normalisation of continuous features as presented in Salvador et al. (2016) are inconvenient and sensitive to outliers. In addition, the current membership function in Eq. (5.1) has to handle continuous and categorical attributes separately and use the α parameter to control the importance of each type of features. Therefore, a unique membership function with the ability to handle both continuous and cate-

gorical features in a natural way is highly desirable. The third issue is related to the process of handling missing values for the categorical features. If the j-th continuous feature of the input pattern X contains a missing value, $x_j^u = 0$ and $x_j^l = 1$ will be set so that the membership value on that continuous feature gets a value of one and does not affect the final membership value of all continuous features when using the minimum operation. This is a very powerful characteristic of the original GFMM learning algorithms working on continuous features only which allows it to process data with missing values without any changes to the learning algorithms (Gabrys 2002c). However, for categorical features with missing values, the missing values need to be considered as an *Unknown* categorical value and handle them as other categorical values. As the exact impact of missing categorical values on the performance of the proposed method is not clear, a separate study needs to be conducted to assess it. All of the above discussed issues are outside of the scope of the current study in this chapter and form a part of future work directions towards making the GFMMNN robust and continuously learning while operating in changing environments.

5.4 Summary

This chapter partly tackled the thesis Objective 2 proposed in Section 1.2, which presented a new online learning algorithm for the GFMMNN with mixed-attribute data. The proposed method expanded the current membership function for both continuous and categorical features. Current architecture of the GFMMNN for mixed-attribute data was also extended, and a new way of learning for categorical dimensions based on the change in the entropy when accommodating new categorical values without using any encoding methods was introduced as well. The experimental results confirmed the superior classification performance of the proposed method in comparison to the current solutions to handle the mixed-type datasets for the GFMMNN.

Chapter 6

Accelerated Learning Algorithms for General Fuzzy Min-Max Neural Network

This chapter presents the methods of accelerating the learning algorithms for the GFMMNN. The first method reformulates and represents learning algorithms in a format allowing for their parallel execution and subsequently leveraging the computational power of the GPUs. The original implementation of the GFMMNN is modified by matrix computations to be executed on the GPUs for the very high-dimensional datasets. The second technique aims to reduce the unsuitable hyperboxes selected as the potential candidates of the expansion step of existing hyperboxes to cover a new input pattern in the online learning algorithms or candidates of the hyperbox aggregation process in the agglomerative learning algorithms. This second method is based on the mathematical formulas to form a novel solution aiming to remove the hyperboxes which are certain not to satisfy expansion or aggregation conditions. The key content of this chapter is taken from the two following papers (Khuat and Gabrys 2019, 2021a):

- **Thanh Tung Khuat**, and Bogdan Gabrys, "Accelerated training algorithms of general fuzzy min-max neural network using gpu for very high dimensional data," in *Proceedings of the 26th International Conference on Neural Information Processing (ICONIP)*, pp. 583-595, 2019.

- **Thanh Tung Khuat**, and Bogdan Gabrys, "Accelerated learning algorithms of general fuzzy min-max neural network using a novel hyperbox selection rule," *Information Sciences*, vol. 547, pp. 887-909, 2021.

6.1 A Solution based on Matrix Operations and Advanced Engineering

6.1.1 Overview

In practical applications, pattern classification of high-dimensional data is a challenging issue. A dataset with N patterns and n features is taken into account as very high dimensional if $n \gg N$, and the value of n is relatively large. The high-dimensional data appear in many real-world applications, especially in genomes data (Tariq et al. 2018) or data sampled from sensor networks as shown in Cuturi (2011); Vergara et al. (2013). In monitoring systems, a network of different sensors is used to supervise the operating statuses. Data of sensor arrays in complex environmental conditions sampled through different time-series exhibit very high dimensionality, even to millions of attributes as in Vergara et al. (2013). The similar phenomena also happen in the gene expression data, where the numbers of samples for both training and testing sets are regularly less than 100, while the number of attributes ranges from 6000 to 60,000 (Tariq et al. 2018). High dimensionality imposes high computational cost for the training process. It is needless to say that the long training time is one of the problems hindering machine learning algorithms from applying to real-world applications. Therefore, a method is proposed to reduce the training time for the GFMMNN by taking advantage of the computational capability of the GPUs and intrinsic parallelization characteristics of GFMMNN.

For leveraging the power of GPUs, one has to provide the data in batches so that it is able to perform many operations in parallel at the same time. Therefore, the matrix operations are built for basic calculation steps in the learning algorithms of the GFMMNN, such as the computation of membership functions, hyperbox expansion conditions, hyperbox overlap checking between two hyperboxes or many pairs of hyperboxes. Coates et al. (2013) showed how to build a system to train the deep neural network with 64 Nvidia GPUs on 16 computers executing the learning process more than 6.5 times faster than the system with 1,000 computers using 16,000 CPUs. These results illustrated the power and potential of deploying the

training phases on GPUs to significantly lower the training time on large-sized or very high dimensional datasets. Motivated by this study as well as to take advantage of the intrinsic parallelizable features of the GFMMNN, a new method of accelerating the training phase of the GFMMNN is proposed by using Pytorch (Pytorch 2021) and GPUs for very high dimensional data.

The main contributions in this part can be summarized as follows:

- Proposing and representing the learning algorithms of the GFMMNN with their various components in a format allowing for parallel execution
- Using matrix computations executed expertly on the GPUs to accelerate the training and testing processes of the GFMMNN
- Implementing training algorithms of the GFMMNN on the GPUs using Pytorch framework
- Comparing the training and testing time of the GFMMNN implemented by the NumPy library and Pytorch on two very high-dimensional practical datasets.

6.1.2 Implementation of Learning Algorithms by Matrix Operations

This study reformulates and represents the learning algorithms of GFMMNN using matrix formats. This representation makes the algorithms execute in parallel effectively. Hence, the GPUs can be deployed to accelerate GFMM algorithms. Two matrices $\mathcal{V} = \{V_1, \ldots, V_N\}$ and $\mathcal{W} = \{W_1, \ldots, W_N\}$ are employed to contain sets of coordinates of lower and upper bounds of N hyperboxes as well as a vector $\mathcal{L} = \{c_1, \ldots, c_N\}$ of hyperbox classes during the training process. Each element V_i of the matrix is an n-dimensional vector $\{v_{i1}, \ldots, v_{in}\}$ maintaining the values of the minimum point of the $i\text{-}th$ hyperbox. The similar setting is formulated for the maximum point of each hyperbox. These matrices and vector are stored on the GPUs and initialized by command $.cuda()$ of Pytorch. The operations of learning algorithms are performed on these operands.

Membership Computation

In the incremental learning algorithm, to find the potential expandable hyperbox, the membership values between the input pattern $X = [X^l, X^u]$ with all hyperboxes belonging to the same class label or being unlabelled have to be computed, which are stored in the matrices $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{W}' \subseteq \mathcal{W}$. Assume that N_1 is the number of hyperboxes represented by $\mathcal{V}'$ (or $\mathcal{W}'$), vectors X^l and X^u corresponding to lower and upper bounds of the input sample are replicated N_1 times to form matrices $\mathbf{X}^l$ and $\mathbf{X}^u$. The vector of sensitivity parameters γ is also copied N_1 times to formulate a matrix $\mathbf{G}$. The Eq. (3.1) is changed to compute the membership degree between the input pattern $X = [X^l, X^u]$ and all hyperboxes representing the same class as follows:

$$B(X^l, X^u, \mathcal{V}', \mathcal{W}') = \min(\min([1 - f(\mathbf{X}^u - \mathcal{W}', \mathbf{G})], [1 - f(\mathcal{V}' - \mathbf{X}^l, \mathbf{G})])) \quad (6.1)$$

The operation $\min(Y, Z)$ generates the resulting matrix whose each element is taken from the minimum value of two corresponding elements within two matrices Y and Z. The procedure $\min(Y)$ is used to find the minimum value within each row of the matrix Z. The details of the membership computation steps are shown in Algorithm 6.1:

Algorithm 6.1 Membership computation on GPUs:

Membership_Computation$(X^l, X^u, \mathcal{V}', \mathcal{W}', \gamma)$

Input:

- X^l, X^u: Minimum and maximum points of an input pattern X
- $\mathcal{V}', \mathcal{W}'$: Matrices storing minimum and maximum points of all hyperboxes representing the same class label as X or being unlabelled
- γ: The speed of decreasing of the membership function

Output:

A list B containing membership values between X and all hyperboxes represented by two matrices $\mathcal{V}', \mathcal{W}'$

1: $N_1 \leftarrow size(\mathcal{V}', 0)$ {*Compute the number of hyperboxes within* $\mathcal{V}'$}
2: Replicate γ, X^l, and X^u N_1 times for each vector to build three matrices $\mathbf{G}$, $\mathbf{X}^l$, and $\mathbf{X}^u$
3: $F_1 \leftarrow 1$ - **F_Ramp**$(\mathbf{X}^u - \mathcal{W}', \mathbf{G})$; $F_2 \leftarrow 1$ - **F_Ramp**$(\mathcal{V}' - \mathbf{X}^l, \mathbf{G})$;
4: $F \leftarrow \min_{element\text{-}wise}(F_1, F_2)$
5: $B \leftarrow \min_{row\text{-}wise}(F)$
6: **return** B

Algorithm 6.2 Computing Ramp function: **F_Ramp**(Y, G):

1: $Z \leftarrow Y \odot G$ $\{\odot$: *Hadamard product*$\}$
2: $F \leftarrow [(Z > 0) \odot (Z < 1)] \odot Z + (Z > 1)$
3: **return** F

Computation of the Similarity among Hyperboxes in the Agglomerative Learning

In the agglomerative learning algorithm, to find the candidate pair of hyperboxes for aggregation, it is required to compute the similarity values among all pairs of hyperboxes representing the same class label in the AGGLO-SM or between currently considered hyperbox and the other hyperboxes representing same class in the AGGLO-2 algorithm. Therefore, Eq. (3.8) can be modified as in Eq. (6.2) to compute the similarity values between the currently considered hyperbox B_i and the other hyperboxes belonging to the same class label shown by matrices $\mathcal{V}'$ and $\mathcal{W}'$.

Let $\mathcal{V}_i^{'}$ be the matrix formed by replicating vector V_i of hyperbox B_i N_1 times, assuming that N_1 is the number of hyperboxes represented by $\mathcal{V}'$. Similar method is performed for W_i to generate $\mathcal{W}_i^{'}$. From Eqs. (6.1) and (6.2), it can be observed that the function **Membership_Computation** can be reused to compute the similarity measure by replacing the parameters X^l and X^u with W_i and V_i respectively.

$$s(V_i, W_i, \mathcal{V}', \mathcal{W}') = \min(\min([1 - f(\mathcal{V}_i^{'} - \mathcal{W}', \mathbf{G})], [1 - f(\mathcal{V}' - \mathcal{W}_i^{'}, \mathbf{G})])) \qquad (6.2)$$

After building the similarity matrix, it is necessary to form a new matrix with three elements in each row, where first two elements are indices of hyperboxes and the last element is the similarity value of those two hyperboxes. It is noted that this matrix only includes pairs of hyperboxes with the similarity values larger than or equal to the predetermined threshold. This matrix is then sorted in descending order according to the values of the last column. All operations are executed on the GPUs. The sorted matrix is used for further hyperbox aggregation processes.

Expansion Constraint Checking

Before expanding the selected hyperbox B_i to cover the new input pattern in the incremental learning algorithm, the condition given by Eq. (3.4) is verified for all dimensions. This operation takes much time to complete in the case that the number of dimensions is very high. The similar phenomenon also occurs when checking the maximum hyperbox size of the aggregated hyperbox in the agglomerative learning algorithms. Fortunately, when the data are stored on the GPUs, and Pytorch provides us with the functions $torch.max()$, $torch.min()$, and $.all()$ to execute these operations rapidly on the GPUs.

Overlap Testing between Two Hyperboxes

In the incremental learning algorithm, it is desirable to test overlap in turn between the expanded hyperbox B_i and hyperboxes B_k belonging to other class labels. Four overlap test cases mentioned in subsection 3.2.2 are performed for n-dimensional vectors V_i, W_i, V_k, and W_k. This study is conducted on the very high dimensional datasets, so this operation takes much time during the training process. Therefore, the operations among vectors are built using logical operators to combine four cases aiming to detect whether there is an overlapping region between two hyperboxes, and this task is executed on the GPUs.

If the overlap occurs, the second step would determine which dimension needs to be adjusted with the minimum influence and which case detected the overlap. The second step should be run on the CPU because its operations are performed in sequential for each element in vectors. All operations are detailed in the function **Overlap_Test** in Algorithm 6.3. The function returns the special data structure containing the test case detecting overlap and the index of the dimension needing to be adjusted to resolve the overlap.

Overlap Testing between a Hyperbox and a Set of Other Hyperboxes

In the agglomerative learning algorithm, two hyperboxes are merged when the resulting hyperbox after aggregation does not overlap with hyperboxes belonging

Algorithm 6.3 Overlap test procedure between two Hyperboxes:

Overlap_Test(V_i, W_i, V_k, W_k)

Input:

- V_i, W_i: Minimum and maximum points of the hyperbox B_i
- V_k, W_k: Minimum and maximum points of the hyperbox B_k

Output:

A tuple *dim* stores the index of overlap test cases and the corresponding dimension that overlap happens

Step 1: **Detecting the occurrence of overlapping regions** (executed on GPUs)

$W_iW_k \leftarrow (W_i - W_k) > 0$; $V_iV_k \leftarrow (V_i - V_k) > 0$;
$W_kV_i \leftarrow (W_k - V_i) > 0$; $W_iV_k \leftarrow (W_i - V_k) > 0$;
$c_1 \leftarrow !W_iW_k$ & $!V_iV_k$ & W_iV_k; $c_2 \leftarrow W_iW_k$ & V_iV_k & W_kV_i;
$c_3 \leftarrow W_iW_k$ & $!V_iV_k$; $c_4 \leftarrow !W_iW_k$ & V_iV_k;
$c \leftarrow c_1 \mid c_2 \mid c_3 \mid c_4$

Step 2: **Finding the dimension with minimum influence when doing contraction** (run on CPUs)

$dim \leftarrow \varnothing$
if $c_j = true$, $\forall c_j \in c$ **then**
  $q \leftarrow 1$
  $n \leftarrow |V_i|$ {*The number of dimentions of vector* V_i}
  for $t = 1 \rightarrow n$ **do**
    if $c_1[t] = true$ and $q > W_i[t] - V_k[t]$ **then**
      $q \leftarrow W_i[t] - V_k[t]$; $dim \leftarrow [1, t]$
    else if $c_2[t] = true$ and $q > W_k[t] - V_i[t]$ **then**
      $q \leftarrow W_k[t] - V_i[t]$; $dim \leftarrow [2, t]$
    else if $c_3[t] = true$ **then**
      if $q > W_k[t] - V_i[t]$ and $W_k[t] - V_i[t] < W_i[t] - V_k[t]$ **then**
        $q \leftarrow W_k[t] - V_i[t]$; $dim \leftarrow [31, t]$
      else if $q > W_i[t] - V_k[t]$ **then**
        $q \leftarrow W_i[t] - V_k[t]$; $dim \leftarrow [32, t]$
      end if
    else if $c_4[t] = true$ **then**
      if $q > W_k[t] - V_i[t]$ and $W_k[t] - V_i[t] < W_i[t] - V_k[t]$ **then**
        $q \leftarrow W_k[t] - V_i[t]$; $dim \leftarrow [41, t]$
      else if $q > W_i[t] - V_k[t]$ **then**
        $q \leftarrow W_i[t] - V_k[t]$; $dim \leftarrow [42, t]$
      end if
    end if
  end for
end if
return *dim*

to other classes. Therefore, it is necessary to verify the occurrence of overlapping regions between the newly aggregated hyperbox B_i and any hyperbox in the set of hyperboxes with other labels represented by two matrices $\mathcal{V}'$ and $\mathcal{W}'$. No hyperbox

contraction step is performed in this case, so it is only needed to detect overlapping regions by using GPU computations. Similar to the overlap detection step between two hyperboxes, logical operators And (&), Or (|), and Not (!) are deployed on each element of the matrices for the overlap verification process considering four test cases simultaneously. The function **Overlap_Test_One_Many** describes the computational steps shown in Algorithm 6.4.

Algorithm 6.4 Overlap test between a hyperbox and a group of hyperboxes executed on GPUs: **Overlap_Test_One_Many**$(V_i, W_i, \mathcal{V}', \mathcal{W}')$

Input:

- V_i, W_i: Minimum and maximum points of the hyperbox B_i
- $\mathcal{V}', \mathcal{W}'$: Matrices storing minimum and maximum points of all hyperboxes representing the classes different from the class label of B_i

Output:

A boolean variable $isOver$ shows the overlap occurred ($true$) or not ($false$)

1: $N_1 \leftarrow$ the number of hyperboxes represented by $\mathcal{V}'$ (also $\mathcal{W}'$)
2: $\mathcal{V}_i^{'} \leftarrow$ Replicate vector V_i N_1 times; $\mathcal{W}_i^{'} \leftarrow$ Replicate vector W_i N_1 times
3: $W_iW_k \leftarrow (\mathcal{W}_i^{'} - \mathcal{W}') > 0$; $\quad V_iV_k \leftarrow (\mathcal{V}_i^{'} - \mathcal{V}') > 0$
4: $W_kV_i \leftarrow (\mathcal{W}' - \mathcal{V}_i^{'}) > 0$; $\quad W_iV_k \leftarrow (\mathcal{W}_i^{'} - \mathcal{V}') > 0$
5: $c_1 \leftarrow !W_iW_k \;\&\; !V_iV_k \;\&\; W_iV_k$; $\quad c_2 \leftarrow W_iW_k \;\&\; V_iV_k \;\&\; W_kV_i$;
6: $c_3 \leftarrow W_iW_k \;\&\; !V_iV_k$; $\quad c_4 \leftarrow !W_iW_k \;\&\; V_iV_k$;
7: $C \leftarrow c_1 \mid c_2 \mid c_3 \mid c_4$
8: if $\exists r \in C : r_j = true$, $\forall r_j \in r$, then $isOver \leftarrow true$; otherwise, $isOver \leftarrow false$
9: **return** $isOver$

6.1.3 Experiments for the First Solution

Experiments were conducted on a computer with one Intel Xeon Gold 6150 2.7GHz processor, running on the Linux operating system and containing one NVIDIA Quadro P5000 GPU. Each GPU has 16GB of memory and can perform about 8.9 TFLOPS with single-precision of well-optimized code.

Table 6.1 : Summary of high dimensional datasets for experiments in Chapter 6

Dataset	**#samples**	**#features**	**#training**	**#testing**	**#classes**
PEMS Database	440	138,672	352	88	7
Complex Hydraulic System	2,205	43,680	1764	441	2

The experiments were conducted on two very high dimensional datasets. The

first dataset is PEMS database taken from (Cuturi 2011). The data values ranging from 0 to 1 show the occupancy rate of various car lanes of San Francisco bay region freeways. Each day in this database is a single time series of 963 sensors sampled every 10 minutes during the day. Therefore, there are total 963 x 6 x 24 = 138,672 features for each record. These data are used to classify each observed day as the correct day of the week. As a result, there are seven labels with integer numbers ranging from one to seven.

The second dataset is obtained from (Helwig et al. 2015), which is measurement data from sensors installed in the hydraulic system. These sensors measure different physical quantities such as pressure, motor power, volume flow, temperature, vibration, efficiency factor, cooling efficiency, and cooling power. Combination of data of all sensors forms a dataset consisting of 2205 patterns with 43,680 features per each sample. This dataset contains many different operational statuses of the complex hydraulic system, but only two basic statuses are considered, which are class 1 if the conditions are stable (1449 samples) and class 2 if the static conditions might not yet be reached (756 patterns).

Table 6.2 : Training time in seconds of the *PEMS Database* dataset

Algorithm	**Mode**	$\theta = 0.1$	$\theta = 0.2$	$\theta = 0.3$	$\theta = 0.4$	$\theta = 0.5$	$\theta = 0.6$
Incremental	CPU	32.895	34.891	33.210	33.159	35.095	34.868
	GPU	3.757	3.747	3.744	3.756	3.759	3.753
AGGLO-2	CPU	189.957	191.026	193.133	190.579	191.864	190.441
	GPU	10.203	10.194	10.199	10.213	10.193	10.179
AGGLO-SM	CPU	204.438	202.868	206.134	205.515	203.240	206.418
	GPU	10.658	10.659	10.655	10.661	10.686	10.657

For each dataset, 80% of the data were used for the training process, and the others were deployed to test the constructed GFMMNN. The information of datasets is summarized in Table 6.1. The GFMMNN was trained using different learning algorithms implemented by NumPy library and Pytorch framework on these two training datasets and recorded the training time concerning different values of the maximum hyperbox size θ. Table 6.2 reports the training time in seconds of the

algorithms on the *PEMS Database* dataset. Table 6.3 shows the training time on the *Complex Hydraulic System* dataset.

Table 6.3 : Training time in seconds of the *Complex Hydraulic System* dataset

Algorithm	**Mode**	$\theta = 0.1$	$\theta = 0.2$	$\theta = 0.3$	$\theta = 0.4$	$\theta = 0.5$	$\theta = 0.6$
Incremental	CPU	1,386.456	1,382.728	1,387.856	1,380.778	1,229.650	1,069.713
	GPU	144.264	145.787	147.296	148.389	157.143	154.372
AGGLO-2	CPU	2,659.041	2,698.802	2,654.382	4,855.536	4,695.572	6,908.221
	GPU	81.639	81.636	81.633	161.222	161.521	242.325
AGGLO-SM	CPU	2,056.106	2,034.172	1,887.212	16,875.429	91,634.154	170,623.012
	GPU	64.898	64.935	64.905	555.486	2,983.057	5,674.349

It can be seen that the GPUs contribute to reducing the training time of both datasets on all algorithms from 10 to 35 times compared to the CPU-based serial computations. When the value of θ increases, the number of hyperboxes reduces, but the overlapping regions appear more often because the expansion or aggregation condition concerning θ is easily met. The number of samples in datasets is quite small, so the number of generated hyperboxes is relatively few. Meanwhile, the number of dimensions is very high, and the overlap test procedure is regularly conducted when the hyperbox expansion or merging process occurs frequently. Hence, the computations for the overlap test on very large dimensional vectors or matrices give rise to the increase of the training time. That is the reason why the training time grows when the maximum hyperbox size increases.

Table 6.4 : Testing time in seconds of the *Complex Hydraulic System* dataset

Algorithm	**Mode**	$\theta = 0.1$	$\theta = 0.2$	$\theta = 0.3$	$\theta = 0.4$	$\theta = 0.5$	$\theta = 0.6$
Incremental	CPU	956.687	934.076	942.507	935.316	822.990	679.668
	GPU	28.151	28.141	28.150	28.022	26.215	21.841
AGGLO-2	CPU	939.199	955.927	935.033	870.593	790.102	773.167
	GPU	28.136	28.133	28.133	28.004	26.247	25.266
AGGLO-SM	CPU	957.944	937.624	887.749	841.823	774.469	747.979
	GPU	28.101	28.105	28.101	27.958	26.215	25.389

Table 6.4 shows the testing time using the model trained on the *Complex Hydraulic System* dataset through different maximum hyperbox sizes. The most time-

consuming computations in the testing process consist of the membership computation and finding the maximum membership values. The obtained results indicated that the testing time using GPUs is much faster than that employing CPU. These results confirmed that the proposed method is very effective for operations of GFMMNN on high-dimensional datasets.

The use of GPUs is only suitable for computations of the matrix with enormous size. In the incremental learning algorithm, hyperboxes are built and adjusted gradually based on the input pattern, so the number of hyperboxes is not high. As a result, if the dimensionality of hyperboxes is from tens to hundreds of dimensions, then the usage of GPU computing is regularly ineffective compared to CPU because of interchanging between CPU with the GPU tensor operations and loop instructions being executed on the CPU. These operations cause a lot of overhead and slow down the computations.

6.2 A Solution based on New Hyperbox Selection Rules

6.2.1 Overview

All of the current learning algorithms for the GFMMNN have the same drawback in the selection of expandable or mergeable hyperbox candidates. The creation of a new hyperbox in the online learning algorithms only occurs when all existing hyperboxes with the same class as the input patterns cannot satisfy the expansion condition to cover the new input pattern. The expansion condition is the maximum hyperbox size and the non-overlap of hyperboxes representing different classes if using the IOL-GFMM. Similarly, in the agglomerative learning algorithm, the process of hyperbox merging only terminates if all pairs of hyperbox candidates are examined with regard to the aggregation criteria but the aggregation process cannot be performed. The aggregation conditions include maximum hyperbox size, minimum similarity value, and the non-overlap of hyperboxes with different classes. The consideration of expansion or merging conditions for all hyperbox candidates leads to a waste of time. Therefore, this study provides a lower bound on the membership functions and similarity measures to reduce the considered hyperbox candidates

for the expansion or merging process. This method contributes to decreasing the training time of the learning algorithms.

The contributions in this part can be summarized as follows:

- The lemmas to reduce significantly the considered hyperboxes for both online and batch learning algorithms for the GFMMNN are proposed and proved.
- The effectiveness of the proposed method is assessed on widely used datasets. Experimental results confirmed the strong points of the method in decreasing the training time of the algorithms.

6.2.2 Accelerated Online Learning Algorithms

It is observed that in online learning algorithms of the GFMMNN, a new hyperbox is only created to cover the input pattern if all hyperbox candidates cannot satisfy the conditions to be expanded for accommodating the new input pattern. However, in the current versions of the online learning algorithms, there is no way to reduce the considered hyperbox candidates. This study, therefore, provides a lemma to narrow down the expandable hyperboxes during the training process. This solution is given in Lemma 6.1.

Lemma 6.1. *When finding the candidates of expandable hyperboxes to cover an input pattern $X \in [0,1]^n$, only hyperboxes (h) with the same class as X and having a membership degree ($b_h(X)$) to the new input pattern satisfying: $b_h(X) \geq 1-\theta\cdot\gamma_{max}$ are considered, where $\gamma_{max} = \max_{j=1}^{n}(\gamma_j); \gamma_j > 0$, and θ is the maximum hyperbox size. If the input pattern X is in the form of a hyperbox, its size must be below θ in all n dimensions.*

The proof of Lemma 6.1 can be found in the Appendix D.1. This lemma shows the relationship between the membership function and the maximum hyperbox size parameter if the sensitivity parameter γ is kept fixed. By using this lemma, the number of hyperbox candidates considered for the expansion step based on their membership values can be reduced. The Algorithm 3.1 is modified into Algorithm

6.5 to accelerate the original online learning algorithm of GFMM model. The only change in this algorithm is that the proposed lemma is used to limit the number of hyperboxes considered for each input pattern (as highlighted in red color). The other steps are the same as in the original version.

Similarly, the steps of the IOL-GFMM shown in Algorithm 4.1 can also be changed to Algorithm 6.6 to accelerate the IOL-GFMM procedure. With the Lemma 6.1, the extendable hyperbox candidates to cover the new input pattern may be reduced by conducting the steps highlighted in red color. The remaining operations are the same as the original version of the IOL-GFMM algorithm.

As will be illustrated in the experimental section, these changes to the algorithms and the use of the proved Lemma 6.1 resulted in from 2 to 3.5 times reduction of learning time on average.

Time complexity of the accelerated online learning algorithms. It is easily observed that the accelerated online learning algorithms are different from the original version in the reduction of the hyperbox candidates considered during the training process. Therefore, the time complexity of these algorithms are similar to the original versions, i.e., $\mathcal{O}(N \cdot \overline{\mathcal{K}}_1 \cdot \overline{\mathcal{R}} \cdot n)$, where $\overline{\mathcal{K}}_1$ is the average number of expandable hyperbox candidates considered during the training process, the meaning of other parameters is the same as in the original versions. The accelerated learning algorithms run faster than the original versions because of $\overline{\mathcal{K}}_1 < \overline{\mathcal{K}}$.

6.2.3 Accelerated Agglomerative Learning Algorithms

In the original agglomerative learning algorithms, the hyperbox aggregation process considers all pairs of hyperboxes for which their similarity values are larger than or equal to a given minimum similarity threshold. If this threshold is set too small, then there might be many candidates considered, and so the training process can be long. However, when the minimum similarity condition is met and two hyperboxes could be merged, the newly aggregated hyperbox has to still be checked for the maximum hyperbox size constraint. This chapter will show the dependency

Algorithm 6.5 The accelerated original online learning algorithm

Input:

- θ: The maximum hyperbox size threshold
- γ: The speed of decreasing of the membership function

Output:

A list $\mathcal{H}$ of hyperboxes with minimum-maximum values and classes

Initialize an empty list of hyperboxes: min-max values $\mathcal{V} = \mathcal{W} = \varnothing$, hyperbox classes: $\mathcal{L} = \varnothing$
for each input pattern $X = [X^l, X^u, c_X]$ **do**
  $n \leftarrow$ The number of dimensions of X
  if $\mathcal{V} = \varnothing$ **then**
    $\mathcal{V} \leftarrow X^l$; $\mathcal{W} \leftarrow X^u$; $\mathcal{L} \leftarrow c_X$
  else
    $\mathcal{H}_1 = [\mathcal{V}_1, \mathcal{W}_1, \mathcal{L}_1] \leftarrow$ Find hyperboxes in $\mathcal{H} = [\mathcal{V}, \mathcal{W}, \mathcal{L}]$ representing the same class as c_X
    $\mathcal{M} \leftarrow$ **ComputeMembershipValue**$(X, \mathcal{V}_1, \mathcal{W}_1, \mathcal{L}_1)$
    $\mathcal{H}_s \leftarrow \{h : h \in \mathcal{H}_1, \mathcal{M}(h) \geq 1 - \theta \cdot \gamma_{max}\}$
    $\mathcal{H}_d \leftarrow$ **SortByDescending**$(\mathcal{H}_s, \mathcal{M}(\mathcal{H}_s))$
    Set $\overline{\mathcal{H}_1} \leftarrow \mathcal{H} \setminus \mathcal{H}_1$
    $flag \leftarrow false$
    for each $h = [V_h, W_h, c_h] \in \mathcal{H}_d$ **do**
      if $\mathcal{M}(h) = 1$ **then**
        $flag = true$
        break
      end if
      if $\max(w_{hj}, x_j^u) - \min(v_{hj}, x_j^l) \leq \theta, \forall j \in [1, n]$ **then**
        $W_h^t \leftarrow \max(W_h, X^u)$; $V_h^t \leftarrow \min(V_h, X^l)$
        for each hyperbox $[V_i, W_i, c_i] \in \overline{H_1}$ **do**
          $isOver \leftarrow$ **OverlapTest**(V_h^t, W_h^t, V_i, W_i)
          if $isOver = true$ **then**
            DoContraction(V_h^t, W_h^t, V_i, W_i)
          end if
        end for
        $flag = true$
        break
      end if
    end for
    if $flag = false$ **then**
      $\mathcal{V} \leftarrow \mathcal{V} \cup X^l$; $\mathcal{W} \leftarrow \mathcal{W} \cup X^u$; $\mathcal{L} \leftarrow L \cup c_X$
    end if
  end if
end for
return $\mathcal{H} = [\mathcal{V}, \mathcal{W}, \mathcal{L}]$

of the similarity value with the maximum hyperbox size parameter. Based on this identified relationship, it can be safe to remove immediately the pairs of hyperboxes

Algorithm 6.6 The accelerated IOL-GFMM algorithm

Input:

- θ: The maximum hyperbox size
- γ: The speed of decreasing of the membership function

Output:

A list $\mathcal{H}$ of hyperbox fuzzy sets containing minimum-maximum values and classes

Initialize an empty list of hyperboxes: min-max values $\mathcal{V} = \mathcal{W} = \varnothing$, hyperbox classes: $\mathcal{L} = \varnothing$
for each input pattern $X = [X^l, X^u, c_X]$ **do**
  $n \leftarrow$ The number of dimensions of X
  if $\mathcal{V} = \varnothing$ **then**
    $\mathcal{V} \leftarrow X^l$; $\mathcal{W} \leftarrow X^u$; $\mathcal{L} \leftarrow c_X$
  else
    $\mathcal{H}_1 = [\mathcal{V}_1, \mathcal{W}_1, \mathcal{L}_1] \leftarrow$ Find hyperboxes in $\mathcal{H} = [\mathcal{V}, \mathcal{W}, \mathcal{L}]$ representing the same class as c_X or being unlabeled
    $\mathcal{M} \leftarrow$ **ComputeMembershipValue**$(X, \mathcal{V}_1, \mathcal{W}_1, \mathcal{L}_1)$
    $\mathcal{H}_s \leftarrow \{h : h \in \mathcal{H}_1, \mathcal{M}(h) \geq 1 - \theta \cdot \gamma_{max}\}$
    $\mathcal{H}_d \leftarrow$ **SortByDescending**$(\mathcal{H}_s, \mathcal{M}(\mathcal{H}_s))$
    Set $\overline{\mathcal{H}_1} \leftarrow \mathcal{H} \setminus \mathcal{H}_1$
    $flag \leftarrow false$
    for each $h = [V_h, W_h, c_h] \in \mathcal{H}_d$ **do**
      if $\mathcal{M}(h) = 1$ **then**
        $flag = true$
        break
      end if
      if $\max(w_{hj}, x_j^u) - \min(v_{hj}, x_j^l) \leq \theta, \forall j \in [1, n]$ **then**
        $W_h^t \leftarrow \max(W_h, X^u)$; $V_h^t \leftarrow \min(V_h, X^l)$
        $isOver \leftarrow$ **IsOverlap**$(W_h^t, V_h^t, \overline{\mathcal{H}_1})$
        if $isOver = false$ **then**
          $V_h \leftarrow V_h^t$; $W_h \leftarrow W_h^t$
          $c_h \leftarrow c_X$ if c_h is unlabeled and c_X is labeled
          $flag \leftarrow true$
          Increase the number of samples contained in h
          break
        end if
      end if
    end for
    if $flag = false$ **then**
      $\mathcal{V} \leftarrow \mathcal{V} \cup X^l$; $\mathcal{W} \leftarrow \mathcal{W} \cup X^u$; $\mathcal{L} \leftarrow L \cup c_X$
    end if
  end if
end for
return $\mathcal{H} = [\mathcal{V}, \mathcal{W}, \mathcal{L}]$

for which the hyperbox aggregated from these candidates cannot, with absolute certainty, satisfy the maximum hyperbox size condition. The details of the proposed method are described in Lemma 6.2.

Lemma 6.2. *Regardless of the similarity measure used, the hyperbox aggregation process only considers pairs of hyperbox candidates that their similarity values satisfy the following condition:* $s(B_i, B_k) \geq \max(\sigma, 1-\theta\cdot\gamma_{max})$*, where* $\gamma_{max} = \max_{j=1}^{n}(\gamma_j); \gamma_j > 0$*,* σ *is the minimum similarity threshold,* n *is the number of dimensions, and* θ *is the maximum hyperbox size parameter. It is noted that the size of all hyperbox candidates must be below* θ*.*

The proof of Lemma 6.2 can be found in the Appendix D.2. By using Lemma 6.2, the steps of the AGGLO-SM algorithm in Algorithm 3.2 can be changed to Algorithm 6.7 (the difference between two algorithms is highlighted in red color), and modify the AGGLO-2 algorithm as shown in Algorithm 6.8 (the red color part shows the difference between two algorithms). The only change in these accelerated algorithms compared to their original versions is the limitation of pairs of candidates considered during the learning process by a stricter lower bound. The remaining operations are kept unchanged as described in the original algorithms.

As will be shown in the experimental section, these changes to the agglomerative learning algorithms and the usage of the proposed Lemma 6.2 led to the acceleration of seven times in the training time of the AGGLO-SM algorithm, while the training time of the AGGLO-2 algorithm is reduced from 25 to 37 times on average depending on the similarity measure and dataset deployed.

Time complexity of the accelerated agglomerative learning algorithms. It can be seen that the accelerated version of the AGGLO-SM algorithm is only reduced by the number of pairs of hyperbox candidates considered in the aggregation process. Therefore, the complexity of the accelerated AGGLO-SM algorithm is $\mathcal{O}(N^2 \cdot (\overline{\mathcal{Z}}_1 \cdot \overline{\mathcal{Y}} \cdot n + \overline{\mathcal{Y}}^2))$, where $\overline{\mathcal{Z}}_1$ is the average number of pairs of hyperbox candidates, the meaning of the other notations is the same as in the description of the original AGGLO-SM algorithm shown in subsection 3.2.3. Similarly, the complexity

Algorithm 6.7 The accelerated AGGLO-SM algorithm

Input:

- $\mathbf{X} = [\mathbf{X}^l, \mathbf{X}^u]$: A list of training features
- $\mathcal{C}$: A vector of pattern classes
- θ: The maximum hyperbox size threshold
- γ: The speed of decreasing of the membership function

Output:

A list $\mathcal{H}$ of hyperboxes with minimum-maximum values and classes

Initialize a list of hyperboxes: min-max values $\mathcal{V} = \mathbf{X}^l, \mathcal{W} = \mathbf{X}^u$, hyperbox classes: $\mathcal{L} = \mathcal{C}$
$loop \leftarrow true$; $n \leftarrow$ the number of features of $\mathbf{X}$
$S \leftarrow$ **ComputeSimilarityValPairWithinEachClass**$(\mathcal{V}, \mathcal{W}, \mathcal{L})$
while $loop = true$ **do**
    $loop \leftarrow false$
    $S \leftarrow S \setminus \{s \in S | s < \max(\sigma, 1 - \theta \cdot \gamma_{max})\}$
    $I, K, S \leftarrow$ **SortByDescending**$(S, \mathcal{V}, \mathcal{W}, \mathcal{L})$
    for each $[i, k, s] \in [I, K, S]$ **do**
        if $\max(w_{ij}, w_{kj}) - \min(v_{ij}, v_{kj}) \leq \theta, \forall j \in [1, n]$ **then**
            $W_t \leftarrow \max(W_i, W_k)$; $V_t \leftarrow \min(V_i, V_k)$
            $\overline{H_1} \leftarrow$ A list of hyperboxes with classes different from $c_i \in \mathcal{L}$
            $isOver \leftarrow$ **IsOverlap**$(V_t, W_t, \overline{\mathcal{H}_1})$
            if $isOver = false$ **then**
                $loop \leftarrow true$
                $V_i \leftarrow V_t$; $W_i \leftarrow W_t$
                $\mathcal{V} \leftarrow \mathcal{V} \setminus V_k$; $\mathcal{W} \leftarrow \mathcal{W} \setminus W_k$; $\mathcal{L} \leftarrow \mathcal{L} \setminus L_k$
                $S \leftarrow$ **UpdateSimilarityMatrix**$(\mathcal{V}, \mathcal{W}, \mathcal{L})$
                break
            end if
        end if
    end for
end while
return $\mathcal{H} = [\mathcal{V}, \mathcal{W}, \mathcal{L}]$

of the accelerated AGGLO-2 algorithm is $\mathcal{O}(N \cdot \overline{\mathcal{Z}}_1 \cdot \overline{\mathcal{R}} \cdot n)$ in the worst-case, where $\overline{\mathcal{Z}}_1$ is also the average number of pairs of hyperbox candidates considered during the training process of the accelerated AGGLO-2 algorithm. The other parameters have the same meaning as shown in subsection 3.2.4 for the original AGGLO-2 algorithm. The accelerated agglomerative algorithms run faster than the original versions due to $\overline{\mathcal{Z}}_1 < \overline{\mathcal{Z}}$.

For the time complexity of the AGGLO-2 algorithm, $\overline{\mathcal{Z}}, N, n$, and $\overline{\mathcal{R}}$ are equally

Algorithm 6.8 The accelerated AGGLO-2 algorithm

Input:

- $\mathbf{X} = [\mathbf{X}^l, \mathbf{X}^u]$: A list of training features
- $\mathcal{C}$: A vector of pattern classes
- θ: The maximum hyperbox size threshold
- γ: The speed of decreasing of the membership function

Output:

A list $\mathcal{H}$ of hyperboxes with minimum-maximum values and classes

Initialize a list of hyperboxes: min-max values $\mathcal{V} = \mathbf{X}^l, \mathcal{W} = \mathbf{X}^u$, hyperbox classes: $\mathcal{L} = \mathcal{C}$
$loop \leftarrow true$; $n \leftarrow$ the number of features of $\mathbf{X}$
while $loop = true$ **do**
    $loop \leftarrow false$; $i \leftarrow 1$
    while $i \leq |\mathcal{L}|$ **do**
        $\mathcal{H}_1 = [\mathcal{V}_1, \mathcal{W}_1, \mathcal{L}_1] \leftarrow$ Find hyperboxes in $[\mathcal{V}, \mathcal{W}, \mathcal{L}]$ representing the same class as $c_i \in \mathcal{L}$
        $S \leftarrow$ **ComputeSimilarityValPair**$(V_i, W_i, \mathcal{H}_1)$
        $S \leftarrow S \setminus \{s \in S | s < \max(\sigma, 1 - \theta \cdot \gamma_{max})\}$
        $K, S \leftarrow$ **SortByDescending**$(S, \mathcal{V}_1, \mathcal{W}_1, \mathcal{L}_1)$
        for each $[k, s] \in [K, S]$ **do**
            if $\max(w_{ij}, w_{kj}) - \min(v_{ij}, v_{kj}) \leq \theta, \forall j \in [1, n]$ **then**
                $W_t \leftarrow \max(W_i, W_k)$; $V_t \leftarrow \min(V_i, V_k)$
                $\overline{H_1} \leftarrow$ A list of hyperboxes with classes different from l_i
                $isOver \leftarrow$ **IsOverlap**$(V_t, W_t, \overline{\mathcal{H}_1})$
                if $isOver = false$ **then**
                    $loop \leftarrow true$
                    $V_i \leftarrow V_t$; $W_i \leftarrow W_t$
                    $\mathcal{V} \leftarrow \mathcal{V} \setminus V_k$; $\mathcal{W} \leftarrow \mathcal{W} \setminus W_k$; $\mathcal{L} \leftarrow \mathcal{L} \setminus L_k$
                    if $i > k$ **then**
                        $i \leftarrow i - 1$
                    end if
                    break
                end if
            end if
        end for
        $i \leftarrow i + 1$
    end while
end while
return $\mathcal{H} = [\mathcal{V}, \mathcal{W}, \mathcal{L}]$

important factors affecting the algorithm time complexity. However, for the complexity of the AGGLO-SM algorithm, the role of $\overline{\mathcal{Z}}$ is much less important than those of N^2 and $\overline{\mathcal{Y}}^2$, especially in the large-sized datasets. Therefore, the impact

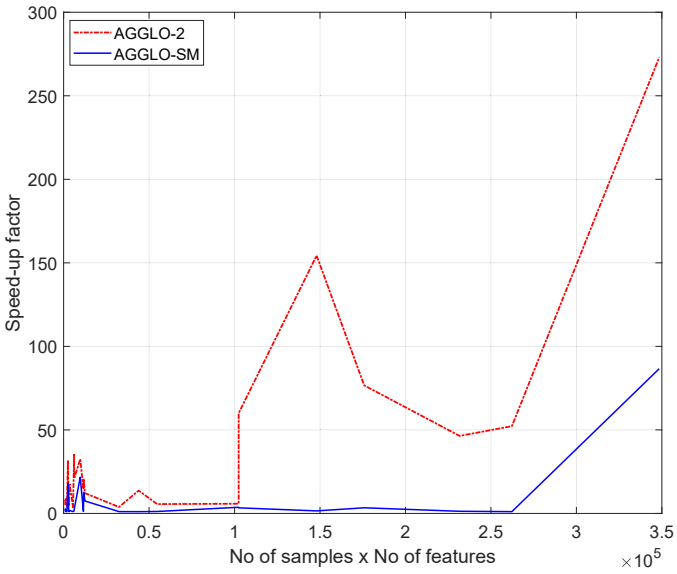


Figure 6.1 : The speedup factor of the AGGLO-2 and AGGLO-SM algorithms according to the number of samples and the number of features over 24 experimental datasets (using the "longest distance" similarity measure).

of the reduction of $\overline{\mathcal{Z}}$ on the AGGLO-2 algorithm is more significant compared to the AGGLO-SM algorithm. This fact is confirmed by the experimental results in section 6.2.4. As an illustrative example, Figure 6.1 shows the speed-up factors of the AGGLO-2 and AGGLO-SM using the proposed method with the "longest distance" similarity value in correlation to the numbers of samples and features for the 24 experimental datasets. It can be seen that the acceleration of the AGGLO-2 algorithm is much higher than that of the AGGLO-SM when using the proposed approach. Apart from the numbers of samples and features, the input data distribution and the complexity of the classification problem have a significant effect on the speed-up factor of the proposed approach. For instance, in Table 6.6, two datasets *plant species leaves texture* and *plant species leaves margin* have the same numbers of samples, features, and classes, but their speed-up factors are significantly different.

6.2.4 Experiments for the Second Solution

The experimental datasets used in this subsection and parameter settings for algorithms can be found in Section D.3 in Appendix D.

Experimental Results for Online Learning Algorithms

Table 6.5 : Speed-up factor and the number of hyperbox candidates considered during the training process of online learning algorithms

Dataset	**IOL-GFMM**			**Onln-GFMM**		
	Speed-up factor	**Number of hyperbox candidates**		**Speed-up factor**	**Number of hyperbox candidates**	
		w/o lemma	**w/ lemma**		**w/o lemma**	**w/ lemma**
blance scale	5.7227	20880	0	5.3978	20880	0
banknote authentication	1.3016	5192.5	914	1.0563	5129	904
blood transfusion	1.2487	2154.5	392	1.0568	1542	320
breast cancer wisconsin	3.8746	14655	0	3.6556	14655	0
breast cancer coimbra	2.3571	768	2	1.9804	768	2
climate model crashes	6.2478	30634	0	6.0559	30634	0
connectionist bench sonar	2.8302	2664.5	0	2.5664	2664.5	0
glass	1.3178	567	49.5	1.1009	567	49.5
haberman	1.3418	960.5	143.5	1.122	913.5	141.5
heart	3.5575	4479.5	1	3.1069	4479.5	1
ionosphere	2.7678	5705	46	1.6163	5705	46
movement libras	1.5575	676	24.5	1.0989	676	24.5
optical digit	4.5451	393452.5	0	3.0049	393452.5	0
page blocks	1.3019	21124.5	2928.5	1.0458	20825.5	2909.5
pendigits	4.2866	892103	3421.5	1.0656	892103	3421.5
pima diabetes	4.9452	26995.5	197.5	1.8977	26995.5	197.5
plant species leaves margin	2.0362	2800	0	1.3029	2800	0
plant species leaves texture	7.2026	309308	16	7.417	309308	16
ringnorm	12.7486	2986751.5	4108.5	3.0377	2986751.5	4108.5
seeds	1.7213	959.5	42.5	1.2	959.5	42.5
image segmentation	1.8082	26153	1052.5	1.0332	26153	1052.5
spambase	3.3182	329619.5	2815.5	1.2982	329619.5	2815.5
spectf heart	3.8481	5929.5	0	3.7564	5929.5	0
landsat satellite	2.76	349032.5	12759	1.0733	349029	12752
Average ratio with over without lemma	**3.527**	**0.038**		**2.373**	**0.040**	

Table 6.5 shows the average speed-up factor and the number of hyperbox candidates considered during the learning process over 10 iterations (5 times 2-fold cross-validation) for each dataset. The speed-up value is computed by dividing the

training time of the algorithm without using the lemma by the training time of the algorithm using the lemma to accelerate the learning process. The training time of online learning algorithms are shown in Table D.2 in the Appendix D.4.

It can be easily observed that the online learning algorithms using the proposed lemma are much faster than ones without deploying the lemma. These figures can be explained based on the number of hyperboxes considered during the training process. It can be seen that the use of lemma has significantly reduced the unsuitable hyperbox candidates that the original versions have to verify. In several datasets, the number of hyperbox candidates is zero in the case of using the proposed lemma because all existing hyperboxes cannot be extended to cover the new input patterns. It means that the resulting models only contain hyperboxes with one data point. In this case, the speed-up of learning process using the proposed lemma is obvious.

In general, the proposed method contributes to the acceleration of the IOL-GFMM algorithm more significantly than the Onln-GFMM. This is because the training time of the IOL-GFMM is usually faster than the Onln-GFMM algorithm with the small value of θ ($\theta = 0.1$ in this work) (Khuat et al. 2020). Therefore, when the number of candidates considered in the IOL-GFMM reduces, the overlap test operation between the extended hyperbox and the existing hyperboxes is conducted much faster. Meanwhile, the original online learning algorithm needs to check overlap and find the dimension to conduct the contraction for each pair of hyperboxes. These operations occupy most of the computational expense of the Onln-GFMM algorithm, so the obtained speed-up of the Onln-GFMM is smaller than that of the IOL-GFMM algorithm.

Experimental Results for Agglomerative Learning Algorithms

This part reports the experimental results of agglomerative learning algorithms with and without using the proposed lemma. Table 6.6 presents the speed-up factor of the AGGLO-2 algorithm with the use of the proposed lemma compared to one without using the lemma for four similarity measures. These values are calculated from the average training time shown in Table D.3 in the Appendix D.4. Table 6.7

describes the number of considered hyperbox candidates of the AGGLO-2 algorithm for each similarity measure. Table 6.8 shows the speed-up factors of the AGGLO-SM algorithm on the experimental datasets, which are computed from their training time in Table D.4 in the Appendix D.4. The number of hyperbox candidates considered during the training process of the AGGLO-SM algorithm with four similarity measures is presented in Table 6.9.

Table 6.6 : Speed-up factor of the AGGLO-2 algorithm

Dataset	**Longest distance**	**Shortest distance**	**Mid-max distance**	**Mid-min distance**
blance scale	31.7626	31.9529	21.156	21.2067
banknote authentication	3.5408	2.6579	2.6302	2.7469
blood transfusion	4.3349	2.7389	2.7078	3.2302
breast cancer wisconsin	21.3396	21.1669	14.7005	14.5726
breast cancer coimbra	8.9	9.0253	6.0635	6.0397
climate model crashes	32.5457	32.2009	20.9304	20.89
connectionist bench sonar	12.1624	12.0424	7.829	7.699
glass	4.1902	3.6753	2.9623	3.1639
haberman	5.0756	3.5084	3.1492	3.7729
heart	17.5549	17.5233	11.4509	11.3709
ionosphere	14.5455	12.7882	8.9468	9.6922
movement libras	3.84	3.4975	2.8035	2.9697
optical digit	272.9711	259.0673	176.7774	183.3464
page blocks	5.5846	3.4114	3.5063	4.1349
pendigits	76.6387	59.2468	51.1295	57.0118
pima diabetes	35.3624	28.1336	21.2801	23.7888
plant species leaves margin	5.8103	5.725	4.1252	4.1257
plant species leaves texture	59.9784	60.189	37.6131	37.3868
ringnorm	154.348	110.4401	89.7151	100.4713
seeds	6.2366	5.2175	4.1833	4.6059
image segmentation	13.6811	9.7371	8.6332	10.1224
spambase	52.2312	29.2803	28.7548	34.9655
spectf heart	20.6977	20.6945	13.2112	13.239
landsat satellite	46.4068	16.1142	20.3232	32.471
Average	**37.906**	**31.668**	**23.524**	**25.543**

In general, the use of the proposed lemma makes the agglomerative learning algorithm much faster because the number of candidates for the hyperbox aggregation process is considerably reduced. Among two versions of the batch learning algo-

Table 6.7 : The number of hyperboxes considered during the training process of the AGGLO-2 algorithm

Dataset	**Longest distance**		**Shortest distance**		**Mid-max distance**		**Mid-min distance**	
	w/o lemma	**w/ lemma**	**w/o lemma**	**w/ lemma**	**w/o lemma**	**w/ lemma**	**w/o lemma**	**w/ lemma**
blance scale	41760	0	41760	0	41760	0	41760	0
banknote authentication	22340	552	22105.5	2896	22300	1675.5	22285.5	1046
blood transfusion	17806.5	447	14332	1952.5	14470.5	1256.5	17637.5	836.5
breast cancer wisconsin	119974	106.5	119974	106.5	119974	106.5	119974	106.5
breast cancer coimbra	3038.5	2	3038.5	2	3038.5	2	3038.5	2
climate model crashes	61268	0	61268	0	61268	0	61268	0
connectionist bench sonar	5329	0	5329	0	5329	0	5329	0
glass	3004	36.5	2954.5	120.5	2954	107.5	3005	60.5
haberman	5912.5	93.5	4913.5	398.5	4905.5	275.5	5746.5	142.5
heart	13311.5	1	13311.5	1	13311.5	1	13311.5	1
ionosphere	33598	27	26748	167.5	26747.5	134.5	33599	36
movement libras	3897	34.5	3127.5	55	3127.5	54	3897	36
optical digit	786905	0	786905	0	786905	0	786905	0
page blocks	235607	2522	164723.5	15074.5	194023	8211.5	231764.5	5870
pendigits	5080436	1276	5077333	19843	5077333	11493.5	5080439	2128.5
pima diabetes	128175.5	61.5	100825.5	576.5	100825.5	372.5	128175.5	84
plant species leaves margin	5600	0	5600	0	5600	0	5600	0
plant species leaves texture	1232425	12.5	1232425	32.5	1232425	32.5	1232425	12.5
ringnorm	11629867	250	11631487	29762.5	11631487	14020.5	11629867	311
seeds	3333	28.5	3289.5	120	3289.5	95	3333	34.5
image segmentation	181993	570	157079.5	4098.5	157231	2709	181772	1173
spambase	3707848	1150	3340482	51622.5	3345250	21569.5	3707269	4208.5
spectf heart	11859	0	11859	0	11859	0	11859	0
landsat satellite	2021658	1187.5	1956507	81882.5	1956399	39069	2438703	11493
Average ratio w/ over w/o lemma	**0.0047**		**0.0266**		**0.0167**		**0.008**	

rithms, the influence of the proposed lemma on the performance of the AGGLO-2 is more significant compared to the AGGLO-SM. It is due to the fact that the number of hyperbox candidates for the aggregation process in the original AGGLO-2 is higher than the AGGLO-SM algorithm. For several datasets in the AGGLO-2 algo-

rithm such as *optical digit*, *pendigits*, *rignorm*, and *spambase*, the use of the proposed lemma accelerates the training process from 50 to nearly 280 times compared to the original version. Although the AGGLO-SM algorithm cannot obtain such speed-up, the training time also reduces considerably when using the proposed lemma. These results confirmed that the proposed method is efficient for all learning algorithms of the GFMMNN.

Table 6.8 : Speed-up factor of the AGGLO-SM algorithm

Dataset	**Longest distance**	**Shortest distance**	**Mid-max distance**	**Mid-min distance**
blance scale	17.9903	17.9709	17.8937	17.8841
banknote authentication	1.1323	1.1549	1.1503	1.1634
blood transfusion	1.1521	1.1381	1.1133	1.1198
breast cancer wisconsin	3.0204	3.0734	2.9323	3.0311
breast cancer coimbra	2.6782	2.6897	2.6782	2.6782
climate model crashes	21.6677	21.4537	21.7	21.4056
connectionist bench sonar	7.4505	7.4286	7.4505	7.4615
glass	1.0887	1.0678	1.0637	1.0775
haberman	1.1469	1.1208	1.1308	1.1521
heart	1.7236	1.7073	1.6763	1.6911
ionosphere	1.2263	1.3166	1.2595	1.2782
movement libras	1.0842	1.0981	1.0841	1.0845
optical digit	86.5943	83.8357	75.9623	83.9551
page blocks	1.1686	1.1677	1.1389	1.1638
pendigits	3.3746	1.6839	1.591	3.0278
pima diabetes	1.5718	1.4924	1.519	1.5698
plant species leaves margin	3.6856	3.7029	3.6874	3.6836
plant species leaves texture	3.3463	3.3465	3.3581	3.3449
ringnorm	1.5222	1.4626	1.4874	1.5244
seeds	1.1595	1.1351	1.1399	1.1459
image segmentation	1.0589	1.0521	1.0545	1.0577
spambase	1.0672	1.0647	1.0658	1.0623
spectf heart	12.6057	12.6098	12.5547	12.5587
landsat satellite	1.2885	1.0914	1.0982	1.2314
Average	**7.492**	**7.286**	**6.950**	**7.348**

In the AGGLO-2 algorithm, among four similarity measures, the proposed lemma has the most impact on the longest distance measure and the least influence on the mid-max distance-based similarity measure. This is because the number of

Table 6.9 : Number of hyperbox candidates considered during the training process of the AGGLO-SM algorithm

Dataset	**Longest distance**		**Shortest distance**		**Mid-max distance**		**Mid-min distance**	
	w/o lemma	**w/ lemma**	**w/o lemma**	**w/ lemma**	**w/o lemma**	**w/ lemma**	**w/o lemma**	**w/ lemma**
blance scale	18032	0	18032	0	18032	0	18032	0
banknote authentication	5003	552.5	21709.5	18143.5	10925	6988.5	6546.5	2390.5
blood transfusion	2426	572	3940	2482.5	3078.5	1375	2754	947.5
breast cancer wisconsin	12731.5	106.5	12731.5	106.5	12731.5	106.5	12731.5	106.5
breast cancer coimbra	767.5	2	767.5	2	767.5	2	767.5	2
climate model crashes	30631	0	30631	0	30631	0	30631	0
connectionist bench sonar	2652.5	0	2652.5	0	2652.5	0	2652.5	0
glass	543.5	37	679	198.5	654.5	161.5	550	56.5
haberman	1018.5	95.5	1276	418.5	1200.5	285	1003.5	119
heart	419	1	419	1	419	1	419	1
ionosphere	4385.5	26.5	4522	219	4480.5	171.5	4403	47
movement libras	690	34.5	727.5	77	720.5	69.5	697	42
optical digit	268395.5	0	268395.5	0	268395.5	0	268395.5	0
page blocks	27414	3158.5	203481	188327.5	76313.5	59269.5	44376.5	20037
pendigits	801368	1266	825055.5	32153	809903	17248.5	801416.5	2085
pima diabetes	27268.5	60	30319.5	3535.5	29136.5	2133.5	27328.5	133
plant species leaves margin	2792.5	0	2792.5	0	2792.5	0	2792.5	0
plant species leaves texture	300635.5	12.5	301010	45.5	301010	45.5	300636.5	12.5
ringnorm	3000191	241	3952685	965343.5	3451147	456658	3002038	2119.5
seeds	946	29.5	1152	268.5	1080.5	183.5	948.5	33
image segmentation	25541	571.5	68281.5	44145.5	49774	25671	28740	4101
spambase	328946.5	1158	665797.5	356241.5	513685	198932	342728	18653.5
spectf heart	5916	0	5916	0	5916	0	5916	0
landsat satellite	384186.5	1179.5	1592177	1258323	1070981	745576	446447.5	66231
Average ratio w/ over w/o lemma	**0.0315**		**0.2410**		**0.1871**		**0.0773**	

hyperbox candidates considered during the original training process using the longest distance-based measure is highest, but after using the proposed method, the number of considered candidates is smallest. Although the number of hyperbox candidates considered in the training process with regard to the middle distance-based similarity

measures using the proposed lemma is lower than that of the shortest distance-based measure, its speed-up factor on average is still lower compared to the value of the shortest-based measure. This is due to the fact that the middle-based measures are asymmetrical values, so for each pair of candidates B_i and B_k, the training step has to spend time computing both similarity values s_{ik} and s_{ki}. The repetition of similarity computation increases the training time and reduces the speed-up factor though there is a reduction in the number of considered candidates.

Similarly, the speed-up factor of the training process using the proposed lemma with regard to the longest distance-based similarity measure in the AGGLO-SM is highest because the number of hyperbox candidates after using the proposed lemma is much lower than those of other similarity measures. Hence, its training time is fastest on average. The influence of the proposed lemma on the AGGLO-SM using the mid-max distance measure is still smallest among four similarity measures since its training process has to calculate the similarity values for each pair of hyperboxes twice and the number of hyperbox candidates is relatively high after using the proposed lemma. In contrast to the outcomes of the AGGLO-2, the impact of the proposed lemma on the training time of the AGGLO-SM algorithm using the mid-min distance-based measure is ranked in the second place as the number of candidates is much smaller than those of the shortest and mid-max distance-based measures.

6.3 Summary

This chapter addressed the thesis Objective 3 presented in Section 1.2. It proposed two solutions to accelerate learning algorithms for the GFMMNN. The first solution reformulated and represented the GFMM algorithms in a matrix format to facilitate the parallel execution on the GPUs. Empirical results indicated the efficiency of the use of GPU computations to accelerate the training and testing processes of the GFMMNN for very high dimensional data. The GPU-based computations implemented by Pytorch framework contributed to reducing the training and testing time of the GFMMNN from 10 to 35 times in comparison to CPU-based

serial execution. These findings advocate a new method to train and expand the GFMMNN for very high dimensional datasets with minimal software engineering effort.

The second solution for accelerating learning algorithms presented and proved stricter lower bounds for the online and agglomerative learning algorithms of the GFMMNN in selecting the hyperbox candidates. The proposed method reduces significantly the unsuitable hyperbox candidates considered during the learning process, especially in the AGGLO-2 algorithm. Therefore, the training operations are accelerated when applying the proposed solutions. Experimental results on many datasets confirmed the effectiveness of the proposed approach. In particular, the acceleration factors of the online learning algorithms are from two to three on average, while the training time of the AGGLO-SM algorithm is reduced about seven times on average. Especially, the speed-up factor in the AGGLO-2 algorithm using the proposed lemma can achieve from 30 to 250 on several datasets when the number of unsuitable hyperbox candidates is considerably reduced.

Chapter 7

Learning at Different Granularity Levels using Hyperbox Representations

This chapter presents a method to construct classifiers from multi-resolution hierarchical granular representations (MRHGRC) using hyperbox fuzzy sets. The proposed approach will build a series of granular inferences hierarchically through many levels of abstraction. An important property of the proposed classifier is that it can maintain a high accuracy at a high degree of abstraction compared to other fuzzy min-max classifiers based on reusing the knowledge learned from lower levels of abstraction. Furthermore, the proposed method can reduce the data size significantly as well as handle the uncertainty and incompleteness associated with data in real-world applications. The construction process of the classifier consists of two phases. The first phase is to formulate the model at the greatest level of granularity, while the later stage aims to reduce the complexity of the constructed model and deduce it from data at higher abstraction levels. The main content of this chapter is taken from the following paper (Khuat et al. 2021a):

1. **Thanh Tung Khuat**, Fang Chen, and Bogdan Gabrys, "An effective multi-resolution hierarchical granular representation based classifier using general fuzzy min-max neural network," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 2, pp. 427- 441, 2021.

7.1 Introduction

In a recent study, Jordan (2019) claimed that intelligent learning algorithms need the ability to emulate the kinds of intelligence of human brains and minds including the ability to form abstractions, to give semantic interpretation to thoughts and percepts, and to reason. This study proposes an effective classifier based on multiple

levels of abstraction of hyperboxes. Based on this design, the classifier is able to reason through different hierarchical granularity levels.

Hierarchical problem solving, where the problems are analyzed in a variety of granularity degrees, is a typical characteristic of the human brain (Wang et al. 2017). Inspired by this ability, granular computing was introduced. One of the critical features of granular computing is to model the data as high-level abstract structures and to tackle problems based on these representations similar to structured human thinking (Morente-Molinera et al. 2017). Information granules (IGs) (Zadeh 1997) are underlying constructs of the granular computing. They are abstract entities describing important properties of numeric data and formulating knowledge pieces from data at a higher abstraction level. They play a critical role in the concise description and abstraction of numeric data (Pedrycz et al. 2015). Information granules have also contributed to quantifying the limited numeric precision in data (Pedrycz 2014).

Utilizing information granules is one of the problem-solving methods based on decomposing a big problem into sub-tasks which can be solved individually. In the world of big data, one regularly departs from specific data entities and discover general rules from data via encapsulation and abstraction. The use of information granules is meaningful when tackling the five Vs of big data (Xu et al. 2018), i.e., volume, variety, velocity, veracity, and value. Granulation process gathering similar data together contributes to reducing the data size, and so the volume issue is addressed. The information from many heterogeneous sources can be granulated into various granular constructs, and then several measures and rules for uniform representation are proposed to fuse base information granules as shown in Xu and Yu (2017). Hence, the data variety is addressed. Several studies constructed the evolving information granules to adapt to the changes in the streams of data as in Al-Hmouz et al. (2018). The variations of information granules in a high-speed data stream assist in tackling the velocity problem of big data. The process of forming information granules is often associated with the removal of outliers and dealing with incomplete data (Xu et al. 2018); thus the veracity of data is guaranteed. Finally,

the multi-resolution hierarchical architecture of various granular levels can disregard some irrelevant features but highlighting facets of interest (Chen and Zhang 2014). In this way, the granular representation may help with cognitive demands and capabilities of different users.

A multi-dimensional hyperbox fuzzy set is a fundamental conceptual vehicle to represent information granules. Each fuzzy min-max hyperbox is determined by the minimum and maximum points and a fuzzy membership function. A classifier can be built from the hyperbox fuzzy sets along with an appropriate training algorithm. A rule set can be directly extracted from hyperbox fuzzy sets or by using it in combination with other methods such as decision trees (Khuat et al. 2021b) to account for the predictive results. However, a limitation of hyperbox-based classifiers is that their accuracy at the low level of granularity (corresponding to large-sized hyperboxes) decreases. In contrast, classifiers at the high granularity level are more accurate, but the building process of classifiers at this level is time-consuming, and it is difficult to extract the rule set interpretable for predictive outcomes because of the high complexity of resulting models. Hence, it is desired to construct a simple classifier with high accuracy. In addition, it is expected to observe the change in the predictive results at different data abstraction levels. This chapter introduces a method of constructing a high-precision classifier at the high data abstraction level based on the knowledge learned from lower abstraction levels. On the basis of classification errors on the validation set, the change in the accuracy of the constructed classifier on unseen data can be predicted, and an abstraction level satisfying both acceptable accuracy and simple architecture on the resulting classifier can be selected. Furthermore, the proposed method is likely to expand for large-sized datasets due to the capability of parallel execution during the construction process of core hyperboxes at the highest level of granularity. In the proposed method, the algorithm starts with a relatively small value of maximum hyperbox size (θ) to produce base hyperbox fuzzy sets, and then this threshold is increased in succeeding levels of abstraction whose inputs are the hyperbox fuzzy sets formed in the previous step. By using many hierarchical resolutions of granularity, the in-

formation captured in earlier steps is transferred to the classifier at the next level. Therefore, the classification accuracy is still maintained at an acceptable value when the resolution of training data is low.

Data generated from complex real-world applications frequently change over time, so the machine learning models used to predict behaviors of such systems need the efficient online learning capability. Many studies considered the online learning capability when building machine learning models such as (Simpson 1992; Gabrys and Bargiela 2000; de Jesús Rubio 2009; Zhang and Han 2017; de Jesús Rubio 2017; Cheng et al. 2018), and (Rubio et al. 2019). Fuzzy min-max neural networks proposed by Simpson (1992) and many of its improved variants presented in Chapter 2 only work on the input data in the form of points. In practice, due to the uncertainty and some abnormal behaviors in the systems, the input data include not only crisp points but also intervals. The GFMMNN (Gabrys and Bargiela 2000) was proposed to handle both fuzzy and crisp input samples. By using hyperbox fuzzy sets for the input layer, this model can accept the input patterns in the granular form and process data at a high-level abstract structure. As a result, the proposed method in this chapter used a similar mechanism as in the GFMMNN to build a series of classifiers through different resolutions, where the small-sized resulting hyperboxes generated in the previous step become the input to be handled at a higher level of abstraction (corresponding to a higher value of the allowable hyperbox size). Going through different resolution degrees, the valuable information in the input data is fuzzified and reduced in size, but the proposed method helps to preserve the amount of knowledge contained in the original datasets. This capability is illustrated via the slow decline in the classification accuracy. In some cases, the predictive accuracy increases at higher levels of abstraction because the noise existing in the detailed levels is eliminated.

Building on the principles of developing GFMMNN with good generalization performance discussed in Gabrys (2004), this study employs different hierarchical representations of granular data with various hyperbox sizes to select a compact classifier with acceptable accuracy at a high level of abstraction. Hierarchical granular repre-

sentations using consecutive maximum hyperbox sizes form a set of multi-resolution hyperbox-based models, which can be used to balance the trade-off between efficiency and simplicity of the classifiers. A model with high resolution corresponds to the use of a small value of maximum hyperbox size, and vice versa. A choice of suitable resolution level results in better predictive accuracy of the generated model. The main contributions in this chapter can be summarized as follows:

- A new data classification model is proposed based on the multi-resolution of granular data representations in combination with the online learning ability of the general fuzzy min-max neural network.
- The proposed method is capable of reusing the learned knowledge from the highest granularity level to construct new classifiers at higher abstraction levels with the low trade-off between the simplification and accuracy.
- The efficiency and running time of the general fuzzy min-max classifier are significantly enhanced in the proposed algorithm.
- The proposed classifier can perform on large-sized datasets because of the parallel execution ability.
- Comprehensive experiments are conducted on synthetic and real datasets to prove the effectiveness of the proposed method compared to other approaches and baselines.

7.2 Multi-Resolution Hierarchical Granular Representation based Classifier

7.2.1 Overview

The learning process of the proposed method consists of two phases. The first phase is to rapidly construct small-sized hyperboxes from similar input data points. This phase is performed in parallel on training data segments. The data in each fragment can be organized according to two modes. The first way is called heterogeneous mode, which uses the data order read from the input file. The second

mode is homogeneous, in which the data are sorted according to groups; each group contains data from the same class. The main purpose of the second phase is to decrease the complexity of the model by reconstructing phase-1 hyperboxes with a higher abstraction level.

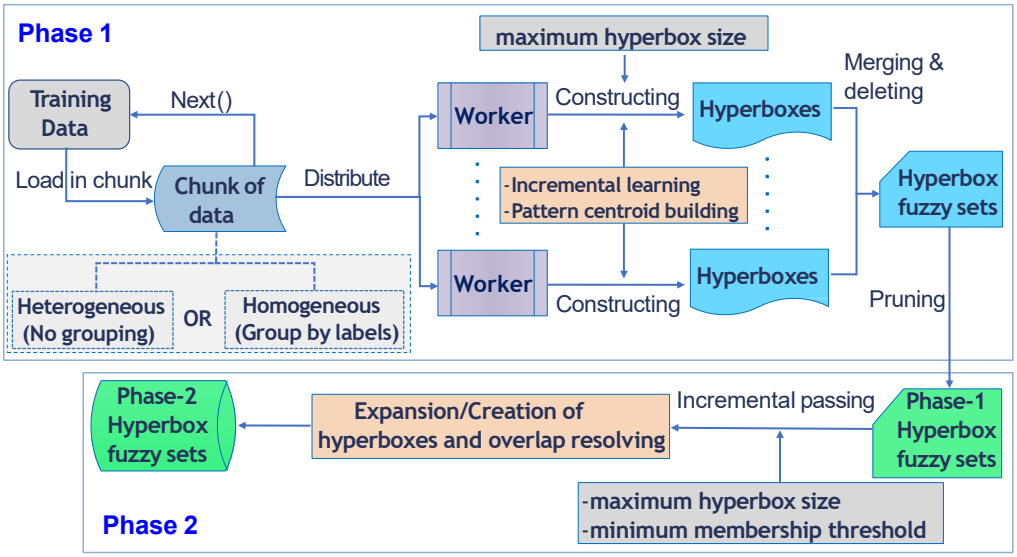


Figure 7.1 : Pipeline of the training process of the proposed method

In the first step of the training process, the input samples are split into disjoint sets and are then distributed to different computational workers. On each worker, an independent GFMMNN is built. When all training samples are handled, all created hyperboxes at different workers are merged to form a single model. Hyperboxes completely included in other hyperboxes representing the same class are eliminated to reduce the redundancy and complexity of the final model. After combining hyperboxes, the pruning procedure needs to be applied to eliminate noise and low-quality hyperboxes. The resulting hyperboxes are called phase-1 hyperboxes.

However, phase-1 hyperboxes have small sizes, so the complexity of the system can be high. As a result, all these hyperboxes are put through phase-2 of the granulation process with a gradual increase in the maximum hyperbox sizes. At a larger value of the maximum hyperbox size, hyperboxes at a low level of abstraction will be reconstructed with a higher data abstraction degree. Previously generated hyperboxes are fetched one at a time, and they are aggregated with newly constructed

hyperboxes at the current granular representation level based on a similarity threshold of the membership degree. This process is repeated for each input value of the maximum hyperbox sizes. The whole process of the proposed method is shown in Figure 7.1. Based on the classification error of resulting classifiers on the validation set, one can select an appropriate predictor satisfying both simplicity and precision. The following part provides the core concepts for both phases of the proposed method in a form of mathematical descriptions. The details of Algorithm E.1 for the phase 1 and Algorithm E.2 corresponding to the phase 2 as well as their implementation aspects are shown in the Appendix E. The readers can refer to this appendix section to find more about the free text descriptions, pseudo-codes, and implementation pipeline of the algorithms.

7.2.2 Formal Description

Consider a training set of N data vectors, $\mathcal{T}_N = \{X_i : X_i \in \mathbb{R}^n, i = 1, \ldots, N\}$, and the corresponding classes, $\mathcal{C} = \{c_i : c_i \in \mathbb{N}, i = 1, \ldots, N\}$; a validation set of N_V data vectors, $\mathcal{T}_{N_V}^{(V)} = \{X_i^{(V)} : X_i^{(V)} \in \mathbb{R}^n, i = 1, \ldots, N_V\}$, and the corresponding classes, $\mathcal{C}^{(V)} = \{c_i^{(V)} : c_i^{(V)} \in \mathbb{N}, i = 1, \ldots, N_V\}$. The details of the proposed method are formally described as follows.

Phase 1

Let n_w be the number of workers to execute the hyperbox construction process in parallel. Let $\mathbb{F}_j(\mathcal{T}_N^{(j)}, \mathcal{C}^{(j)}, \theta_0)$ be the procedure to construct hyperboxes on the j-th worker with maximum hyperbox size θ_0 using training data $\{\mathcal{T}_N^{(j)}, \mathcal{C}^{(j)}\}$. Procedure $\mathbb{F}_j$ is a modified fuzzy min-max neural network model which only creates new hyperboxes or expands existing hyperboxes. It accepts the overlapping regions among hyperboxes representing different classes, because it is expected to capture rapidly similar samples and group them into specific clusters by small-sized hyperboxes without spending much time on computationally expensive hyperbox overlap test and resolving steps. Instead, each hyperbox B_i is added a centroid G_i of patterns contained in that hyperbox and a counter n_i to store the number of data samples covered by it in addition to maximum and minimum points. This information is

used to classify data points located in the overlapping regions. When a new pattern X is presented to the classifier, the operation of building the pattern centroid for each hyperbox (line 12 and line 15 in Algorithm E.1 in the Appendix E) is performed according to Eq. (7.1).

$$G_i^{new} = \frac{n_i \cdot G_i^{old} + X}{n_i + 1} \tag{7.1}$$

where G_i is the sample centroid of the hyperbox B_i, n_i is the number of current samples included in the B_i. Next, the number of samples is updated: $n_i = n_i + 1$. It is noted that G_i is the same as the first pattern covered by the hyperbox B_i when B_i is newly created.

After the process of building hyperboxes in all workers finishes, merging step is conducted (lines 19-23 in Algorithm E.1 in the Appendix E), and it is mathematically represented as:

$$\mathbf{M} = \{B_i | B_i \in \bigcup_{j=1}^{n_w} \mathbb{F}_j(\mathcal{T}_N^{(j)}, \mathcal{C}^{(j)}, \theta_0)\} \tag{7.2}$$

where $\mathbf{M}$ is the model after performing the merging procedure. It is noted that hyperboxes contained in the larger hyperboxes representing the same class are eliminated (line 24 in Algorithm E.1 in the Appendix E) and the centroids of larger hyperboxes are updated using Eq. (7.3).

$$G_1^{new} = \frac{n_1 \cdot G_1^{old} + n_2 \cdot G_2^{old}}{n_1 + n_2} \tag{7.3}$$

where G_1 and n_1 are the centroid and the number of samples of the larger sized hyperbox, G_2 and n_2 are the centroid and the number of samples of the smaller sized hyperbox. The number of samples in the larger sized hyperbox is also updated: $n_1 = n_1 + n_2$. This whole process is similar to the construction of an ensemble classifier at the model level shown in Gabrys (2002b).

Pruning step is performed after merging hyperboxes to remove noise and low-quality hyperboxes (line 26 in Algorithm E.1 in the Appendix E). Mathematically,

it is defined as:

$$\mathbf{H}_0 = \begin{cases} \mathbf{M}_1 = \mathbf{M} \setminus \{B_k | \mathcal{A}_k < \alpha \ \vee \mathcal{A}_k = Nil\}, \text{ if } \mathcal{E}_V(\mathbf{M}_1) \leq \mathcal{E}_V(\mathbf{M}_2) \\ \mathbf{M}_2 = \mathbf{M} \setminus \{B_k | \mathcal{A}_k < \alpha\}, \text{ otherwise} \end{cases} \tag{7.4}$$

where $\mathbf{H}_0$ is the final model of stage 1 after applying the pruning operation, $\mathcal{E}_V(\mathbf{M}_i)$ is the classification error of the model $\mathbf{M}_i$ on the validation set $\{\mathcal{T}_{N_V}^{(V)}, \mathcal{C}^{(V)}\}$, α is the minimum accuracy of each hyperbox to be retained, and $\mathcal{A}_k$ is the predictive accuracy of hyperbox $B_k \in \mathbf{M}$ on the validation set defined as follows:

$$\mathcal{A}_k = \frac{\sum_{j=1}^{S_k} \mathcal{R}_{kj}}{\sum_{j=1}^{S_k} (\mathcal{R}_{kj} + \mathcal{I}_{kj})} \tag{7.5}$$

where S_k is the number of validation samples classified by hyperbox B_k, $\mathcal{R}_k$ is the number of samples predicted correctly by B_k, and $\mathcal{I}_k$ is the number of incorrect predicted samples. If $S_k = 0$, then $A_k = Nil$.

The classification step of unseen samples using model $\mathbf{H}_0$ is performed in the same way as in the GFMMNN with an exception of the case of many winning hyperboxes with the same maximum membership value. In such a case, the Euclidean distance from the input sample X to centroids G_i of winning hyperboxes B_i is computed using Eq. (7.6). If the input sample is a hyperbox, X is the coordinate of the center point of that hyperbox.

$$d(X, G_i) = \sqrt{\sum_{j=1}^{n} (x_j - G_{ij})^2} \tag{7.6}$$

The input pattern is then classified to the hyperbox B_i with the minimum value of $d(X, G_i)$.

Phase 2

Unlike phase 1, the input data in this phase are hyperboxes generated in the previous step. The purpose of phase 2 is to reduce the complexity of the model by aggregating hyperboxes created at a higher resolution level of granular data representations. At the high level of data abstraction, the confusion among hyperboxes

representing different classes needs to be removed. Therefore, the overlapping regions formed in phase 1 have to be resolved, and there is no overlap allowed in this phase. Phase 2 can be mathematically represented as:

$$\mathbf{H}_H(\Theta, m_s) = \{\mathbf{H}_i | \mathbf{H}_i = \mathbb{G}(\mathbf{H}_{i-1}, \theta_i, m_s)\}, \forall i \in [1, |\Theta|], \theta_i \in \Theta \tag{7.7}$$

where $\mathbf{H}_H$ is a list of models $\mathbf{H}_i$ constructed through different levels of granularity represented by maximum hyperbox sizes θ_i, Θ is a list of maximum hyperbox sizes, $|\Theta|$ is the cardinality of Θ, m_s is the minimum membership degree of two aggregated hyperboxes, and $\mathbb{G}$ is a procedure to construct the models in phase 2 (it uses the model at previous step as input), $\mathbf{H}_0$ is the model in phase 1. The aggregation rule of hyperboxes, $\mathbb{G}$, is described as follows:

For each input hyperbox B_h in $\mathbf{H}_{i-1}$, the membership values between B_h and all existing hyperboxes with the same class as B_h in $\mathbf{H}_i$ are computed. The winner hyperbox with maximum membership degree with respect to B_h is selected, denoted B_k, to aggregate with B_h. The following constraints are verified before conducting the aggregation:

- Maximum hyperbox size as defined in Eq. (3.4)
- The minimum membership degree:

$$b(B_h, B_k) \geq m_s \tag{7.8}$$

- Overlap test. New hyperbox aggregated from B_h and B_k does not overlap with any existing hyperboxes in $\mathbf{H}_i$ belonging to other classes

If the hyperbox B_k has not met all of the above conditions, the hyperbox with the next highest membership value is selected and the above process is repeated until the aggregation step occurs or no hyperbox candidate is left. If the input hyperbox cannot be mergered with existing hyperboxes in $\mathbf{H}_i$, it will be directly inserted into the current list of hyperboxes in $\mathbf{H}_i$. After that, the overlap test operation between the newly inserted hyperbox and hyperboxes in $\mathbf{H}_i$ representing other classes is

performed, and then the contraction process will be executed to resolve overlapping regions. The algorithm is iterated for all input hyperboxes in $\mathbf{H}_{i-1}$.

The classification process for unseen patterns using the hyperboxes in phase 2 is realized as in the GFMMNN. A detailed description of the implementation steps for the proposed method can be found in the Appendix E.1.2.

7.2.3 Missing Value Handling

The proposed method can deal with missing values since it inherits this characteristic from the GFMMNN as shown in Gabrys (2002c). A missing feature x_j is assumed to be able to receive values from the whole range, and it is presented by a real-valued interval as follows: $x_j^l = 1, x_j^u = 0$. By this initialization, the membership value associated with the missing value will be one, and thus the missing value does not cause any influence on the membership function of the hyperbox. During the training process, only observed features are employed for the update of hyperbox minimum and maximum vertices while missing variables are disregarded automatically. For the overlap test procedure in phase 2, only the hyperboxes which satisfy $v_{ij} \leq w_{ij}$ for all dimensions $j \in [1, n]$ are verified for the undesired overlapping areas. The second change is related to the way of calculating the membership value for the process of hyperbox selection or classification step of an input sample with missing values. Some hyperboxes' dimensions have not been set, so the membership function shown in Eq. (3.1) is changed to $b_i(X, \min(V_i, W_i), \max(W_i, V_i))$. With the use of this method, the training data uncertainty is represented in the classifier model.

7.2.4 Interpretability of the Proposed Classifier

One of the strong points of the use of hyperbox fuzzy sets for building classifiers is the obvious traceability, which can be used to explain the predictive results to users. For each input pattern coming to the network, membership functions are used to find the winner hyperbox, which is the hyperbox with the highest membership value among all existing hyperboxes, to assign the class label. This hyperbox together with the representative hyperboxes belonging to other classes, which are hyperboxes showing the maximum membership values among the hyperboxes within the same

class, can be presented to the users together with their membership values. For example, the explanation rules can be provided to users in the form of "If X is in the hyperbox B_i then the predicted class of X is the class of B_i" or "The predicted class of X is the class of B_i because the hyperbox B_i is nearest to X with a membership value of b_i. The next close hyperbox of X is B_k with class c_k showing a membership values of $b_k < b_i$." These factors help users to better understand the reason behind the selection of predicted classes. By using the learning method at higher levels of abstraction, the number of generated hyperboxes is significantly reduced, and thus the users may be easier to trace the hyperboxes and verify the predicted results.

However, the use of hyperbox fuzzy sets comes with a price of losing the linguistically interpretable fuzzy rules such as Mamdani-type fuzzy rules (Mamdani and Assilian 1975) or Takagi-Sugeno-type fuzzy rules (Sugeno 1985). To extract such linguistic fuzzy rules, the GFMMNN needs to be integrated with an additional layer to conduct the fuzzification of the crisp input values of features into linguistic values. This operation is not trivial for the current learning algorithms of the GFMMNN. To achieve this goal, it requires further studies in the future to possibly build a new membership function as well as adjusting the learning steps for linguistic features, and this is out of scope of this thesis.

7.3 Experiments

Macia et al. (2013) argued that data set selection poses a considerable impact on conclusions of the accuracy of learners, and then the authors advocated for considering properties of the datasets in experiments. They indicated the importance of employing artificial data sets constructed based on previously defined characteristics. In these experiments, therefore, two types of synthetic datasets with linear and non-linear class boundaries were considered. The number of features, the number of samples, and the number of classes for synthetic datasets were also changed to assess the variability in the performance of the proposed method. In practical applications, the data are usually not ideal as well as not following a standard distribution rule and including noisy data. Therefore, experiments were also carried

out on real datasets with diversity in numbers of samples, features, and classes.

For medium-sized real datasets such as *Letter*, *Magic*, *White wine quality*, and *Default of credit card clients*, the density-preserving sampling (DPS) method (Budka and Gabrys 2013) was used to separate the original datasets into training, validation, and test sets. For large-sized datasets, the hold-out method was used for splitting datasets, which is the simplest and the least computationally expensive approach to assessing the performance of classifiers because more advanced resampling approaches are not essential for large amounts of data (Budka and Gabrys 2013). The classification model is then trained on the training dataset. The validation set is used for the pruning step and evaluating the performance of the constructed classifier aiming to select a suitable model. The testing set is employed to assess the efficiency of the model on unseen data.

The experiments aim to answer the following questions:

- How is the classification accuracy of the predictor using multi-resolution hierarchical granular representations improved in comparison to the model using a fixed granulation value?
- How good is the performance of the proposed method compared with other types of fuzzy min-max neural networks and popular algorithms based on other data representations such as support vector machines, Naive Bayes, and decision trees?
- Whether a classifier with high accuracy can be obtained at high abstraction levels of granular representations?
- Whether the performance of the model on validation sets can be relied on to select a good model for unseen data, which satisfies both simplicity and accuracy?
- How good is the ability of handling missing values in datasets without requiring data imputation?

- How critical are the roles of the pruning process and the use of sample centroids?

The limitation of runtime for each execution is seven days. If an execution does not finish within seven days, it will be terminated, and the result is reported as N/A. In the experiments, parameters were set up as follows: $n_w = 4, \alpha = 0.5, m_s = 0.4, \gamma = 1$ because they gave the best results on a set of preliminary tests with validation sets for the parameter selection. All datasets are normalized to the range of [0, 1] because of the characteristic of the fuzzy min-max neural networks. Most of the datasets except the *susy* datasets utilized the value of 0.1 for θ_0 in phase 1, and $\Theta = \{0.2, 0.3, 0.4, 0.5, 0.6\}$ for different levels of granularity in phase 2. For the *susy* dataset, due to the complexity and limitation of runtime for the proposed method and other compared types of fuzzy min-max neural networks, the $\theta_0 = 0.3$ was used for phase 1, and $\Theta = \{0.4, 0.5, 0.6\}$ was employed for phase 2. For Naive Bayes, Gaussian Naive Bayes (GNB) algorithm was used for classification. The radial basis function (RBF) was used as a kernel function for the support vector machines (SVM). The default setting parameters in the *scikit-learn* library were used for GNB, SVM, and decision tree (DT) in the experiments. The performance of the proposed method was also compared to other types of fuzzy min-max neural networks such as the original fuzzy min-max neural network (FMNN) (Simpson 1992), the enhanced fuzzy min-max neural network (EFMNN) (Mohammed and Lim 2015), the enhanced fuzzy min-max neural network with a K-nearest hyperbox expansion rule (KNEFMNN) (Mohammed and Lim 2017a), and the general fuzzy min-max neural network (GFMMNN) (Gabrys and Bargiela 2000). These types of fuzzy min-max neural networks used the same pruning procedure as the proposed method.

Synthetic datasets in the experiments were generated by using Gaussian distribution functions, so GNB and SVM with RBF kernel which use Gaussian distribution assumptions to classify data will achieve nearly optimal error rates because they match perfectly with underlying data distribution. Meanwhile, fuzzy min-max classifiers employ the hyperboxes to cover the input data, thus they are not an optimal

representation for underlying data. Therefore, the accuracy of hyperbox-based classifiers on synthetic datasets cannot outperform the predictive accuracy of Gaussian NB or SVM with RBF kernel. However, Gaussian NB is a linear classifier, and thus, it only outputs highly accurate predictive results for datasets with linear decision boundary. In contrast, decision tree, fuzzy min-max neural networks, and SVM with RBF kernel are universal approximators, and they can deal effectively with both linear and non-linear classification problems.

All experiments were conducted on the computer with Xeon 6150 2.7GHz CPU and 180GB RAM. Each experiment was repeated five times to compute the average training time. The accuracy of types of fuzzy min-max neural networks remains the same through different iterations because they only depend on the data presentation order and the same order of training samples was maintained during the experiments.

7.3.1 Performance of the Proposed Method on Synthetic Datasets

The first experiment was conducted on the synthetic datasets with the linear or non-linear boundary between classes. For each synthetic dataset, a testing set containing 100,000 samples and a validation set with 10,000 instances were generated using the same probability density function as the training sets.

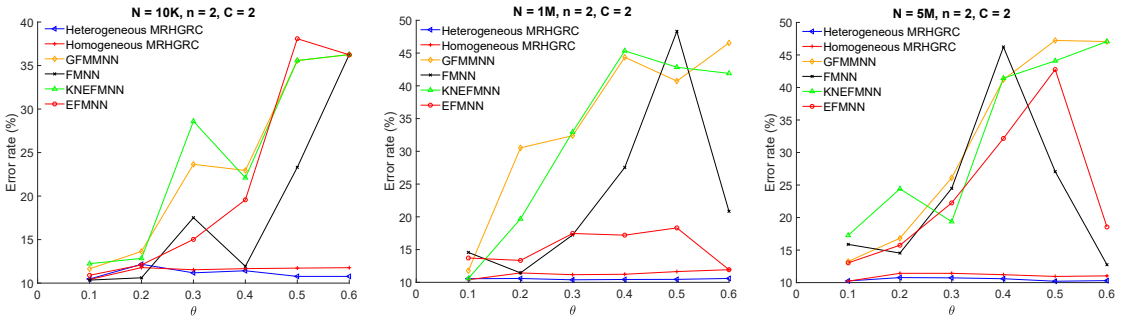
Linear boundary datasets

Increase the number of samples:

Both the number of dimensions $n = 2$ and the number of classes $C = 2$ were kept the same, and only the number of samples was changed to evaluate the impact of the number of patterns on the performance of classifiers. Gaussian distribution was used to construct synthetic datasets as described in Fukunaga (1990). The means of the Gaussians of two classes are given as follows: $\mu_1 = [0, 0]^T, \mu_2 = [2.56, 0]^T$, and the covariance matrices are as follows:

$$\Sigma_1 = \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

With the use of these configuration parameters, training sets with different sizes (10K, 1M, and 5M samples) were formed. Fukunaga (1990) indicated that the



(a) 10K samples (b) 1M samples (c) 5M samples

Figure 7.2 : The error rate of classifiers on synthetic linear boundary datasets with the different number of samples

general Bayes error of the datasets formed from these settings is around 10%. An equal number of samples for each class was generated to remove the impact of imbalanced class property on the performance of classifiers. Figure 7.2 shows the change in the error rates of different fuzzy min-max classifiers on the testing synthetic linear boundary datasets with the different numbers of training patterns when the level of granularity (θ) changes. The other fuzzy min-max neural networks used the fixed value of θ to construct the model, while the proposed method builds the model starting from the defined lowest value of θ (phase 1) to the current threshold. For example, the model at $\theta = 0.3$ in the proposed method is constructed with $\theta = 0.1$, $\theta = 0.2$, and $\theta = 0.3$.

It can be seen from Figure 7.2 that the error rates of the proposed method are lower than those of other fuzzy min-max classifiers, especially in high abstraction levels of granular representations. At high levels of abstraction (corresponding to high values of θ), the error rates of other classification models are relatively high, while the proposed classifier still maintains the low error rate, just a little higher than the error at a high-resolution level of granular data. The lowest error rates of the different classifiers on validation (E_V) and testing (E_T) sets, as well as total training time for six levels of abstraction, are shown in Table 7.1. Best results are highlighted in bold in each table. The training time reported in this chapter consists of time for reading training files and model construction.

Table 7.1 : The lowest error rates and training time of classifiers on synthetic linear boundary datasets with different number of samples ($n = 2, C = 2$)

N	**Algorithm**	$\min E_V$	$\min E_T$	θ_V	θ_T	**Time (s)**
10K	He-MRHGRC	10.25	10.467	0.1	0.1	1.1378
	Ho-MRHGRC	10.1	10.413	0.1	0.1	1.3215
	GFMM	11.54	11.639	0.1	0.1	8.6718
	FMNN	10.05	10.349	0.1	0.1	46.4789
	KNEFMNN	12.07	12.232	0.1	0.1	9.4459
	EFMNN	10.44	10.897	0.1	0.1	48.9892
	GNB	**9.85**	**9.964**	-	-	0.5218
	SVM	9.91	9.983	-	-	1.5468
	DT	15.33	14.861	-	-	0.5405
1M	He-MRHGRC	10.31	10.386	0.3	0.3	20.0677
	Ho-MRHGRC	10.24	10.401	0.1	0.1	16.0169
	GFMM	11.47	11.783	0.1	0.1	405.4642
	FMNN	10.98	11.439	0.2	0.2	13163.1404
	KNEFMNN	10.36	10.594	0.1	0.1	413.8296
	EFMNN	11.61	11.923	0.6	0.6	10845.1280
	GNB	9.87	**9.972**	-	-	5.0133
	SVM	**9.86**	9.978	-	-	21798.2803
	DT	14.873	14.682	-	-	9.9318
5M	He-MRHGRC	10.11	10.208	0.5	0.5	101.9312
	Ho-MRHGRC	10.04	10.222	0.1	0.1	75.2254
	GFMM	13.14	13.243	0.1	0.1	1949.2138
	FMNN	12.68	12.751	0.6	0.6	92004.7253
	KNEFMNN	17.31	17.267	0.1	0.1	1402.1173
	EFMNN	12.89	13.032	0.1	0.1	41888.5296
	GNB	**9.88**	**9.976**	-	-	22.9343
	SVM	N/A	N/A	-	-	N/A
	DT	15.253	14.692	-	-	70.2041

It can be seen that the accuracy of the proposed method on unseen data using the heterogeneous data distribution (He-MRHGRC) regularly outperforms the accuracy of the classifier built based on the homogeneous data distribution (Ho-MRHGRC) using large-sized training sets. It is also observed that the proposed method is less affected by overfitting when increasing the number of training samples while keeping the same testing set. For other types of fuzzy min-max neural networks, their error rates frequently increase with the increase in training size because of overfitting. The total training time of the proposed algorithm is faster than that

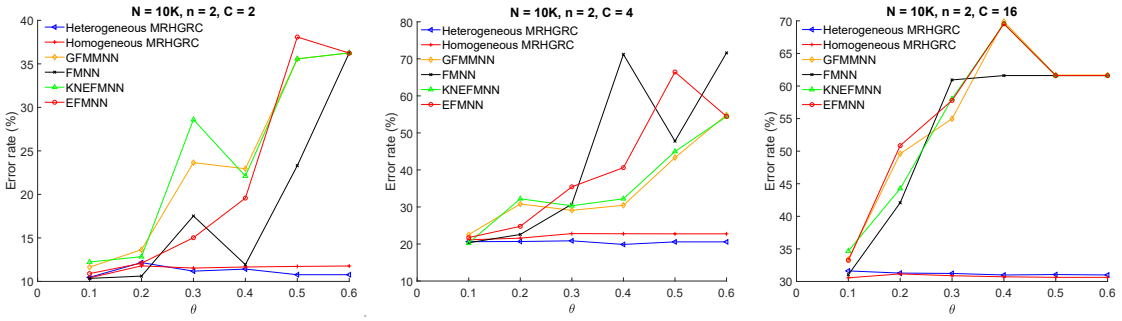
of other types of fuzzy min-max classifiers since the proposed method executes the hyperbox building process at the lowest value of θ in parallel, and the overlapping areas among hyperboxes representing different classes are allowed to rapidly capture the characteristics of sample points locating near each other. The hyperbox overlap resolving step is only performed at higher abstraction levels with a smaller number of input hyperboxes.

The proposed method also achieves better classification accuracy compared to the decision tree, but it cannot overcome the support vector machines and Gaussian Naive Bayes methods on synthetic linear boundary datasets. However, the training time of the support vector machines on large-sized datasets is costly, even becomes unacceptable on training sets with millions of patterns. The synthetic datasets were constructed based on the Gaussian distribution, so the Gaussian Naive Bayes method can reach the minimum error rate, but the proposed approach can also obtain the error rates relatively near these optimal error values. It can be observed that the best performance of the He-MRHGRC attains at quite high abstraction levels of granular representations because some noisy hyperboxes at high levels of granularity are eliminated at lower granulation levels. These results demonstrate the efficiency and scalability of the proposed approach.

Increase the number of classes:

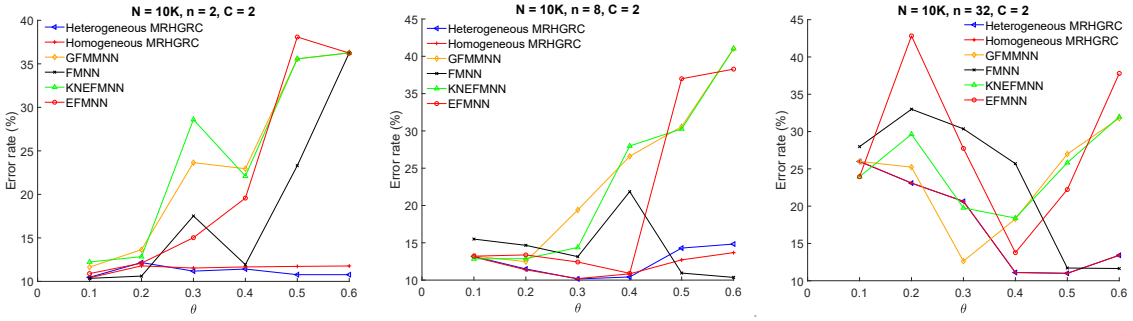
The purpose of the experiment in this subsection is to evaluate the performance of the proposed method on multi-class datasets. The number of dimensions $n = 2$, the number of samples $N = 10,000$ were remained unchanged, and only changed the number of classes to form synthetic multi-class datasets with $C \in \{2, 4, 16\}$. The covariance matrices stay the same as in the case of changing the number of samples.

Figure 7.3 shows the change in error rates of fuzzy min-max classifiers with a different number of classes on the testing sets. It can be easily seen that the error rates of the proposed method are lowest compared to other fuzzy min-max neural networks on all multi-class synthetic datasets at high abstraction levels of granular representations. At high abstraction levels, the error rates of other fuzzy min-max neural networks increase rapidly, while the error rate of the proposed



(a) 2 classes (b) 4 classes (c) 16 classes

Figure 7.3 : The error rate of classifiers on synthetic linear boundary datasets with the different number of classes



(a) 2 features (b) 8 features (c) 32 features

Figure 7.4 : The error rate of classifiers on synthetic linear boundary datasets with the different number of features

classifier still maintains the stability. In addition, the error rates of the proposed method also slowly augment in contrast to the behaviors of other considered types of fuzzy min-max neural networks when increasing the abstraction level of granular representations. These facts demonstrate the efficiency of the proposed method on multi-class datasets. The lowest error rates of classifiers on validation and testing sets, as well as total training time, are shown in Table 7.2. It is observed that the predictive accuracy of the proposed method outperforms all considered types of fuzzy min-max classifiers and decision tree, but it cannot overcome the Gaussian Naive Bayes and support vector machine methods. The training time of the proposed method is faster than other fuzzy min-max neural networks and support vector

Table 7.2 : The lowest error rates and training time of classifiers on synthetic linear boundary datasets with different classes ($N = 10K, n = 2$)

C	Algorithm	min E_V	min E_T	θ_V	θ_T	Time (s)
2	He-MRHGRC	10.25	10.467	0.1	0.1	1.1378
	Ho-MRHGRC	10.10	10.413	0.1	0.1	1.3215
	GFMM	11.54	11.639	0.1	0.1	8.6718
	FMNN	10.05	10.349	0.1	0.1	46.4789
	KNEFMNN	12.07	12.232	0.1	0.1	9.4459
	EFMNN	10.44	10.897	0.1	0.1	48.9892
	GNB	**9.85**	**9.964**	-	-	0.5218
	SVM	9.91	9.983	-	-	1.5468
	DT	15.33	14.861	-	-	0.5405
4	He-MRHGRC	19.76	19.884	0.4	0.4	1.0754
	Ho-MRHGRC	19.97	21.135	0.1	0.1	1.5231
	GFMM	22.34	22.515	0.1	0.1	10.8844
	FMNN	20.00	20.350	0.1	0.1	65.7884
	KNEFMNN	20.54	20.258	0.1	0.1	12.5618
	EFMNN	21.75	21.736	0.1	0.1	55.1921
	GNB	19.35	**19.075**	-	-	0.5492
	SVM	**19.34**	19.082	-	-	1.6912
	DT	26.94	27.014	-	-	0.5703
16	He-MRHGRC	30.11	30.996	0.1	0.4	1.2686
	Ho-MRHGRC	28.70	30.564	0.1	0.1	1.8852
	GFMM	32.66	33.415	0.1	0.1	18.0554
	FMNN	29.78	31.035	0.1	0.1	69.6761
	KNEFMNN	33.42	34.670	0.1	0.1	22.3418
	EFMNN	31.80	33.239	0.1	0.1	76.0920
	GNB	**27.12**	28.190	-	-	0.5764
	SVM	27.29	**28.103**	-	-	1.6455
	DT	38.813	39.644	-	-	0.6023

machines on the considered multi-class synthetic datasets.

Increase the number of features:

To generate the multi-dimensional synthetic datasets with the number of samples $N = 10K$ and the number of classes $C = 2$, similar settings were used as in generation of datasets with different number of samples. The means of classes are $\mu_1 = [0, \ldots, 0]^T, \mu_2 = [2.56, 0, \ldots, 0]^T$, and the covariance matrices are as follows:

$$\Sigma_1 = \Sigma_2 = \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix} \tag{7.9}$$

The size of each expression corresponds to the number of dimensions n of the problem. Fukunaga (Fukunaga 1990) stated that the general Bayes error of 10% and this Bayes error stays the same even when n changes.

Table 7.3 : The lowest error rates and training time of classifiers on synthetic linear boundary datasets with different features ($N = 10K, C = 2$)

n	**Algorithm**	$\min E_V$	$\min E_T$	θ_V	θ_T	**Time (s)**
2	He-MRHGRC	10.250	10.467	0.1	0.1	1.1378
	Ho-MRHGRC	10.100	10.413	0.1	0.1	1.3215
	GFMM	11.540	11.639	0.1	0.1	8.6718
	FMNN	10.050	10.349	0.1	0.1	46.4789
	KNEFMNN	12.070	12.232	0.1	0.1	9.4459
	EFMNN	10.440	10.897	0.1	0.1	48.9892
	GNB	**9.850**	**9.964**	-	-	0.5218
	SVM	9.910	9.983	-	-	1.5468
	DT	15.330	14.861	-	-	0.5405
8	He-MRHGRC	10.330	10.153	0.3	0.3	21.9131
	Ho-MRHGRC	10.460	10.201	0.3	0.3	23.0554
	GFMM	12.170	12.474	0.1	0.2	196.0682
	FMNN	10.250	10.360	0.6	0.6	302.8683
	KNEFMNN	12.720	12.844	0.1	0.1	618.2524
	EFMNN	11.300	10.907	0.4	0.4	579.3113
	GNB	**9.940**	**9.919**	-	-	0.5915
	SVM	9.980	9.927	-	-	2.0801
	DT	15.383	15.087	-	-	0.6769
32	He-MRHGRC	11.070	10.995	0.5	0.5	226.3193
	Ho-MRHGRC	11.070	10.995	0.5	0.5	226.0611
	GFMM	12.390	12.625	0.3	0.3	847.6977
	FMNN	11.830	11.637	0.5	0.6	1113.6836
	KNEFMNN	17.410	18.395	0.1	0.4	837.9571
	EFMNN	13.890	13.766	0.4	0.4	1114.4976
	GNB	10.280	10.088	-	-	0.7154
	SVM	**10.220**	**10.079**	-	-	4.5937
	DT	15.400	15.201	-	-	1.0960

Figure 7.4 shows the change in the error rates with different levels of granularity on multi-dimensional synthetic datasets. In general, with a low number of dimen-

sions, the proposed method outperforms other fuzzy min-max neural networks. With high dimensionality and a small number of samples, the high levels of granularity result in high error rates, and misclassification results considerably drops when the value of θ increases. The same trend also happens to the FMNN when its accuracy at $\theta = 0.5$ or $\theta = 0.6$ is quite high. Apart from the FMNN on high dimensional datasets, the proposed method is better than three other fuzzy min-max classifiers at high abstraction levels. Table 7.3 reports the lowest error rates of classifiers on validation and testing multi-dimensional sets as well as the total training time through six abstraction levels of granular representations. The training time of the proposed method is much faster than other types of fuzzy min-max neural networks. Generally, the performance of the proposed method overcomes the decision tree and other types of fuzzy min-max neural networks, but its predictive results cannot defeat the Gaussian Naive Bayes and support vector machines. It can be observed that the best performance on validation and testing sets obtains at the same abstraction level of granular representations on all considered multi-dimensional datasets. This fact indicates that the validation set can be used to choose the best classifier at a given abstraction level among constructed models through different granularity levels.

Non-linear boundary datasets

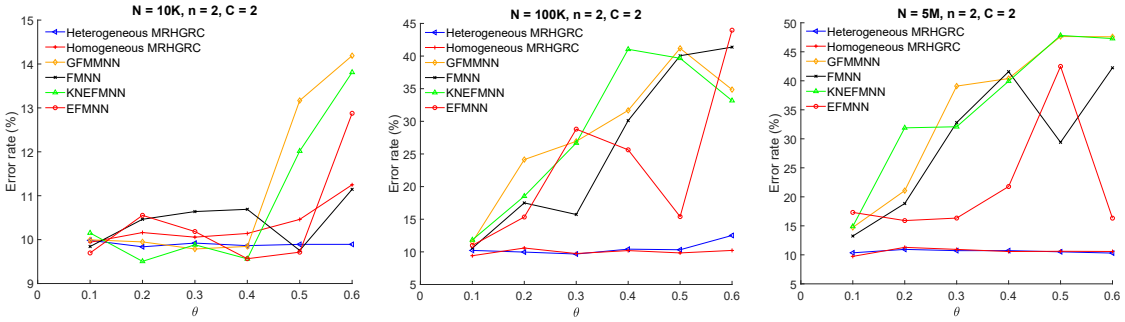
To generate non-linear boundary datasets, the Gaussian means of the first class were set up as follows: $\mu_1 = [-2, 1.5]^T, \mu_2 = [1.5, 1]^T$ and the Gaussian means of the second class: $\mu_3 = [-1.5, 3]^T, \mu_4 = [1.5, 2.5]^T$. The covariance matrices for the first class Σ_1, Σ_2 and for the second class Σ_3, Σ_4 were established as follows:

$$\begin{aligned} \Sigma_1 &= \begin{bmatrix} 0.5 & 0.05 \\ 0.05 & 0.4 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 0.5 & 0.05 \\ 0.05 & 0.3 \end{bmatrix}, \\ \Sigma_3 &= \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}, \Sigma_4 = \begin{bmatrix} 0.5 & 0.05 \\ 0.05 & 0.2 \end{bmatrix}, \end{aligned} \tag{7.10}$$

The number of samples for each class was equal, and the generated samples were normalized to the range of [0, 1]. Only a testing set including 100,000 samples and a validation set with 10,000 patterns were created. Three different training sets

containing 10K, 100K, and 5M samples were used to train classifiers. The purpose is to evaluate the predictive results of the proposed method on the non-linear boundary dataset when changing the sizes of the training set.

Figure 7.5 shows the changes in the error rates through different levels of granularity of classifiers on non-linear boundary datasets. It can be observed that the error rates of the proposed method trained on the large-sized non-linear boundary datasets are better than those using other types of fuzzy min-max neural networks, especially at high abstraction levels of granular representations. While other fuzzy min-max neural networks show the increase in the error rates if the value of θ grows up, the proposed method is capable of maintaining the stability of predictive results even in the case of high abstraction levels. When the number of samples increases, the error rates of other fuzzy min-max classifiers usually rise, whereas the error rate in the proposed approach only fluctuates a little. These results indicate that the proposed method may reduce the influence of overfitting because of constructing higher abstraction level of granular data representations using the learned knowledge from lower abstraction levels.



(a) 10K samples (b) 100K samples (c) 5M samples

Figure 7.5 : The error rate of classifiers on synthetic non-linear boundary datasets with the different number of samples

The best performance of the proposed approach does not often happen at the smallest value of θ on these non-linear datasets. Results regarding accuracy on validation and testing sets reported in Table 7.4 confirm this statement. These figures also illustrate the effectiveness of the processing steps in phase 2. Unlike

the linear boundary datasets, the proposed method overcomes GNB to become two best classifiers (together with SVM) among classifiers considered. Although SVM outperformed the proposed approach, its runtime on large-sized datasets was much slower than the proposed method. The training time of the proposed algorithm is much faster than other types of fuzzy min-max neural networks and SVM, but it is still slower than GNB and decision tree techniques.

Table 7.4 : The lowest error rates and training time of classifiers on synthetic non-linear boundary datasets with different number of samples ($n = 2, C = 2$)

N	**Algorithm**	$\min E_V$	$\min E_T$	θ_V	θ_T	**Time (s)**
10K	He-MRHGRC	9.950	9.836	0.2	0.2	0.9616
	Ho-MRHGRC	9.820	9.940	0.1	0.1	1.1070
	GFMM	10.200	9.787	0.4	0.5	10.5495
	FMNN	9.770	9.753	0.5	0.5	61.1130
	KNEFMNN	9.890	9.505	0.2	0.2	16.1099
	EFMNN	**9.750**	9.565	0.1	0.4	60.6073
	GNB	10.740	10.626	-	-	0.5218
	SVM	**9.750**	**9.490**	-	-	1.5565
	DT	14.107	13.831	-	-	0.5388
100K	He-MRHGRC	10.130	9.670	0.3	0.3	2.5310
	Ho-MRHGRC	9.910	9.412	0.1	0.1	2.3560
	GFMM	11.810	11.520	0.1	0.1	44.7778
	FMNN	10.880	10.575	0.1	0.1	588.4412
	KNEFMNN	12.470	11.836	0.1	0.1	42.9151
	EFMNN	11.020	10.992	0.1	0.1	485.7613
	GNB	10.830	10.702	-	-	0.9006
	SVM	**9.650**	**9.338**	-	-	93.4474
	DT	14.277	13.642	-	-	1.1767
5M	He-MRHGRC	10.370	10.306	0.1	0.6	91.7894
	Ho-MRHGRC	**9.940**	**9.737**	0.1	0.1	69.5106
	GFMM	15.260	14.730	0.1	0.1	1927.6191
	FMNN	13.160	13.243	0.1	0.1	53274.4387
	KNEFMNN	15.040	14.905	0.1	0.1	1551.5220
	EFMNN	15.660	15.907	0.2	0.2	54487.6978
	GNB	10.840	10.690	-	-	22.9849
	SVM	N/A	N/A	-	-	N/A
	DT	13.790	13.645	-	-	49.9919

7.3.2 Performance of the Proposed Method on Real Datasets

Real datasets used for the experiments in this section are described in Table E.1 in Appendix E. To assess the significant reduction in the number of generated hyperboxes while still maintaining good performance, medium and large sized datasets were used in this chapter. Therefore, these datasets are different from small sized datasets employed in previous chapters. From the results of synthetic datasets, it can be seen that the performance of the multi-resolution hierarchical granular representation based classifier using the heterogeneous data distribution technique is more stable than that utilizing the homogeneous distribution method. Therefore, the experiments in the rest of this chapter were conducted for only the heterogeneous classifier.

Table 7.5 : The real datasets and their statistics for experiments in Chapter 7

Dataset	**#Dimensions**	**#Classes**	**#Training**	**#Validation**	**#Testing**	**Source**
Poker Hand	10	10	25,010	50,000	950000	LIBSVM
SensIT Vehicle	100	3	68,970	9,852	19,706	LIBSVM
Skin_NonSkin	3	2	171,540	24,260	49,257	LIBSVM
Covtype	54	7	406,709	58,095	116,208	LIBSVM
White wine quality	11	7	2,449	1,224	1,225	Kaggle
PhysioNet MIT-BIH Arrhythmia	187	5	74,421	13,133	21,892	Kaggle
MAGIC Gamma Telescope	10	2	11,887	3,567	3,566	UCI
Letter	16	26	15,312	2,188	2,500	UCI
Default of credit card clients	23	2	18,750	5,625	5,625	UCI
MoCap Hand Postures	36	5	53,104	9,371	15,620	UCI
MiniBooNE	50	2	91,044	12,877	26,143	UCI
SUSY	18	2	4,400,000	100,000	500,000	UCI

Table 7.6 shows the number of generated hyperboxes for the He-MRHGRC on real datasets at different abstraction levels of granular representations. It can be seen that the number of hyperboxes at the value of $\theta = 0.6$ is significantly reduced in comparison to those at $\theta = 0.1$. However, the error rates of the classifiers on testing

Table 7.6 : The change in the number of generated hyperboxes through different levels of granularity of the proposed method

Dataset	θ					
	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**
Skin_NonSkin	1012	248	127	85	64	51
Poker Hand	11563	11414	10905	3776	2939	2610
Covtype	94026	13560	5224	2391	1330	846
SensIT Vehicle	5526	2139	1048	667	523	457
PhysioNet MIT-BIH Arrhythmia	60990	26420	15352	8689	5261	3241
White wine quality	1531	676	599	559	544	526
Default of credit card clients	2421	529	337	76	48	29
Letter	9236	1677	952	646	595	556
MAGIC Gamma Telescope	1439	691	471	384	335	308
MiniBooNE	444	104	24	10	6	6
SUSY	-	-	26187	25867	16754	13017

sets at $\theta = 0.6$ do not change so much compared to those at $\theta = 0.1$. This fact is illustrated in Figure 7.6 and Figure E.3 in the Appendix E. From these figures, it is observed that at the high values of the maximum hyperbox size such as $\theta = 0.5$ and $\theta = 0.6$, the proposed classifier achieves the best performance compared to other considered types of fuzzy min-max neural networks. It can also be observed that the prediction accuracy of the proposed method is usually much better than that using other types of fuzzy min-max classifiers on most of the data granulation levels. The error rate of the proposed classifier regularly increases slowly with the increase in the abstraction level of granules, even in some cases, the error rate declines at a high abstraction level of granular representations. The best performance of classifiers on validation and testing sets, as well as training time through six granularity levels, are reported in Table E.2 in the Appendix E.

Although the proposed method cannot achieve the best classification accuracy on all considered datasets, its performance is located in the top 2 for all datasets. The Gaussian Naive Bayes classifiers obtained the best predictive results on synthetic linear boundary datasets, but it fell to the last position and became the worst classifier on real datasets because real datasets are highly non-linear. On datasets with highly non-linear decision boundaries such as *covtype*, *PhysioNet MIT-BIH Ar-*

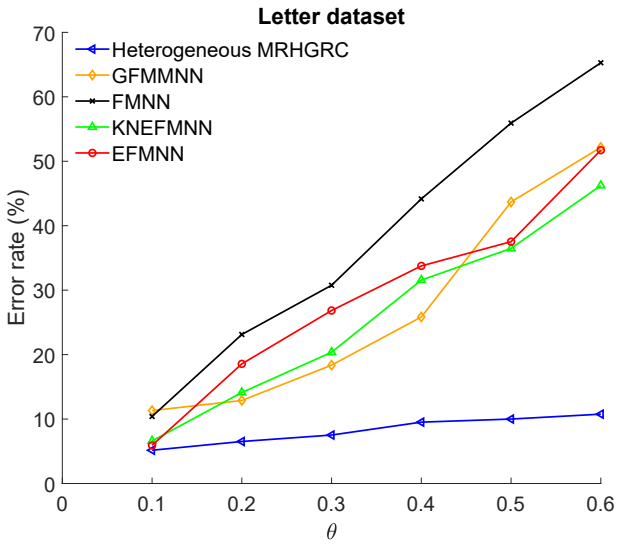


Figure 7.6 : The error rate of classifiers on the *Letter* datasets through data abstraction levels

rhythmia, and *MiniBooNE*, the proposed method still produces the good predictive accuracy.

The training process of the proposed method is much faster than other types of fuzzy min-max neural networks on all considered datasets. Notably, on several large-sized complex datasets such as *covtype* and *susy*, the training time of other fuzzy min-max classifiers is costly, but their accuracy is worse than the proposed method, which takes less training time. The proposed approach is frequently faster than SVM and can deal with datasets with millions of samples, while the SVM predictor cannot perform.

On many datasets, the best predictive results on validation and testing sets were achieved at the same abstraction level of granular representations. In the case that the best model on the validation set has different abstraction level compared to the best model on the testing set, the error rate on the testing set if using the best classifier on the validation set is also near the minimum error. These figures show that the proposed method is stable, and it can achieve a high predictive accuracy on both synthetic and real datasets.

7.3.3 The Vital Role of the Pruning Process and the Use of Sample Centroids

This experiment aims to assess the important roles of the pruning process and the use of sample centroids on the performance of the proposed method. The experimental results related to these issues are presented in Table 7.7. It is easily observed that the pruning step contributes to significantly reducing the number of generated hyperboxes, especially in *SensIT Vehicle*, *Default of credit card clients*, *susy* datasets. When the poorly performing hyperboxes are removed, the accuracy of the model increases considerably. These figures indicate the critical role of the pruning process with regards to reducing the complexity of the model and enhancing the predictive performance.

Table 7.7 : The role of the pruning process and the use of sample centroids

Dataset	**Num hyperboxes**		**Error rate before pruning (%)**	**Error rate after pruning (%)**	**No. of predicted samples using centroids before pruning**		**No. of predicted samples using centroids after pruning**	
	Before pruning	**After pruning**			**Total**	**Wrong**	**Total**	**Wrong**
Skin_NonSkin	1,358	1,012	0.1726	0.0974	1,509	73	594	30
Poker Hand	24,991	11,563	53.5951	49.8128	600,804	322,962	725,314	362,196
SensIT Vehicle	61,391	5,526	23.6730	20.9073	2	1	0	0
Default of credit card clients	9,256	2,421	22.3822	19.7689	662	291	312	127
Covtype	95971	94026	7.7335	7.5356	2700	975	2213	783
PhysioNet MIT-BIH Arrhythmia	61,419	60,990	3.6589	3.5492	49	9	48	8
MiniBooNE	1,079	444	16.4289	13.9043	14,947	3,404	11,205	2,575
SUSY	55,096	26,187	30.8548	28.3456	410,094	145,709	370,570	124,850

It can also be seen that the use of sample centroids and Euclidean distance may predict accurately from 50% to 95% of the samples located in the overlapping regions between different classes. The predictive accuracy depends on the distribution and

complexity of underlying data. With the use of sample centroids, the overlap test and contraction process are unnecessary to be used in phase 1 at the highest level of granularity. This strategy leads to accelerating the training process of the proposed method compared to other types of fuzzy min-max neural networks, especially in large-sized datasets such as *covtype* or *susy*. These facts point to the effectiveness of the pruning process and the usage of sample centroids on improving the performance of the proposed approach in terms of both accuracy and training time.

7.3.4 Ability to Handling Missing Values

This experiment was conducted on two datasets containing many missing values, i.e., *PhysioNet MIT-BIH Arrhythmia* and *MoCap Hand Postures* datasets. The aim of this experiment is to demonstrate the ability to handle missing values of the proposed method to preserve the uncertainty of input data without doing any pre-processing steps. Three other training datasets were generated from the original data by replacing missing values with the zero, mean, or median value of each feature. Then, these values were used to fill in the missing values of corresponding features in the testing and validation sets. The obtained results are presented in Table 7.8. The predictive accuracy of the classifier trained on the datasets with missing values cannot be superior to ones trained on the datasets imputed by the median, mean or zero values. However, the training time is reduced, and the characteristic of the proposed method is still preserved, in which the accuracy of the classifier is maintained at high levels of abstraction, and its behavior is nearly the same on both validation and testing sets. The replacement of missing values by other values is usually biased and inflexible in real-world applications. The capability of deducing directly from data with missing values ensures the maintenance of the online learning property of the fuzzy min-max neural network on the incomplete input data.

7.3.5 Comparison to State-of-the-art Studies

The purpose of this section is to compare the proposed method with recent studies of classification algorithms on large-sized datasets in physics and medical diagnostics. The first experiment was performed on the *susy* dataset to distinguish

Table 7.8 : The training time and the lowest error rates of the proposed method on the datasets with missing values

Dataset	**Training time (s)**	$\min E_{Val}$	$\min E_{Test}$
Arrhythmia with replacing missing values by zero values	53,100.2895	3.0762 ($\theta = 0.1$)	3.5492 ($\theta = 0.1$)
Arrhythmia with replacing missing values by mean values	60,980.5110	2.6879 ($\theta = 0.1$)	3.3848 ($\theta = 0.1$)
Arrhythmia with replacing missing values by median values	60,570.4315	2.7031 ($\theta = 0.1$)	3.2980 ($\theta = 0.2$)
Arrhythmia with missing values retained	58,188.8138	2.6955 ($\theta = 0.1$)	3.1473 ($\theta = 0.1$)
Postures with replacing missing values by zero values	5,845.9722	6.6482 ($\theta = 0.1$)	7.7529 ($\theta = 0.4$)
Postures with replacing missing values by mean values	5,343.0038	8.5370 ($\theta = 0.1$)	9.7631 ($\theta = 0.3$)
Postures with replacing missing values by median values	4,914.4475	8.4089 ($\theta = 0.1$)	9.9936 ($\theta = 0.3$)
Postures with missing values retained	2,153.8121	14.5662 ($\theta = 0.4$)	13.7900 ($\theta = 0.4$)

between a signal process producing super-symmetric particles and a background process. To attain this purpose, Baldi et al. (2014) compared the performance of a deep neural network with boosted decision trees using the area under the curve (AUC) metrics. In another study, Sakai et al. (2018) evaluated different methods of AUC optimization in combination with support vector machines to enhance the efficiency of the final predictive model. The AUC values of these studies together with the proposed method are reported in Table 7.9. It can be seen that the proposed approach overcomes all approaches in Sakai's research, but it cannot outperform the deep learning methods and boosted trees on the considered dataset.

The second experiment was conducted on a medical dataset (*PhysioNet MIT-BIH Arrhythmia*) containing Electrocardiogram (ECG) signal used for the classification of heartbeats. There are many studies on ECG heartbeat classification such as deep residual convolution neural network (Kachuee et al. 2018), a 9-layer deep convolutional neural network on the augmentation of the original data (Acharya et al. 2017), combinations of a discrete wavelet transform with neural networks, SVM (Martis et al. 2013), and random forest (Li and Zhou 2016). The *PhysioNet MIT-*

Table 7.9 : The AUC value of the proposed method and other methods on the *susy* dataset

Method	**AUC**
Boosted decision tree (Baldi et al. 2014)	0.863
Deep neural network (Baldi et al. 2014)	0.876
Deep neural network with dropout (Baldi et al. 2014)	**0.879**
Positive-Negative and unlabeled data based AUC optimization (Sakai et al. 2018)	0.647
Semi-supervised rankboost based AUC optimization (Sakai et al. 2018)	0.709
Semi-supervised AUC-optimized logistic sigmoid (Sakai et al. 2018)	0.556
Optimum AUC with a generative model (Sakai et al. 2018)	0.577
He-MRHGRC (The proposed method)	0.799

Table 7.10 : The accuracy of the proposed method and other methods on the *PhysioNet MIT-BIH Arrhythmia* dataset

Method	**Accuracy(%)**
Deep residual Convolutional neural network (Kachuee et al. 2018)	93.4
Augmentation + Deep convolutional neural network (Acharya et al. 2017)	93.5
Discrete wavelet transform + SVM (Martis et al. 2013)	93.8
Discrete wavelet transform + NN (Martis et al. 2013)	94.52
Discrete wavelet transform + Random Forest (Li and Zhou 2016)	94.6
The proposed method on the dataset with the missing values	**96.85**
The proposed method on the dataset with zero padding	96.45

BIH Arrhythmia dataset contains many missing values and above studies used the zero padding mechanism for these values. The proposed method can directly handle missing values without any imputations. The accuracy of the proposed method on the datasets with missing values and zero paddings is shown in Table 7.10 along with results taken from other studies. It is observed that the proposed approach on the dataset including missing values outperforms all other methods considered. From these comparisons, it can be concluded that the proposed method is extremely competitive to other state-of-the-art studies published on real datasets.

7.4 Summary

This chapter addressed the thesis Objective 4 shown in Section 1.2. A method to construct classification models was presented based on multi-resolution hierarchical granular representations using hyperbox fuzzy sets. The proposed approach can maintain good classification accuracy at high abstraction levels with a low number of hyperboxes. The best classifier on the validation set usually produces the best predictive results on unseen data as well. One of the interesting characteristics of the proposed method is the capability of handling missing values without the need for missing values imputation. This property makes it flexible for real-world applications, where the data incompleteness usually occurs. In general, the proposed method outperformed other typical types of fuzzy min-max neural networks using the contraction process for dealing with overlapping regions in terms of both accuracy and training time. Furthermore, the proposed technique can be scaled to large-sized datasets based on the parallel execution of the hyperbox building process at the highest level of granularity to form core hyperboxes from sample points rapidly. These hyperboxes are then refined at higher abstraction levels to reduce the complexity and maintain consistent predictive performance.

Chapter 8

Ensemble Learning using Hyperboxed-based Classifiers

The learning algorithms proposed from Chapters 4 to 7 focus on building effective single predictive models. However, as shown in the literature, ensembles usually achieve better performance on classification problems than a single classifier (Zhang and Ma 2012). Another reason that ensemble models are commonly used in practice over a single model is their robustness in terms of the mean performance, because the ensemble models may reduce the dispersion of the predictive outcomes on different parts of training data in comparison to the use of a single learning model (Hastie et al. 2009). From these benefits, it is desired to build an effective ensemble model based on single hyperbox-based models. Therefore, this chapter presents a simple yet powerful ensemble classifier, called Random Hyperboxes, constructed from individual hyperbox-based classifiers trained on the random subsets of sample and feature spaces of the training set. A generalization error bound of the proposed classifier is also introduced based on the strength of the individual hyperbox-based classifiers as well as the correlation among them. The effectiveness of the proposed classifier is analyzed using a carefully selected illustrative example and compared empirically with other popular single and ensemble classifiers via 20 datasets using statistical testing methods. Finally, the existing issues related to the generalization error bounds of the real datasets are identified, and the potential research directions are informed. The main content of this chapter is taken from the following paper (Khuat and Gabrys 2021c):

- **Thanh Tung Khuat**, and Bogdan Gabrys, "Random hyperboxes," *IEEE Transactions on Neural Networks and Learning Systems* (Early Access), 2021, doi: 10.1109/TNNLS.2021.3104896.

8.1 Introduction

A random Hyperboxes (RH) classifier is an ensemble model containing many individual hyperbox-based learners, e.g., GFMMNNs, trained on random subsets of both instances and feature spaces. One of the key characteristics of hyperbox-based classifiers is the single-pass through the training data learning ability. Based on this incremental learning ability, new data and classes can be added to the model without retraining the whole network. Another interesting characteristic of hyperbox-based models is their interpretability thanks to the human understandable rule sets which can be extracted directly or indirectly from hyperboxes. Interpretability is one of the key requirements when applying machine learning algorithms to high-stakes applications such as medical diagnostics, financial investment, self-driving systems, and criminal justice (Rudin 2019).

The random hyperboxes model can be categorized into the family of ensemble classifiers, which build many base estimators and then combine them to create a final model. It is well-known that ensemble models are usually much more accurate than their base learners (Biau et al. 2008). There are two main methods to construct an ensemble model when using resampling methods and the same type of base learners. The first one aims to build many independent or low correlation individual estimators and combining their predictive outputs using majority voting or averaging approach. The representative models for this group include Bagging (Breiman 1996) and Random Forests (Breiman 2001). The second paradigm consists of algorithms building base estimators in a sequential manner, where the newly added learner tries to correct errors generated by previous classifiers. Adaptive boosting (Adaboost) (Freund and Schapire 1997) and Gradient Boosting Machines (Friedman 2001) are typical algorithms under the boosting framework. Extreme Gradient Boosting (XGBoost) (Chen and Guestrin 2016) and LightGBM (Ke et al. 2017) are two recent effective and scalable implementations of the gradient boosting algorithm.

The proposed random hyperboxes classifier belongs to the first group because it shares the same principle with the bagging, i.e., using individual hyperbox-based

learners with low correlation and combining their outputs by the majority voting. As shown in a recent survey on hyperbox-based machine learning algorithms (Khuat et al. 2021b) and in Chapter 2, there has been only one study (Gabrys 2002b) related to the use of bagging techniques with hyperbox-based models as base learners and another one which has been concerned with method independent learning approaches for constructing either ensembles or individual hyperbox-based classifiers (Gabrys 2004). In their work, after training individual hyperbox-based estimators on different subsets of the training sets, the resulting base learners are combined at the decision level using the majority voting or averaging of membership values or combined at the model level into a single model. However, as it has been frequently shown resampling methods used with bagging like algorithms operating only in the sample space can generate a limited level of diversity amongst the base classifiers trained in this way. As the diversity amongst the base learners is of key importance (Ruta and Gabrys 2005), there is another mechanism needed for making the resulting ensembles more effective and well performing. Based on Lemma 8.1 (the proof of this lemma can be found in the Appendix F), adapted from (Hastie et al. 2009), it can be seen that the high correlation between base learners leads to a high testing error for the average classifier. To cope with this problem, the correlation should be lowered but without significantly increasing the variance σ^2 of individual hyperbox-based learners by using only a subset of features when building base estimators. This fact can be achieved by utilizing feature subsets selected randomly for training each base classifier besides the subsets of samples. The use of a subsampling technique for both sample and feature spaces to construct the ensemble model constitutes the core principle of the random hyperboxes classifier. From surveys on hyperbox-based machine learning algorithms (Khuat et al. 2021b) and fuzzy min-max neural networks (Al Sayaydeh et al. 2019), it can be observed that this study is the first work using randomized hyperbox estimators trained on subsets of both samples and features to construct an ensemble model.

Lemma 8.1. *Given M identically distributed random variables (not necessarily independent) with the variance of each variable σ^2 and positive pairwise correlation ρ, the variance of the average random variable is:*

$$\rho \cdot \sigma^2 + \frac{1-\rho}{M} \cdot \sigma^2 \tag{8.1}$$

The use of subsets of features in building classifiers results in many effective models such as randomized trees on geometric feature selection (Amit and Geman 1997), the random subspace-based decision forests (Tin Kam Ho 1998), and random forests (Breiman 2001). Recently, there have been several studies focusing on employing random projections of the feature vectors into a lower-dimensional space to form training data for classifiers such as Fisher's linear discriminant (Durrant and Kabán 2015), random projection neural network (Andras 2018), or a general framework of random-projection based ensemble models (Cannings and Samworth 2017). These results have provided further motivation for the proposed random hyperboxes classifier.

One of the interesting characteristics of the proposed classifier is that it is easy to scale with large-sized training sets because each base learner can be constructed independently, so the learning process may be parallelized easily. The contributions in this chapter can be summarized as follows:

- A new ensemble classifier built from individual hyperbox-based learners is proposed using random subsets of both sample and feature spaces.
- A generalization error bound of the RH classifier is derived based on the strength and correlation between base learners.
- The effectiveness of the RH classifier has been analyzed in comparison to its base learners concerning the decrease in the variance of the ensemble model and the increase in the accuracy. Extensive experiments have been conducted on 20 datasets to compare the performance of the proposed method to other FMNNs as well as popular single and ensemble classifiers.
- The generalization error bounds on the real datasets are discussed and the open research directions are informed.

8.2 Random Hyperboxes Model

8.2.1 Formal Description

Let us denote by $\mathcal{T}_N = \{(X_i, c_i)\}_{i=1}^N$ a training data where $X_i \in \mathbf{X} \subset \mathbb{R}^n$ is a n-dimensional vector of observations (i.e. features) and $c_i \in \mathcal{C}$, $\mathcal{C}$ is a set of categorical variables denoting classes to which the observations fall. Given an input X, the main goal is to build an ensemble classifier which predicts class c from X using the training data $\mathcal{T}_N$.

Please note that for the theoretical considerations of the proposed algorithm covered in this section and the discussion of the convergence properties and the derivation of generalisation error bounds presented in Section 8.2.3, an assumption is made that the observations are independent and identically distributed (i.i.d.) random variables.

A random hyperboxes model with M hyperbox-based learners is a classifier including a set of randomized base hyperbox models $h(X, \Phi_1), \ldots, h(X, \Phi_M)$, where $\Phi_1, \ldots, \Phi_M$ are i.i.d. random vectors of a randomizing vector Φ, independent conditionally on $\mathbf{X}, \mathcal{C}$, and $\mathcal{T}_N$. Each individual hyperbox-based learner $h(X, \Phi_i)$ is constructed using the training set $\mathcal{T}_N$ and a random vector Φ_i. Φ_i introduces the randomness to the building process of hyperbox-based learners including the determining of a subset $\mathcal{T}_{\Phi_i}$ of the full training data $\mathcal{T}_N$ as well as determining a subset of features X_{Φ_i} used. After a large number of hyperbox-based learners genereated, the random hyperboxes estimator takes the class with most votes among base learners as its predictive result. Formally, the definition of the random hyperboxes classifier can be stated as follows:

Definition 8.1. *A random hyperboxes model is a classifier including a set of hyperbox-based learners* $\{h(X, \Phi_i) : i = 1, \ldots, M\}$*, where* $\{\Phi_i\}$ *are independent and identically distributed random vectors of a model random vector* Φ *independent conditionally on sample space* $(\mathbf{X}, \mathcal{C})$ *and the training set* $\mathcal{T}_N$*. Each hyperbox-based learner gives a unit vote based on the class of the hyperbox with the maximum membership degree with respect to the input pattern X. The predictive result of the random hyperboxes*

model is the aggregation of predictive results from its base learners using a majority voting method.

In particular, the predictive class ($c_k \in \mathcal{C}$) with respect to input data X of a random hyperboxes classifier including M base learners (let $\Phi^{(M)} = \{\Phi_1, \ldots, \Phi_M\}$) can be shown as follows:

$$h(X, \Phi^{(M)}) = arg \max_{c_k \in \mathcal{C}} \frac{1}{M} \sum_{i=1}^{M} \mathbb{1}(h(X, \Phi_i) = c_k) \tag{8.2}$$

where $\mathbb{1}(\cdot)$ is the indicator function. According to the strong law of large numbers, when the number of base learners increases, it is almost surely to obtain $\lim_{M \to +\infty} h(X, \Phi^{(M)}) = \overline{h}(X, \Phi)$, where $\overline{h}(X, \Phi) = arg \max_{c_k \in \mathcal{C}} \mathbb{E}_{\Phi}[\mathbb{1}(h(X, \Phi) = c_k)]$ (Here $\mathbb{E}_{\Phi}$ denotes the expectation with regard to the random variable Φ).

Algorithm 8.1 Training algorithm of the Random hyperboxes

Input: training set $\mathcal{T}_N$, sampling rate for samples r_s, maximum number of used features m_f, number of base estimators M, maximum hyperbox size θ, sensitivity parameter γ

Output: A Random Hyperboxes model **H**

1: $i = 1; \mathbf{H} \leftarrow \varnothing$
2: **for** $i \leq m$ **do**
3: $\quad \mathcal{T}_l \leftarrow$ Perform subsampling on $\mathcal{T}_N$ with rate r_s
4: $\quad p \leftarrow$ Generate a uniform random number in the range of $[1, m_f]$
5: $\quad \mathcal{T}_l^{(p)} \leftarrow$ Random sampling p features of $\mathcal{T}_l$
6: $\quad h_i \leftarrow$ **IOL-GFMM**$(\mathcal{T}_l^{(p)}, \gamma, \theta)$
7: $\quad \mathbf{H} \leftarrow \mathbf{H} \cup h_i$
8: $\quad i = i + 1$
9: **end for**
10: **return H**

The basic steps of the building process of the random hyperboxes classifier are shown in Algorithm 8.1. Each random hyperbox-based learner $h(X, \Phi)$ is formed as follows. A subset $\mathcal{T}_l$ (*line 3*) including $l < N$ samples is randomly selected from the full training data $\mathcal{T}_N$ using subsampling method without replacement under weak assumptions $l \to 0$ and $r_s = l/N \to 0$ as $N \to \infty$. According to (Politis et al. 1999), under the weak convergence hypothesis, the sampling distributions of $\mathcal{T}_l$ and $\mathcal{T}_N$ should be close, and they will converge to the true unknown distribution of whole

sample space. After that, p ($1 \leq p \leq m_f \leq n$) features from n features of $\mathcal{T}_l$ (*line 4*) will be selected at uniformly random to form a training set $\mathcal{T}_l^{(p)}$ for $h(X, \Phi)$ (*line 5*), where m_f is the maximum features used for each base learner. There are many learning algorithms which could be used to train the base hyperbox-based classifier $h(X, \Phi)$ on $T_l^{(p)}$ (*line 6*). This study uses the IOL-GFMM as presented in Chapter 4 to build the base estimators. It is noted that the base model $h(X, \Phi)$ is trained on only p features of $\mathcal{T}_N$, so in the classification step, $h(X, \Phi)$ only makes prediction using the same p features with respect to the unseen sample X. The learning and classification steps for each base learner are kept the same as in the IOL-GFMM algorithm. The above process is repeated M times to build M base learners for the random hyperboxes model **H**.

8.2.2 Time Complexity

Based on Algorithm 8.1, it is easily observed that the time complexity of a random hyperboxes model depends mainly on the time complexity of the training process for each base learner. As discussed in Chapter 4, the time complexity of the IOL-GFMM algorithm trained on a dataset containing N samples with n features is $\mathcal{O}(N \cdot \overline{\mathcal{K}} \cdot \overline{\mathcal{R}} \cdot n)$, where $\overline{\mathcal{K}}$ is the average number of expandable hyperbox candidates and $\overline{\mathcal{R}}$ is the average number of hyperboxes representing classes different from the input pattern class for each iteration in the training process. For the random hyperboxes model, each base learner is trained on only $l < N$ samples with the maximum $m_f < n$ features. Therefore, the time complexity of each base learner in the worst case is $\mathcal{O}(l \cdot \overline{\mathcal{K}} \cdot \overline{\mathcal{R}} \cdot m_f)$. It is required to build M base learners for a random hyperboxes classifier. As a result, if the base learners are sequentially constructed, the time complexity of training a random hyperboxes model in the worst case is $\mathcal{O}(M \cdot l \cdot \overline{\mathcal{K}} \cdot \overline{\mathcal{R}} \cdot m_f)$.

8.2.3 Properties of the Random Hyperboxes

The Convergence of the Random Hyperboxes Model

Let X be a random sample, drawn from the sample space, to be classified with true class c. Let $\mathcal{T}_N$ be a random training set drawn i.i.d. from the true distribution

of sample space $(\mathbf{X}, \mathcal{C})$. Given an ensemble of M base learners $h_1(X), \ldots, h_M(X)$, where $h_i(X) \equiv h(X, \Phi_i)$, a margin function of a random hyperboxes model with M base estimators for an input sample X can be defined as Eq. (8.3):

$$\mathcal{M}(X, c) = \frac{1}{M}\sum_{i=1}^{M} \mathbb{1}(h_i(X) = c) - \max_{j \neq c} \frac{1}{M}\sum_{i=1}^{M} \mathbb{1}(h_i(X) = j) \tag{8.3}$$

where $\mathbb{1}(\cdot)$ is the indicator function.

Remark 8.1. *The margin can be considered as a confidence measure with respect to the classification result of the random hyperboxes model. A large margin increases the confidence in predictive results for observations and vice versa.*

Based on the above margin function, the generation error of the random hyperboxes model is defined as follows:

Definition 8.2. *The generalization error is the probability* $\mathbf{P}_{\mathbf{X},\mathcal{C}}$ *measured in the sample space* $(\mathbf{X}, \mathcal{C})$ *that gives a negative margin:* $\mathcal{E} = \mathbf{P}_{\mathbf{X},\mathcal{C}}(\mathcal{M}(X, c) < 0)$

Lemma 8.2. *When the number of base estimators increases* $(M \to \infty)$ *and base estimators are independent, for almost surely all i.i.d. random vectors* $\Phi_1, \Phi_2, \ldots$*, the margin function for a random hyperboxes model* $\mathcal{M}(X, c)$ *at each input* X *converges to:*

$$\mathcal{M}^*(X, c) = \mathbf{P}_\Phi(h(X, \Phi) = c) - \max_{j \neq c} \mathbf{P}_\Phi(h(X, \Phi) = j) \tag{8.4}$$

The proof of Lemma 8.2 can be found in the Appendix F.2. From Definition 8.2 and Lemma 8.2, the following theorem for the convergence of generalization error can be achieved:

Theorem 8.1. *When the number of base learners increases* $(M \to \infty)$*, for almost surely all random vectors* $\Phi_1, \Phi_2, \ldots$*, the generalization error* $\mathcal{E}$ *converges to:* $\mathcal{E}^* = \mathbf{P}_{\mathbf{X},\mathcal{C}}\left[\mathcal{M}^*(X, c) < 0\right]$

This theorem explains that the random hyperboxes model does not overfit when more base learners are added to the model if hyperbox-based learners are independent and under the i.i.d. assumption. In the next subsection, the upper bound of the generalization error will be derived.

Generalization Error Bound

Based on Lemma 8.1, it can be observed that to decrease the variance of the average classifier, it is necessary to reduce the correlation of base learners. However, if the correlation decreases, the variance of base learners usually increases, and it makes the reduction of the prediction error harder. The correlation among base learners can be easily decreased by increasing base models' randomness. However, in this way the variance of the base learners will also be increased. Therefore, it is expected not to let the variance increase too fast. To cope with this issue, the change in the generalization error bound can be inspected and monitored.

Instead of having a fixed number of base estimators M, assuming that a fixed probability distribution is given for the random vector Φ from which base models are constructed. Similarly to random forests (Breiman 2001), the strength of the random hyperbox model can be defined based on the limit of the margin function as follows:

Definition 8.3. *The strength of the random hyperboxes model is defined as:*

$$\mathcal{S} = \mathbb{E}_{\mathbf{X},\mathcal{C}} \mathcal{M}^*(X, c) \tag{8.5}$$

where $\mathbb{E}_{\mathbf{X},\mathcal{C}}$ is the expectation through the $(\mathbf{X}, \mathcal{C})$ space. Strength can be considered as a fitness measure representing how accurate, on average, the individual hyperbox-based estimators generated from the model's random vector Φ are.

Assuming that $\mathcal{S} > 0$, according to Chebyshev's inequality, one has:

$$\begin{aligned} \mathcal{E}^* &= \mathbf{P}_{\mathbf{X},\mathcal{C}} \left[\mathcal{M}^*(X, c) < 0 \right] \leq \mathbf{P}_{\mathbf{X},\mathcal{C}} \left[\mathcal{S} - \mathcal{M}^*(X, c) \geq \mathcal{S} \right] \\ &= \mathbf{P}_{\mathbf{X},\mathcal{C}} \left[|\mathcal{M}^*(X, c) - \mathcal{S}| \geq \mathcal{S} \right] \leq \frac{\mathtt{Var}_{\mathbf{X},\mathcal{C}}(\mathcal{M}^*(X, c))}{\mathcal{S}^2} \end{aligned} \tag{8.6}$$

This is a weak upper bound of the generalization error, and it indicates that the prediction error is always lower than an explicit but unknown limit. The value of $\mathcal{S}$

can be estimated over the training set $\mathcal{T}_N$ as follows:

$$\begin{aligned}\overline{\mathcal{S}} &= \frac{1}{N}\sum_{i=1}^{N}\mathcal{M}(X_i, c_i) \\ &= \frac{1}{NM}\sum_{i=1}^{N}\Big(\sum_{k=1}^{M}\mathbb{1}(h_k(X_i) = c_i) - \max_{j\neq c_i}\sum_{k=1}^{M}\mathbb{1}(h_k(X_i) = j)\Big)\end{aligned} \tag{8.7}$$

Let $J(X, c) = arg\max_{j\neq c}\mathbf{P}_\Phi(h(X, \Phi) = j)$ be the class j leading to the most incorrect classification of base learners with respect to the input X. Then, a raw margin function can be defined for each base learner at each input X as follows:

Definition 8.4. *The raw margin function is defined by:*

$$\mathcal{R}(\Phi) = \mathcal{R}(X, c, \Phi) = \mathbb{1}(h(X, \Phi) = c) - \mathbb{1}(h(X, \Phi) = J(X, c)) \tag{8.8}$$

Following from the above definition,

$$\begin{aligned}\mathcal{M}^*(X, c) &= \mathbf{P}_\Phi(h(X, \Phi) = c) - \mathbf{P}_\Phi(h(X, \Phi) = J(X, c)) \\ &= \mathbb{E}_\Phi\left[\mathbb{1}(h(X, \Phi) = c) - \mathbb{1}(h(X, \Phi) = J(X, c))\right] \\ &= \mathbb{E}_\Phi\mathcal{R}(\Phi)\end{aligned} \tag{8.9}$$

It means that the limit of the margin values is the expectation of raw margin values computed over all realizations of Φ.

From the above raw margin function, the correlation between two hyperbox-based learners $h(X, \Phi_i)$ and $h(X, \Phi_j)$ generated from two i.i.d. random vectors Φ_i and Φ_j can now be defined as follows:

Definition 8.5. *The correlation between two hyperbox-based learners $h(X, \Phi_i)$ and $h(X, \Phi_j)$ of a random hyperboxes model can be calculated from the raw margin function through all observations as follows:*

$$\rho_{\mathbf{X},\mathcal{C}}(\Phi_i, \Phi_j) = \frac{\mathtt{Cov}_{\mathbf{X},\mathcal{C}}(\mathcal{R}(\Phi_i), \mathcal{R}(\Phi_j))}{\sigma_{\mathbf{X},\mathcal{C}}(\mathcal{R}(\Phi_i))\sigma_{\mathbf{X},\mathcal{C}}(\mathcal{R}(\Phi_j))} \tag{8.10}$$

where $\mathtt{Cov}$ is the covariance, $\sigma_{\mathbf{X},\mathcal{C}}(\mathcal{R}(\Phi_i))$ denotes the standard deviation of $\mathcal{R}(\Phi_i)$, holding Φ_i fixed, computed over observations.

Generally, the average correlation between base learners in the random hyper-boxes models is computed through all pairs of two i.i.d. random vectors Φ and Φ' as follows:

$$\overline{\rho} = \mathbb{E}_{\Phi,\Phi'}[\rho_{\mathbf{X},\mathcal{C}}(\Phi, \Phi')] \tag{8.11}$$

From the average correlation between base learners and the strength $\mathcal{S}$, the following theorem for the upper bound of the generalization error is given as below. The proof of Theorem 8.2 can be found in the Appendix F.3.

Theorem 8.2. *An upper bound of the generalization error for the random hyper-boxes model can be estimated from the strength of base learners and average correlation between base learners as follows:*

$$\mathcal{E}^* \leq \overline{\rho}\left(\frac{1}{\mathcal{S}^2} - 1\right) \tag{8.12}$$

8.3 Experimental Results

It is noted that the derivations and proofs in the previous section have been carried out under the i.i.d. assumption which in practice is difficult to verify and is very often not satisfied. In this section and the supplementary materials in the Appendix F, therefore, extensive bench-marking and experimental evaluation of the proposed method have been conducted to also verify its practical characteristics and performance. Experimental datasets used in this study were real-world multi-class problems and so they usually show the class imbalanced properties. As a result, the weighted-F1 measure has been used as a more suitable and less biased performance assessment measure than the often used classification accuracy. This measure was effectively used in many recent studies to assess the classification performance of predictive models on many practical multi-class imbalanced datasets (Ahmedt-Aristizabal et al. 2020; Shafaei et al. 2020; Bai et al. 2018; Canzanese et al. 2015). Weighted-F1 score is the average F1 score of each class weighted by the support which is the number of patterns of each class. Formally, the weighted-F1 score is

defined as Eq. (8.13):

$$\texttt{Weighted F}_1 = \sum_{i=1}^{|\mathcal{C}|} \frac{n_{c_i}}{n_t} \cdot \frac{2 \cdot \texttt{precision}_i \cdot \texttt{recall}_i}{\texttt{precision}_i + \texttt{recall}_i} \tag{8.13}$$

where $|\mathcal{C}|$ is the number of classes, n_t is the total number of testing samples, and n_{c_i} is the number of samples for the *i-th* class in the testing set.

8.3.1 Analyzing the Random Hyperboxes Classifier

The Decrease in the Variance Compared to Base Learners

To conduct this experiment, six datasets with diversity in the numbers of samples, features, and classes were used. All of the experimental results are shown in subsection F.4.1 in the Appendix F. This section only illustrates the results for a dataset of the one-hundred plant species leaves for margin (Mallah et al. 2013). This dataset includes 1600 samples with 64 features and 100 classes. Ten times repeated 4-fold cross-validation were performed to evaluate the ensemble model with 100 base learners. Therefore, there are 4000 base learners using the IOL-GFMM algorithm and 40 random hyperboxes models generated. The variance values in terms of weighted-F1 scores of base learners and the random hyperboxes models are shown in Figure 8.1. The variance values of other datasets are shown in Figure F.1 in the Appendix F. These results confirmed that the variance of random hyperboxes models using simple majority voting is significantly reduced compared to their base learners, so its classification accuracy is also higher than that of base estimators.

In this experiment, the maximum number of used features $m_f = 2\sqrt{n} = 16$ was set (for the *plant species leaves margin* dataset) and 50% of the training data samples were randomly selected to train each base learner. The probability of the number of features, p, used to build the 4000 base learners is shown in Figure F.2 in the Appendix F. The importance scores of features through all base learners can be identified using the used probability of each feature, as shown in Figure F.3.

Based on the probability that each feature is used in 4000 base learners, the contribution of the combination of features to the performance of each classifier can be determined. Therefore, a single model has been trained by the IOL-GFMM algo-

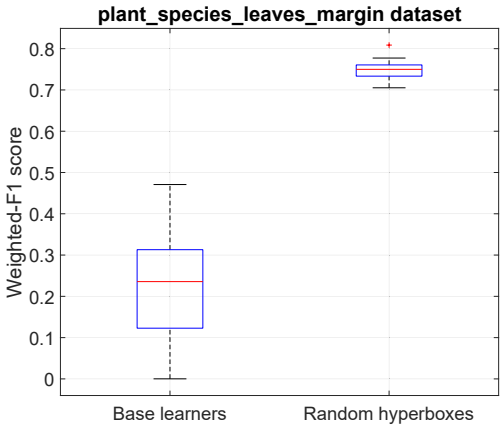


Figure 8.1 : The variances of RH models and their base learners (*plant species leaves margin* dataset).

rithm using top-K most used features ($K = 1, \ldots, n$) ($n = 64$ for the *plant species leaves margin* dataset) in each iteration. Figure 8.2 shows the average weighted-F1 scores for 40 testing folds (10 times repeated 4-fold cross-validation) for each top-K of the most often used features in the *plant species leaves margin* dataset. The results for the other datasets can be found in Figure F.4 in the Appendix F. It can be seen that the single model usually achieves the best performance if it is trained on all features. However, by using the random hyperboxes method with base learners trained on only a maximum of m_f features, a higher accuracy than the single model trained on all features can be obtained. Furthermore, in several datasets such as *ringnorm* and *connectionist bench sonar*, the best performance is often obtained when using a subset of the most crucial features. It is due to the fact that the redundant features can prevent the single GFMMNN from learning the true distribution of the underlying data with a given finite number of training samples. Therefore, the use of the random hyperboxes model of which base learners are trained on a subset of features can capture the data distribution more effectively and achieve better classification performance compared to the case of employing of a single GFMMNN.

In general, the RH classifier can achieve much better performance compared to the single GFMMNN using the IOL-GFMM algorithm with a full feature space,

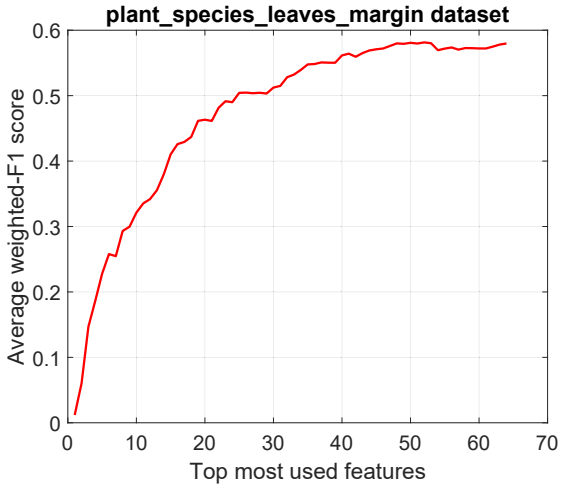


Figure 8.2 : Average weighted-F1 scores through 40 testing folds of a single model using training sets with top-k most used features (*plant species leaves margin* dataset).

especially for very high dimensional datasets. These results are shown in subsection F.4.2 in the Appendix F.

The Roles of the Number of Base Learners and Maximum Number of Used Features

This experiment is to assess the sensitivity of hyper-parameters such as the number of base learners and the maximum number of used features on the performance of the random hyperboxes model. Eight datasets with diversity in the numbers of samples, classes, and features were used for this purpose. All of the empirical results can be found in subsection F.4.3 in the Appendix F. This section only illustrates the outcomes of the same dataset used in subsection 8.3.1. To evaluate the impact of the number of base learners on the performance of the random hyperboxes model, the maximum number of used features $m_f = 2 \cdot \sqrt{n}$ ($m_f = 16$ in this case) and the maximum hyperbox size of each base learner $\theta = 0.1$ were kept unchanged, while 50% of samples were randomly selected to train each base estimator. The number of base learners is set from 5 to 200 with a step of 5. Figure 8.3 shows the average weighted-F1 scores over 10 times repeated 4-fold cross-validation at each threshold for the *plant species leaves margin* dataset. The results for the other datasets can

be found in Figure F.7 in the Appendix F. It can be observed that the performance of the random hyperboxes classifier is not reduced as more base learners are added. These figures confirmed that the random hyperboxes classifier does not overfit when adding more base learners.

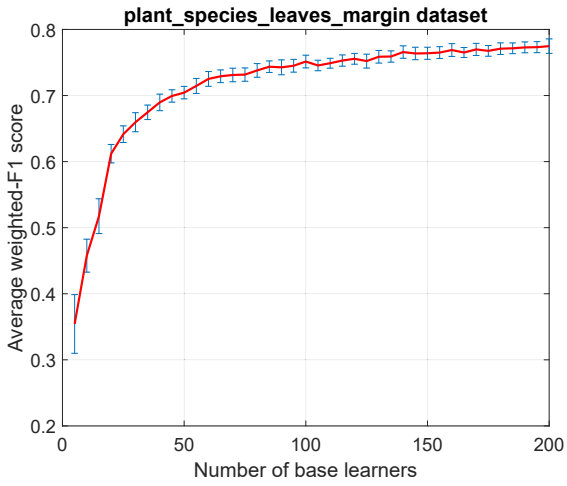


Figure 8.3 : The change in the average weighted-F1 scores when increasing the number of base learners (*plant species leaves margin* dataset).

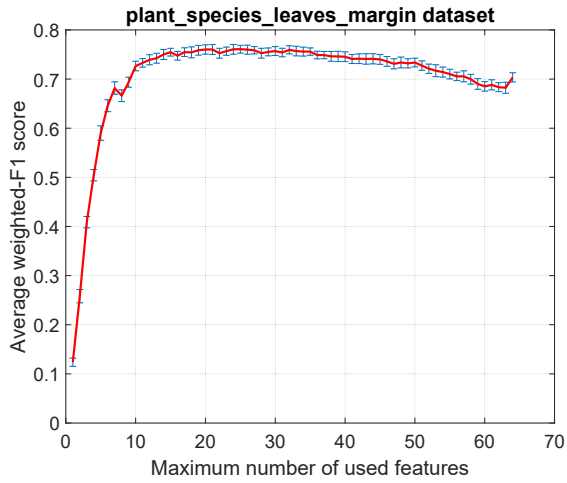


Figure 8.4 : The change in the average weighted-F1 scores when increasing the maximum number of used dimensions (*plant species leaves margin* dataset).

To assess the influence of the maximum number of used features m_f, the number of base learners $M = 100$, $\theta = 0.1$, $r_s = 0.5$ were kept unchanged, and the maximum

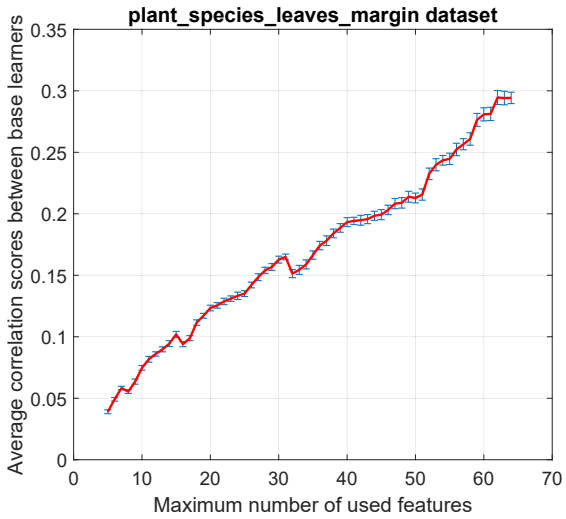


Figure 8.5 : Average correlation scores between base learners when increasing the maximum number of used dimensions (*plant species leaves margin* dataset).

numbers of used features from 1 to n ($n = 64$ in this case) were changed. Figure 8.4 depicts the average weighted-F1 scores for 10 times repeated 4-fold cross-validation at each value of the maximum number of used features for the *plant species leaves margin* dataset. The outcomes for the remaining datasets are shown in Figure F.8 in the Appendix F.

It can be easily observed that the overall trend when increasing the maximum number of used features is that the accuracy of the random hyperboxes classifier only increases to a certain threshold, and then its accuracy will decrease. It is due to the fact that the correlation between base learners will be higher when using too many features for each base learner. In contrast, if too few features are used, the strength of each base learner gets a low value, so the error of the ensemble model will increase. This fact confirms that the maximum number of used features is an important parameter, which needs to be carefully selected to achieve the high accuracy for the random hyperboxes classifier. To demonstrate the increase of the correlation between base learners when many features are used for each of the base learners, the average correlation scores were computed (using Eq. (8.11)) of all base learners generated from 10 times repeated 4-fold cross-validation on the *plant*

species leaves margin dataset and based on the testing prediction results. In this experiment, all training samples were used to train base learners aiming to eliminate the impact of the numbers of used samples on the correlation score. Therefore, the correlation scores, in this case, are only impacted by the maximum number of used features. Figure 8.5 shows the average correlation scores for 100 base learners trained and evaluated within 40 cross-validatory iterations.

The Impact of the Maximum Hyperbox Size Parameter

The experimental results in Chapters 3 and 4 indicated that the GFMMNN's classification performance usually decreases when increasing the values of the maximum hyperbox size (θ). Therefore, in this section, the impact of θ is assessed on the classification performance of the RH models. The number of base learners $M = 100$, $m_f = 2\sqrt{n}$, $r_s = 0.5$ were kept unchanged, while the values of θ were changed from 0.1 to 0.7 with a step of 0.1. This experiment was conducted on eight datasets with a diversity in numbers of samples, features, and classes.

Figure 8.6 describes the average weighted-F1 scores over 10 times repeated 4-fold cross-validation at each threshold of θ for the *plant_species_leaves_margin* dataset. The results for the other datasets can be found in Figure F.9 in the Appendix F. In most of the datasets (six out of eight datasets), the classification performance of the RH models slightly decreased when increasing the values of θ, while the remaining two datasets (e.g., *plant_species_leaves_margin* in Fig. 8.6) only slightly increased in the classification performance.

Generally, it can be seen that the performance of the RH models is less impacted by the choice of the values of θ than other single hyperbox-based classifiers (as shown in subsection 8.3.2), and the difference in the classification performance between different values of θ is usually smaller than 5%. This relative insensitivity to the choice of hyper-parameter θ can be regarded as additional advantage of RH method.

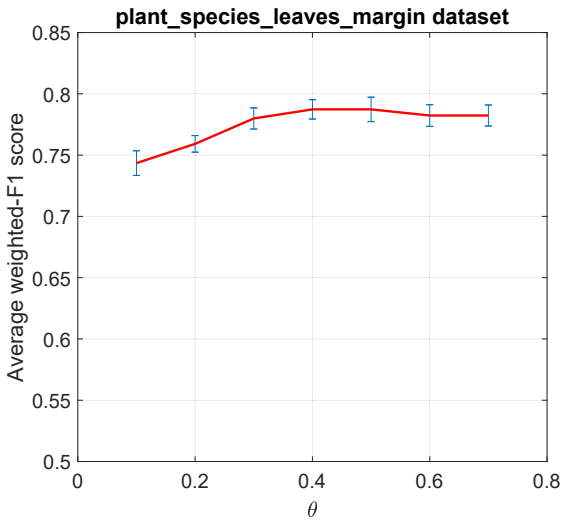


Figure 8.6 : The change in the average weighted-F1 scores when increasing the maximum hyperbox size (*plant_species_leaves_margin* dataset).

8.3.2 Comparing the Performance of the Random Hyperboxes to Other Classifiers

The datasets used and parameter settings for models are presented in subsection F.4.4 in the Appendix F. The random hyperboxes model is built based on a subset of features, and thus it is more appropriate for datasets with high dimensionality. Therefore, most of datasets used in this chapter have a high number of features and taken from the datasets employed in Chapter 6. The following results are the average weighted-F1 scores using 10 times repeated 4-fold cross-validation. In each iteration, three folds were used for training and hyper-parameter tuning (if used), and one remaining fold was used as a testing set.

A Comparison of the Random Hyperboxes With Other FMNNs

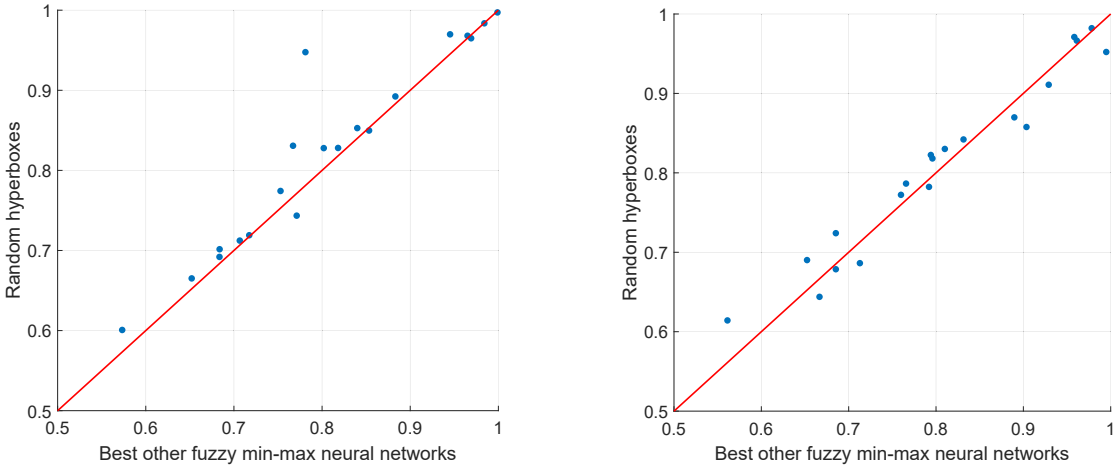
This experiment compares the RH model with FMNN (Simpson 1992), online learning version of GFMMNN (Onln-GFMM) (Gabrys and Bargiela 2000), agglomerative learning algorithm version 2 of GFMMNN (AGGLO-2) (Gabrys 2002a), combination of Onln-GFMM at $\theta = 0.05$ and AGGLO-2 (Gabrys 2002b), IOL-GFMM (Khuat et al. 2020), EFMNN (Mohammed and Lim 2015), KNEFMNN (Mohammed

and Lim 2017a), and RFMNN (Al Sayaydeh et al. 2020). The classification accuracy results of fuzzy min-max neural networks at low values of θ are usually better than those at high values of θ (as presented in Chapter 3). Therefore, in this experiment, the RH model will be compared with other FMNNs using $\theta = 0.1$ and $\theta = 0.7$. For the RH model, $m_f = 2\sqrt{n}$, $r_s = 0.5$, and $M = 100$ were set up. All of these fuzzy min-max neural networks have been implemented in Python.

The average weighted-F1 scores of classifiers using 10 times repeated 4-fold cross-validation are shown in Table 8.1 for the maximum hyperbox size $\theta = 0.1$ and Table 8.2 for $\theta = 0.7$. The best result for each dataset is highlighted in bold in the respective Tables. To facilitate the process of evaluating the performance and performing statistical testing, the performance of classifiers on each dataset is ranked with the best classifier with the highest average weighted-F1 score ranked first, and the next best performing classifier ranked second and so on. The classifiers with the same average weighted-F1 scores are assigned the average value of their ranks.

Figure 8.7 summarizes these results by comparing the results of the RH classifier with the best values of other FMNNs. It can be seen that in both subplots most points are located above the diagonal line. In addition, the random hyperboxes classifier usually obtains the highest average weighted-F1 scores on almost all considered datasets. These figures illustrate the efficiency and robustness of the random hyperboxes for both low and high thresholds of θ. It can also be seen that the random hyperboxes classifier achieves the best rank for both high and low values of θ. Its average ranks are nearly twice as low as those of the second-best classifiers. These figures show the superior performance of the RH classifier in comparison to other types of fuzzy min-max neural networks.

Using the Friedman rank-sum test (Friedman 1940), the F-distribution value $F_F = 8.0166$ can be computed from the average ranks of models at $\theta = 0.1$. Since the critical value of $F(8, 152)$ for the significance level $\epsilon = 0.05$ is 1.9998, the null hypothesis is rejected. It means that there are significant differences between the average weighted-F1 scores of these models. To further compare the peformance of the RH model to other FMNNs at $\theta = 0.1$, the Critical Difference (CD) diagram



(a) $\theta = 0.1$ (b) $\theta = 0.7$

Figure 8.7 : Comparison of average weighted-F1 scores of the random hyperboxes and the best value from single FMNNs.

with Bonferroni-Dunn test (Demsar 2006) for $\epsilon = 0.05$ is computed and shown in Figure 8.8.

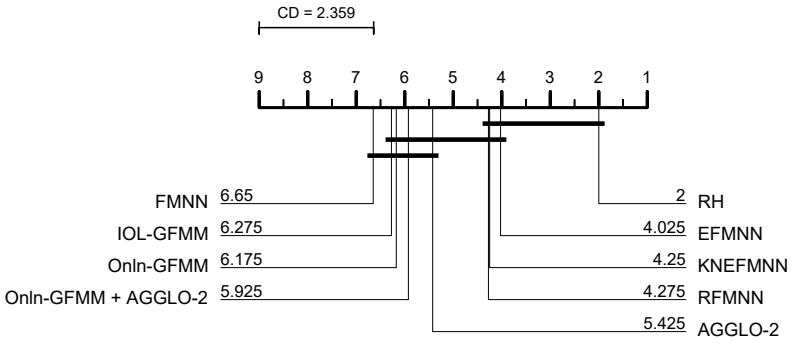


Figure 8.8 : Critical difference diagram for the performance of the RH classifier and other FMNNs ($\theta = 0.1$).

Similarly, with results of average ranks at $\theta = 0.7$, the F-distribution value can be calculated using the Friedman test $F_F = 17.4406 > F(8, 152) = 1.9998$. Therefore, there are significant differences among models using $\theta = 0.7$. By applying the Bonferroni-Dunn test, the CD diagram can be drawn as in Figure 8.9.

It can be seen that at the low value of θ, the RH classifier is significantly better

Table 8.1 : The average weighted-F1 scores of the random hyperboxes and other fuzzy min-max neural networks ($\theta = 0.1$)

ID	Dataset	RH	IOL-GFMM	Onln-GFMM	FMNN	EFMNN	KN EFMNN	RFMNN	AGGLO-2	Onln-GFMM + AGGLO-2
1	Balance scale	0.84976	0.85166	0.78643	0.75247	0.78619	0.78619	0.78619	**0.85332**	0.85166
2	banknote authentication	0.99723	0.99774	0.99774	0.99854	0.99891	0.99884	**0.99898**	0.99796	0.9976
3	blood transfusion	**0.71903**	0.7089	0.66383	0.68116	0.67464	0.66669	0.66774	0.71737	0.6728
4	breast cancer wisconsin	**0.96807**	0.94863	0.95227	0.96502	0.96281	0.96281	0.96281	0.94758	0.94863
5	Breast-Cancer-Coimbra	**0.69199**	0.68361	0.6722	0.64681	0.66408	0.66408	0.66408	0.6722	0.68361
6	connectionist bench sonar	**0.8528**	0.79679	0.79725	0.81323	0.83993	0.83993	0.83993	0.79725	0.79679
7	haberman	**0.66515**	0.64908	0.63228	0.62469	0.64068	0.63899	0.64002	0.65031	0.65209
8	heart	**0.82781**	0.73711	0.76455	0.80191	0.78621	0.78621	0.78621	0.75117	0.73711
9	movement libras	**0.82799**	0.81226	0.8152	0.80345	0.81816	0.8164	0.81816	0.8152	0.81251
10	pima diabetes	**0.71234**	0.6999	0.69686	0.67605	0.70184	0.70664	0.70184	0.69394	0.6989
11	plant species leaves margin	0.74348	0.58294	0.58413	0.69625	**0.7712**	**0.7712**	**0.7712**	0.57974	0.58294
12	plant species leaves shape	**0.60077**	0.5552	0.55773	0.5003	0.50534	0.53708	0.49769	0.57325	0.57101
13	ringnorm	**0.94761**	0.61643	0.61425	0.78111	0.6391	0.5809	0.6391	0.61594	0.61924
14	landsat satellite	**0.89232**	0.88035	0.88104	0.82841	0.87867	0.88315	0.87984	0.88145	0.88177
15	twonorm	**0.96986**	0.93642	0.93703	0.94191	0.94523	0.94523	0.94523	0.93703	0.93642
16	vehicle silhouettes	**0.70157**	0.66154	0.66417	0.66377	0.68376	0.67881	0.68376	0.66505	0.66419
17	vertebral column	**0.7743**	0.72024	0.74241	0.73968	0.74319	0.74631	0.74283	0.75287	0.72582
18	vowel	0.96492	0.96504	0.96333	0.95463	**0.96909**	0.96818	**0.96909**	0.96312	0.9655
19	waveform	**0.83075**	0.75629	0.75849	0.75236	0.76704	0.76704	0.76704	0.75838	0.75629
20	wireless indoor localization	0.98361	0.97906	0.97937	0.9779	0.9811	**0.9841**	0.981	0.97831	0.97831
Average rank		**2**	6.275	6.175	6.65	4.025	4.25	4.275	5.425	5.925

than Onln-GFMM, IOL-GFMM, FMNN, AGGLO-2, and Onln-GFMM + AGGLO2 in terms of the average weighted-F1 scores. However, its performance still has no significant difference compared to EFMNN, KNEFMNN, and RFMNN, although the average ranking of the RH classifier is lowest among nine fuzzy min-max models over

Table 8.2 : The average weighted-F1 scores of the random hyperboxes and other fuzzy min-max neural networks ($\theta = 0.7$)

ID	Dataset	RH	IOL-GFMM	Onln-GFMM	FMNN	EFMNN	KNEFMNN	RFMNN	AGGLO-2	Onln-GFMM + AGGLO-2
1	Balance scale	**0.84206**	0.72088	0.72088	0.65996	0.74475	0.71665	0.78225	0.83147	0.82594
2	banknote authentication	0.95216	0.7017	0.75738	0.84161	0.74829	0.76439	0.82741	**0.9949**	0.99351
3	blood transfusion	0.68629	**0.71283**	0.50747	0.51013	0.60178	0.59096	0.66797	0.70812	0.66711
4	breast cancer wisconsin	**0.96644**	0.96098	0.92804	0.91483	0.92646	0.9492	0.96119	0.95651	0.95757
5	Breast-Cancer-Coimbra	**0.72403**	0.61573	0.63197	0.55274	0.54407	0.64074	0.55574	0.6853	0.67133
6	connectionist bench sonar	**0.78649**	0.75466	0.73908	0.55492	0.61079	0.71767	0.60349	0.76568	0.76265
7	haberman	0.6439	**0.6666**	0.64321	0.65691	0.62859	0.62292	0.64803	0.64683	0.62664
8	heart	**0.81813**	0.76232	0.78471	0.69063	0.78213	0.7959	0.78248	0.76214	0.75531
9	movement libras	**0.82251**	0.73695	0.70277	0.56669	0.68259	0.7102	0.655	0.79195	0.79407
10	pima diabetes	0.67864	0.67542	0.64567	0.62277	0.61957	0.64848	0.66615	0.67683	**0.6853**
11	plant species leaves margin	0.78233	0.643	0.648	**0.79197**	0.78218	0.78603	0.68474	0.6291	0.63016
12	plant species leaves shape	**0.6142**	0.51384	0.44816	0.42757	0.43559	0.43559	0.44598	0.55561	0.56121
13	ringnorm	0.85772	0.77197	0.75873	0.81912	0.60252	0.70688	0.7151	0.86597	**0.90358**
14	landsat satellite	0.86986	0.85073	0.68601	0.61078	0.54928	0.67686	0.77834	0.88479	**0.88965**
15	twonorm	**0.97102**	0.9153	0.78093	0.76907	0.81234	0.7598	0.74947	0.9583	0.95802
16	vehicle silhouettes	**0.69017**	0.65234	0.56617	0.30581	0.51839	0.53064	0.56136	0.65224	0.64578
17	vertebral column	**0.77232**	0.62315	0.7489	0.75078	0.74176	0.75121	0.71126	0.75619	0.75988
18	vowel	0.91093	0.86002	0.57575	0.44263	0.53349	0.5673	0.72508	0.92471	**0.92902**
19	waveform	**0.8301**	0.80398	0.74535	0.70421	0.71902	0.72766	0.54723	0.81008	0.80806
20	wireless indoor localization	**0.98213**	0.85241	0.92548	0.92129	0.84104	0.84739	0.84548	0.97823	0.97715
Average rank		**1.85**	4.55	5.85	7	7.275	6.125	5.95	3.05	3.35

20 considered datasets. With a high value of θ, the RH model is significantly better than KNEFMNN, IOL-GFMM, Onln-GFMM, RFMNN, EFMNN, and FMNN. In this case, however, there is no statistical difference in the performance among the RH model, Onln-GFMM + AGGLO2 and AGGLO-2, although the performance of

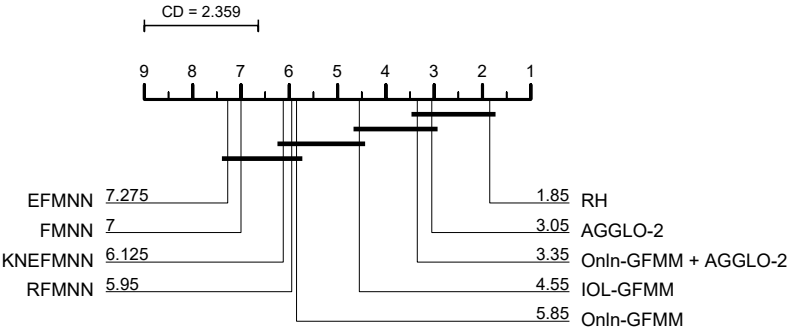


Figure 8.9 : Critical difference diagram for the performance of the RH classifier and other FMNNs ($\theta = 0.7$).

the RH classifier outperforms those of Onln-GFMM + AGGLO2 and AGGLO-2.

A Comparison of the Random Hyperboxes to Other Ensemble Classifiers

This experiment compares the perfomance of the random hyperboxes classifier (with and without hyperparameter tuning) to other prevalent ensemble models including Random Forest (Breiman 2001), Rotation Forest (Rodriguez et al. 2006), XGBoost (Chen and Guestrin 2016), LightGBM (Ke et al. 2017), Gradient Boosting (Friedman 2001), and ensemble of GFMMNNs using the IOL-GFMM algorithm at the decision level (Ens-IOL-GFMM (DL)) and at the model level (Ens-IOL-GFMM (ML)) (Gabrys 2002b). The hyperparameters of these models were tuned using the settings presented in subsection F.4.4. The Ens-IOL-GFMM (DL) model was formed by training many individual GFMMNNs using the IOL-GFMM algorithm with all of the training features. Then, the predictive results of this model are aggregation of predictions from all of its base learners using a majority voting mechanism. The Ens-IOL-GFMM (ML) model also trains many base learners with all features using the IOL-GFMM learning algorithm in the first step. Unlike the Ens-IOL-GFMM (DL) model, however, first for all the resulting hyperboxes from all of these base learners the undesired overlapping regions are eliminated before they are used as input patterns to build a single GFMMNN adopting the IOL-GFMM.

The average weighted-F1 scores of classifiers obtained from 10 times repeated

Table 8.3 : The average weighted-F1 scores of the random hyperbox model and other ensemble models

ID	Dataset	Tuned Random Forest	Tuned Rotation Forest	Tuned XG-Boost	Tuned Light-GBM	Tuned Gradient Boosting	Tuned Ens-IOL-GFMM (DL)	Tuned Ens-IOL-GFMM (ML)	Non-Tuned RH	Tuned RH
1	Balance scale	0.84657	0.8354	0.8734	**0.92132**	0.85399	0.85086	0.84168	0.84976	0.74308
2	banknote authentication	0.99111	0.99388	0.99636	0.99417	0.99432	**0.9984**	0.99767	0.99723	0.99213
3	blood transfusion	**0.75383**	0.74006	0.73982	0.7259	0.73938	0.74451	0.7176	0.71903	0.70569
4	breast cancer wisconsin	**0.96947**	0.96873	0.96484	0.96862	0.96802	0.9585	0.94801	0.96807	0.96455
5	Breast-Cancer-Coimbra	**0.71956**	0.68227	0.65512	0.6952	0.70325	0.67474	0.69184	0.69199	0.66304
6	connectionist bench sonar	0.78173	0.81073	0.81911	0.84303	0.80998	0.80103	0.8061	**0.8528**	0.83608
7	haberman	0.68526	0.666	0.69473	**0.69598**	0.66856	0.65809	0.65013	0.66515	0.66565
8	heart	0.82876	0.81862	0.83217	**0.83983**	0.8219	0.74656	0.75032	0.82781	0.79963
9	movement libras	0.76902	0.80929	0.74371	0.77936	0.70987	0.80231	0.82094	**0.82799**	0.82326
10	pima diabetes	**0.75671**	0.73857	0.73753	0.73702	0.73593	0.71999	0.69793	0.71234	0.70386
11	plant species leaves margin	0.72804	0.63063	0.80826	**0.81833**	0.55679	0.61404	0.58517	0.74348	0.78007
12	plant species leaves shape	0.52327	0.48445	0.56407	0.57754	0.4511	0.59155	0.58127	0.60077	**0.61721**
13	ringnorm	0.95029	0.92298	0.98059	**0.98219**	0.97848	0.61796	0.58931	0.94761	0.9688
14	landsat satellite	0.89042	0.89798	0.91841	**0.92102**	0.91814	0.88653	0.88049	0.89232	0.89756
15	twonorm	0.971	0.967	0.97265	0.97267	0.97351	0.96516	0.93608	0.96986	**0.97486**
16	vehicle silhouettes	0.73042	0.72433	0.75314	**0.75702**	0.75157	0.67201	0.66082	0.70157	0.70334
17	vertebral column	**0.83331**	0.78491	0.8123	0.82625	0.82524	0.75397	0.74838	0.7743	0.74919
18	vowel	0.90091	0.91476	0.91602	0.92655	0.94053	0.95669	0.96234	0.96492	**0.96633**
19	waveform	0.85043	0.85038	0.85297	**0.85667**	0.85461	0.80326	0.75615	0.83075	0.8403
20	wireless indoor localization	0.98281	0.97651	**0.9837**	0.98289	0.9828	0.98243	0.9799	0.98361	0.9816
Average rank		4.6	5.6	3.85	**2.85**	4.65	6.25	7.2	4.55	5.45

4-fold cross-validation and their ranking are given in Table 8.3. It can be observed that the average performance of the RH without hyperparameter tuning is much better than the results of the tuned Rotation Forest and the ensemble models of IOL-GFMM learners using all features in their training. It is also slightly better than the

tuned Random Forest and the tuned Gradient Boosting, but the RH classifier cannot outperform the tuned XGBoost and LightGBM models on 20 considered datasets. In spite of using the same base learners and sampling method, the RH classifier is much better than the Ens-IOL-GFMM with decision and model combination levels. It is due to the fact that the random hyperboxes classifier uses only a subset of features to train each base learner. This method reduces the correlation between base learners, and so it leads to the reduction of the generalization error. These empirical results are consistent with the theoretical results presented in subsection 8.2.3. However, it is also noted that the correlation is linked with variance, so achieving a low correlation but high variance will not decrease the prediction error. In addition, when reducing correlation by using a smaller number of features, it will also increase the variance of each base learner. Therefore, to achieve the reduction of prediction error, the correlation between base learners has to decrease faster than the growth of the variance. This issue needs to be analyzed in more details in the future study, especially the relationship between the maximum number of used features and the number of base learners.

Based on their average rank for 20 datasets, Friedman rank-sum test can be applied to calculate the F-distribution value $F_F = 5.4609 > F(8, 152) = 1.9998$. Therefore, there are differences in the performance of classifiers. Using the Bonferroni-Dunn test, the CD diagram of the RH model and other ensemble classifiers is shown in Figure 8.10.

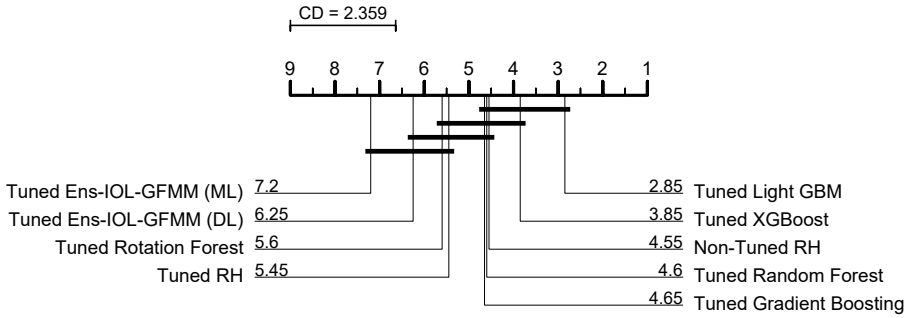


Figure 8.10 : Critical difference diagram for the performance of the RH classifier and other ensemble models.

Although the average rank of over 20 datasets of the RH model without hyperparameter tuning is higher than XGBoost and Light GBM, there are no significant differences in the weighted-F1 values among Random hyperboxes, XGBoost, LightGBM, Random Forest, and Gradient Boosting. In contrast, the RH classifier (without hyperparameter tuning) is statistically better than ensemble at the model level of IOL-GFMM base learners using full features and hyperparamenter tuning on 20 considered datasets. It can also be seen that the performance of the RH model without hyperparameter tuning outperforms that of the RH model using hyperparameter tuning. This result indicates that the RH model can achieve high performance without the need for tuning hyperparameters.

A Comparison of the Random Hyperboxes to Other Machine Learning Algorithms

This experiment compares the RH classifier (with and without hyperparameter tuning) to other popular machine learning algorithms including Decision Tree (DT) (Breiman et al. 1984), Naive Bayes (NB) (Zhang 2004), support vector machine (SVM) (Suykens and Vandewalle 1999), K-nearest neighbors (KNN) (Altman 1992), and Linear Discriminant Analysis (LDA) (Ye 2007). Apart from LDA and NB which do not have any hyperparameters, the remaining models were tuned using the settings shown in section F.4.4 in the Appendix F. Table 8.4 shows the average weighted-F1 scores and average rank of the random hyperboxes and other classifiers for the 20 datasets.

Using Friedman rank-sum test, the F-distribution value $F_F = 2.6833 > F(6, 114) = 2.1791$ can be obtained. Hence, there are statistical differences in the performance among classifiers. Similarly, using the Bonferroni-Dunn test, the CD diagram in this case is shown in Figure 8.11.

In this case, there is no statistically significant difference in the performance between the RH model using default parameters and other learning algorithms using hyperparameter tuning mechanisms on the considered datasets.

Table 8.4 : The average weighted-F1 scores of the random hyperboxes and other machine learning algorithms

ID	Dataset	Tuned Decision trees	Tuned SVM	Tuned KNN	LDA	Naive Bayes	Non-Tuned RH	Tuned RH
1	Balance scale	0.75988	**0.97229**	0.84967	0.83725	0.86568	0.84976	0.74308
2	banknote authentication	0.98193	0.98908	**0.99854**	0.97644	0.84001	0.99723	0.99213
3	blood transfusion	**0.75683**	0.71825	0.75453	0.70104	0.70773	0.71903	0.70569
4	breast cancer wisconsin	0.93955	0.95973	0.96679	0.9568	0.9594	**0.96807**	0.96455
5	BreastCancer-Coimbra	0.68505	0.683	0.66734	**0.69417**	0.60585	0.69199	0.66304
6	connectionist bench sonar	0.72193	**0.87293**	0.8236	0.74465	0.67663	0.8528	0.83608
7	haberman	0.68624	0.69068	0.66489	0.69172	**0.69973**	0.66515	0.66565
8	heart	0.78853	0.7936	0.80693	0.83762	**0.83939**	0.82781	0.79963
9	movement libras	0.635	**0.84433**	0.82906	0.60429	0.61467	0.82799	0.82326
10	pima diabetes	0.74638	0.73151	0.72352	**0.75864**	0.74742	0.71234	0.70386
11	plant species leaves margin	0.44808	**0.83169**	0.75339	0.79402	0.72753	0.74348	0.78007
12	plant species leaves shape	0.41721	**0.69639**	0.61528	0.48546	0.51983	0.60077	0.61721
13	ringnorm	0.88983	0.98099	0.72459	0.76902	**0.9867**	0.94761	0.9688
14	landsat satellite	0.85382	**0.91031**	0.90658	0.83205	0.80403	0.89232	0.89756
15	twonorm	0.84042	0.97524	0.97295	0.97735	**0.97819**	0.96986	0.97486
16	vehicle silhouettes	0.68481	**0.82559**	0.6881	0.77492	0.41933	0.70157	0.70334
17	vertebral column	0.80737	0.80516	0.76791	0.81346	**0.82186**	0.7743	0.74919
18	vowel	0.72827	0.93225	**0.97308**	0.58748	0.66367	0.96492	0.96633
19	waveform	0.7629	0.85382	0.84054	**0.85945**	0.79677	0.83075	0.8403
20	wireless indoor localization	0.9689	0.97921	0.98311	0.97161	0.98321	**0.98361**	0.9816
Average rank		5.3	**2.8**	3.75	4.1	4.25	3.6	4.2

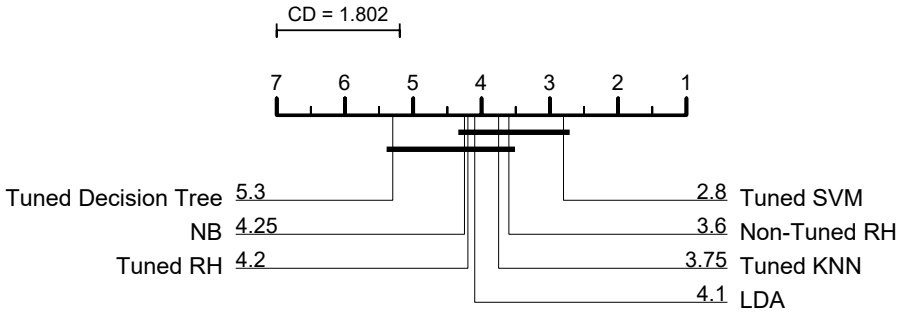


Figure 8.11 : Critical difference diagram for the performance of the RH classifier and other popular learning algorithms.

8.4 On the Estimation of Generalization Error Bounds and Open Problems

The upper generalization error bound of the random hyperboxes model is computed based on the i.i.d. assumption of samples in both training and testing sets. However, in practice, this assumption is usually violated for the real world datasets. This means that it is very difficult to obtain the training and testing sets which are representatives of a true distribution of the sample space. This section will estimate the upper generalization error bounds of datasets used for the experiments in section 8.3. The purpose of this section is to identify the effectiveness of the upper generalization error bound on real datasets and the existing problems when applying a strong assumption from the theoretical derivations to the practical issues. The upper bound values were estimated from the training set and 100 base learners trained by the IOL-GFMM algorithm with $\theta = 0.1$. The estimated results of the upper generalization error bound are the average values from 40 iterations (10 times repeated 4-fold cross-validation). To strengthen the comparison and conclusion, the upper generalization error bounds were also estimated from the base learners trained in turn on each of four folds generated by using the density preserving sampling (DPS) method (Budka and Gabrys 2013). The DPS method aims to preserve the data density and the classes shapes when splitting an original dataset into many folds, so it is possible to create the testing sets which are representatives for the training data. Hence, the testing errors on the DPS folds are usually smaller than those calculated from folds of the cross-validation method. This fact is confirmed with the results shown in Table 8.5. This table presents the real average testing errors of 4-DPS-fold cross-validation and 10 times repeated 4-fold cross-validation as well as their upper generalization error bounds estimated from corresponding training sets.

In general, there have been eleven datasets in which the estimated upper bounds are higher than real testing errors. Among them, there are a number of datasets with real errors close to the estimated upper bounds, such as *pima diabetes*, *banknote authentication*, *vowel*, and *twonorm*. One explanation for these good estimations is that the training sets and testing sets are good representatives of each other and

Table 8.5 : Estimated upper generalization error bounds (%), real testing error (%), and their standard deviations computed from different assessment methods

ID	Dataset	10 times repeated 4-fold cross-validation		4-DPS-fold cross-validation	
		Testing error (%)	Estimated upper error bound (%)	Testing error (%)	Estimated upper error bound (%)
1	Balance scale	12.224 ± 1.188	55.093 ± 3.694	11.839 ± 1.98	42.744 ± 2.242
2	banknote authentication	0.269 ± 0.259	2.448 ± 0.397	0.219 ± 0.279	2.251 ± 0.331
3	blood transfusion	22.915 ± 1.445	89.405 ± 6.018	21.39 ± 0.617	82.004 ± 1.409
4	breast cancer wisconsin	3.404 ± 1.312	13.014 ± 1.708	3.29 ± 1.265	12.638 ± 3.668
5	BreastCancerCoimbra	30.259 ± 8.319	10.048 ± 1.012	24.138 ± 5.631	10.278 ± 1.145
6	connectionist bench sonar	14.415 ± 3.501	7.076 ± 0.54	11.538 ± 2.72	7.579 ± 0.516
7	haberman	27.752 ± 3.223	56.919 ± 4.751	23.855 ± 1.214	51.372 ± 5.858
8	heart	17.475 ± 4.021	19.2 ± 1.987	17.395 ± 4.16	16.737 ± 1.852
9	movement libras	16.5 ± 3.545	11.477 ± 0.864	13.333 ± 3.741	14.421 ± 1.649
10	pima diabetes	26.38 ± 2.501	27.373 ± 2.02	23.698 ± 0.672	24.799 ± 0.6
11	plant species leaves margin	24.113 ± 1.778	12.479 ± 0.575	22.938 ± 3.098	11.565 ± 0.353
12	plant species leaves shape	37.606 ± 2.25	17.615 ± 0.444	34.5 ± 2.908	19.155 ± 0.769
13	ringnorm	5.191 ± 0.566	4.76 ± 0.134	5.824 ± 0.222	5.103 ± 0.104
14	landsat satellite	10.409 ± 0.678	17.224 ± 0.355	10.287 ± 0.507	16.95 ± 0.227
15	twonorm	2.972 ± 0.419	3.798 ± 0.064	2.919 ± 0.153	3.838 ± 0.118
16	vehicle silhouettes	28.238 ± 2.054	19.644 ± 0.982	27.181 ± 3.81	20.148 ± 1.029
17	vertebral column	21.32 ± 3.029	12.526 ± 0.702	19.68 ± 2.221	14.431 ± 1.493
18	vowel	3.545 ± 0.962	5.226 ± 0.231	2.022 ± 0.994	5.808 ± 0.244
19	waveform	16.322 ± 0.926	8.396 ± 0.172	15.98 ± 0.734	8.296 ± 0.088
20	wireless indoor localization	1.66 ± 0.486	10.102 ± 0.466	1.85 ± 0.252	9.827 ± 0.638

the whole sample space. It can be seen that, for these datasets, the real testing errors of 10 times repeated 4-fold cross-validation and 4-DPS-fold cross-validation are relatively close to each other.

In the nine remaining datasets, the estimated values of upper bounds are lower than the real testing errors when applying the 10 times repeated 4-fold cross-validation method. The same behavior but with a smaller error can be found with the 4-DPS-fold cross-validation method on nine datasets. Interestingly, there are two datasets, *heart* and *movement libras*, in which the estimated values are bad when using 10 times repeated 4-fold cross-validation, but very good estimated upper bounds can be obtained when deploying the 4-DPS-fold cross-validation. This fact indicates that if the representativeness of training sets with regard to the whole sample space is good, it can be achieved a much better estimation of the upper generalization error bounds which is close to the testing error on unseen data with the same distribution.

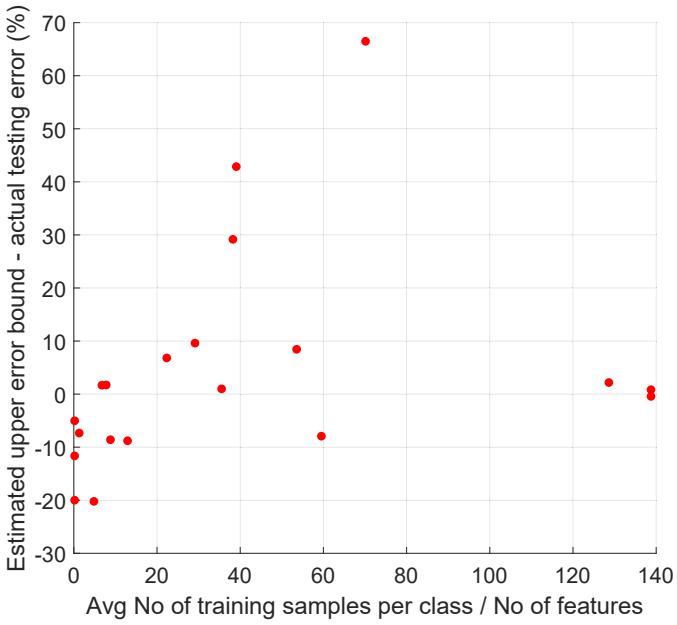


Figure 8.12 : The relationship of the difference in the estimated upper error bound and actual testing error with respect to the ratio of the average number of training samples per class and the number of features.

One general characteristic of datasets resulting in the poor estimated upper bounds is their sparsity with regard to a small number of samples and a relatively high number of dimensions. For these datasets, there are not sufficient number of samples to accurately enough capture the underlying distribution of the whole sample space. As a result, the base estimators overfit with their training data, and the estimated values of the upper error bounds are usually small. Meanwhile, the testing errors on unseen data are fairly high. Here, one open problem identified is the relationship between the number of samples, classes, and dimensions so that a good estimation of the generalization error bounds can be obtained from the training data. This is a critical issue that needs to be tackled in future work. As an example demonstration for this issue, Figure 8.12 shows the relationship of the difference in the estimated upper error bound and actual testing error to the ratio of the average training samples per class and the number of features for 20 datasets used in this experiment. It can be seen that a good estimation of the upper error bound can be obtained if the ratio of the average training samples per class and the number of

features is larger than 20. If this ratio is higher than 120, it is more likely to achieve an estimated upper error bound close to the actual testing error.

In summary, the i.i.d. assumption of training and testing sets is usually not met in practical datasets. Therefore, to reduce the classification error on unseen data, it is necessary to use several methods to guarantee the representativeness of the training and testing sets when assessing the performance of models. Moreover, identification of the relationship between the numbers of samples, classes, and features is crucial to building a representative training set.

One of the strong points of the general fuzzy min-max neural network is the interpretability. However, the significantly improved predictive accuracy of the proposed random hyperboxes method comes at a price of loss of interpretability as is common with other ensemble methods. As previously shown in Gabrys (2002b), hyperbox representation allows for combination at the model level rather than the decision level and therefore retaining the interpretability of the final model. Nonetheless, the combination of the individual hyperbox-based learners which are built from different random subspaces of features is not a trivial problem. Therefore, the future study should focus on building interpretable random hyperboxes models.

8.5 Summary

This chapter addressed the thesis Objective 5 presented in Section 1.2. A novel random hyperboxes classifier was proposed, and its properties were discussed and provided derivations of its generalization error bounds. The experimental results confirmed the efficiency of the proposed method in comparison to other single fuzzy min-max neural networks as well as single learning algorithms. The random hyperboxes model is also competitive with other popular ensemble methods. Furthermore, several discussions on the estimation of the upper generalization error bounds for real-world datasets were provided, and identified some open issues for future work.

Chapter 9

Conclusion and Future Work

9.1 Key Findings and Conclusions

The main purpose of this study is to construct robust, scalable, and interpretable learning algorithms using hyperbox representations. To address this aim, the following objectives were successfully achieved.

Evaluating the strong and weak points of the existing hyperbox-based learning algorithms. Chapter 2 presented the progression of hyperbox-based learning models from the first version of FMNN. Attractive characteristics of this class of learning algorithms were identified together with their drawbacks. Next, Chapter 3 described the empirical assessments regarding a particular learning model using hyperbox fuzzy sets, i.e., GFMMNN. The factors affecting the classification performance of the GFMMNN were analyzed in detail. The obtained results enabled to detect issues which needed to be addressed in the future studies.

Developing and evaluating new robust learning algorithms for the GFMMNN. Based on the analyses shown in Chapters 2 and 3, new robust learning algorithms for the GFMMNN were proposed in Chapters 4 and 5. A new online learning algorithm was introduced in Chapter 4 to address the drawbacks of the existing online learning algorithm related to the contraction process. The empirical results indicated the improvement in the classification accuracy and stability of the proposed method compared to the original version and other fuzzy min-max classifiers. However, all of the current online learning algorithms for the GFMMNN have not dealt effectively with mixed-attribute data. Therefore, Chapter 5 extended the IOL-GFMM learning algorithm proposed in Chapter 4 by using the changes in entropy values regarding the categorical values to determine whether the hyperbox can be expanded to include the input pattern. One of the interesting properties of the EIOL-GFMM algorithm

is that it does not use any encoding methods for categorical features. Empirical results confirmed the superior performance of the proposed method compared to other existing techniques in the literature for mixed-attribute data classification.

Developing different solutions to accelerate learning algorithms for the GFMMNN. One of the drawbacks of prototype-based learning algorithms is the long training time due to the continuous computation of distance/similarity values, especially in the datasets with high dimensionality. The learning algorithms of the GFMMNN have also faced the same issue. Therefore, a solution of reformulating the learning steps was proposed so that they can be executed on the GPUs in parallel. The operations on matrices were used to implement the learning procedures aiming to take benefits from the strength of GPUs. The empirical results on two very high-dimensional datasets indicated that the training and testing processes performed on the GPUs were from 10 to 35 times faster compared to those running serially on the CPU while retaining the same classification accuracy. Apart from using the GPUs to speed-up learning algorithms, another acceleration technique was given in Chapter 6. This solution relied on mathematical formulas to reduce the hyperbox candidates considered in the learning process, which certainly do not satisfy the expansion conditions. The experimental results indicated the significant decrease in training time of the proposed approach for both online and agglomerative learning algorithms. Notably, the training time of the online learning algorithms was reduced from 1.2 to 12 times when using the proposed method, while the agglomerative learning algorithms were accelerated from 7 to 37 times on average.

Increasing the interpretability of hyperbox-based learning models. Chapter 7 proposed the way of building high-performing classifiers with simple structures using multi-level granular information from the input data. The construction process of the proposed classifier comprises two phases. The first phase is to build a model at the lowest level of abstraction, while the later stage aims to reduce the complexity of the constructed model and deduce it from data at higher abstraction levels. The proposed learning algorithm can maintain a high accuracy at a low degree of granularity. Therefore, it can reduce the data size significantly and handle the uncertainty

and incompleteness associated with the input data. Experimental analyses carried out on both synthetic and real datasets illustrated the efficiency of the proposed method in terms of training time, simple structure, and predictive performance compared to other types of fuzzy min-max neural networks and common machine learning algorithms.

Building an effective ensemble model from hyperbox-based learners. Chapter 8 proposed a random hyperboxes classifier, which is built from many GFMMNN base learners trained on different subsets of both samples and features. The important mathematical properties of this model such as the convergence and generalization error bound were also proposed and proved. The effectiveness of the method and the sensitivity of its parameters were carefully analyzed across many examples. The experimental results confirmed that the proposed method outperformed other fuzzy min-max neural networks, popular learning algorithms, and is competitive with other ensemble methods.

9.2 Future Work

Although hyperbox-based machine learning algorithms have achieved remarkable results as presented in previous chapters in this thesis, there are still outstanding issues and room for the improvements. In addition to potential research directions presented in section 2.6.2, the following presents other research topics which can be performed as an extension of this thesis:

- **Building an interpretable GFMMNN for both numerical and categorical features**. Although the GFMMNN for mixed-attribute data presented in Chapter 5 can explain the predicted results using the membership function to select the appropriate hyperbox, to make it friendly and easy-to-read for users, it is necessary to extract and optimize *if-then* rule sets from the resulting hyperboxes for both continuous and discrete features in the future studies. The interpretability of predictive models is a critical factor when applying the machine learning algorithms for high-stakes applications such as medicine, finance, or criminal justice (Rudin 2019).

- **Dynamic rescaling and outlier detection**. The GFMMNN only works if the input samples are in the range of [0, 1] in each dimension. Hence, the input samples need to be normalized before training the GFMMNN. As a result, if the testing sample is outside of the range of the training data, then the value after normalizing would fall outside the required range of [0, 1] and the classifier does not work correctly. In the online learning process on the streaming data, it is also faced the same problem because the range of input features may be not known in advance and input samples are usually not normalized to the range of [0, 1]. A solution is to perform a dynamic rescaling of the ranges covered by the features as shown in Salvador et al. (2016). However, this method will be affected by the outliers when the current hyperboxes are normalized to accommodate the outliers. Therefore, it is necessary to build an effective outlier handling mechanism to decide whether the current model should be renormalized or not. Therefore, the construction of an online adaptive GFMMNN is an exciting field of research which should receive more attention in the future.

- **Construction of robust interpretable ensemble models using hyperbox representations**. One of the strong points of the GFMMNN is the interpretability of the trained model. However, this interpretability is lost when building ensemble models such as random hyperboxes. The use of hyperbox representations enables us to merge the hyperboxes generated from many base learners to form an interpretable single model. However, the relearning method using the AGGLO-2 algorithm to aggregate resulting hyperboxes from base estimators presented in Gabrys (2002b) only works when all base learners use the same number of features. Hence, future studies should pay more attention to expand this technique for base learners trained on subsets of features.

- **Expansion of theoretical analyses for random hyperboxes**. The relationship between correlation and variance between base learners as well as the trade-off between variance and bias of the random hyperboxes model need to be analyzed in more details. In addition, the influence of hyperparame-

ters of the random hyperboxes model should be assessed by a comparative study. In Chapter 8, it is assumed that the strength $\mathcal{S} > 0$ when analyzing the generalization error bound. In the case of highly imbalanced classes, this assumption may be false because the strength usually focuses on the majority class. Therefore, the efficiency of the random hyperboxes classifier and its theoretical results should be investigated and extended for imbalanced datasets.

- **Securing hyperbox-based machine learning algorithms**. When most businesses and governments are more and more dependent on machine learning algorithms for their decision making, there is an increase in attacking these algorithms to receive the expected results from hackers. Therefore, securing the machine learning algorithms in general and hyperbox-based learning algorithms, in particular, plays a crucial role. Many defense methods can be used to make the hyperbox-based learning models secure under adversarial attacks. An adversarial training mechanism (Goodfellow et al. 2015) can be employed, in which the hyperbox-based models are retrained on adversarial patterns created at each iteration based on the current state of the model. This technique may enforce invariance in outputs of the model given the original samples and their adversarial counterparts. Another method is to introduce randomness to the model against adversarial samples by pruning or dropping several hyperboxes during test time. The data density can also be estimated and integrated into the process of building hyperboxes to detect patterns located far away from a class manifold or outside of the data distribution. These are potential techniques to secure hyperbox-based learning models.

- **Real-time big data analytics**. In the era of digital information, a critical growing trend of machine learning research based on hyperbox representations is to discover and extract valuable knowledge from real-time big data. However, it is extremely challenging to analyze the real-world big data sets because of their exceedingly high volume and velocity. To cope with these problems effectively, more research efforts should focus on developing parallel or distributed hyperbox-based models to make use of the strengths of grid com-

puting and high performance computing systems. Ilager and Prasad (2017) have been the first ones who proposed the use of MapReduce mechanism to scale the original FMNN for large data sets. However, this research direction needs many enhancements and expansions to realize its full potential. In recent years, re-configurable hardware, for instance Field-Programmable Gate Arrays (FPGAs), has attracted much attention due to the programmable property, massive parallelism, and energy efficiency (Ghasemi and Chow 2017). The FPGAs have recently been applied to accelerate large-scale tasks and improve significant performance with considerable power savings, which suggests their potential for large-scale high-performance computations (Putnam et al. 2014). Hence, implementation of hyperbox-based machine learning algorithms on FPGAs is also interesting research direction in near future.

- **Automatic adaptation of models**. In order to discover information from real-time data, machine learning models need to adapt with the changes of data. Therefore, significant research efforts should be put into this problem to formulate adaptive hyperbox-based predictors.

- **Streaming data mining**. In the practical applications, the data may change constantly over time, so hyperboxes need an adaptive ability. It is essential to find a way of re-sizing or evolving the previously learned hyperboxes or ignoring these hyperboxes if they are no longer appropriate for the new situations (Mirzamomen and Kangavari 2017). Another problem for streaming data is the case where the target concepts to be predicted are likely to change over time in unanticipated ways, which is called the "concept drift" problem. In general, there are still no formal hyperbox-based techniques for dealing with all kinds of issues, especially in the situations that target concepts change constantly in arbitrary ways.

- **Automatic optimization of hyperparameters and parameters.** It has been frequently observed that the performance of current hyperbox-based models depends on various user definable hyperparameters and parameters. Therefore, it is necessary to investigate the importance of each parameter to

the performance of algorithms. Studies also need to find the optimal configurations for parameters or adapt the values of parameters according to the features of data sets in an automated manner.

- **Learning from multiple data sources.** All current hyperbox based machine learning algorithms have learned from only single source-structured data. In the real applications, data can come from many sources in many different formats (structured, unstructured, or semi-structured data). Therefore, future research should handle these challenges by building multi-source hyperbox-based learning algorithms.

- **Improvement of membership functions.** The classification performance of current hyperbox-based classifiers significantly depends on the membership function. Therefore, one can propose a novel membership function based on the features of data and the influence of noise for both numerical and categorical data. Current membership function for both numerical and categorical data proposed by Castillo and Cardenosa (2012) and Shinde and Kulkarni (2016) did not take the features of data and noise into account. In Chapter 5, a novel membership function for mixed-attribute data was proposed. However, this membership function still does not take noisy data into account. Another problem of current membership functions is that they do not yet classify data in the overlapping regions among hyperboxes belonging to different classes effectively. Certainly, current studies have not yet fully handled all of these problems.

- **Data editing.** The data quality is one of the factors influencing the accuracy of final classification results. In terms of real-time data analysis, the input data may contain noise or missing values or incomplete data. The use of preprocessing methods might not be sufficient to enhance the quality of input data. The ways of dealing with missing values using hyperbox fuzzy sets proposed by Gabrys (2002c) for numerical data and Castillo and Cardenosa (2012) for categorical data are efficient techniques to handle missing or incomplete data. Therefore, many research efforts should concentrate on enhancing the robust-

ness of hyperbox-based learning algorithms when resolving the poor quality of real data.

- **Extraction of linguistic fuzzy rules**. As discussed in Section 7.2.4, it is easy to track the hyperbox used to generate the predicted class for each input pattern. This hyperbox together with its membership value can be utilized to help users to better understanding the learning models. However, when the number of generated hyperboxes significantly increase in the case of large number of input samples, the transparency and traceability of hyperbox-based models can reduce overtime. It is highly desirable that the learned knowledge of models can be extracted and shown a compact set of linguistically interpretable fuzzy rules. By this way, the interpretablity of the learning models may dramatically increase and the users can interact with learning systems easier. However, existing learning algorithms for hyperbox-based models have not yet possessed the ability to extract and represent such linguistic rules. This is an interesting research direction for the future studies.

9.3 Final Summary

Nowadays, there is a high demand for interpretable models to substitute black-box models in assisting decision-makers in areas with the requirement of high safety and trust such as finance, health-care, and criminal justice. Using hyperboxes as fundamental building blocks and representations can help to formulate interpretable learning models. In addition, hyperbox-based learning model can be trained in incremental ways. This is an attractive characteristic of this type of learning algorithms because it does not require retraining the models periodically. As a result, they can be deployed for applications working in a dynamically changing environment with a high volume of data.

This study has been formed on top of principles of constructing robust classifiers with good generalization using hyperboxes as foundational building blocks. The obtained results in this thesis can form foundations to develop smart adaptive systems in the near future. This thesis presented fundamental information on formation and

development of hyperbox-based learning algorithms. Building on the existing studies, this thesis also proposed new robust learning algorithms for both single models and ensemble models to address the limitations of existing learning algorithms. In addition, different methods were offered to accelerate current learning algorithms on large-sized datasets. The obtained results from this study have led to the following list of external publications:

1. Thanh Tung Khuat, and Bogdan Gabrys, "Random hyperboxes," *IEEE Transactions on Neural Networks and Learning Systems* (Early Access).
   DOI: 10.1109/TNNLS.2021.3104896
2. Thanh Tung Khuat, and Bogdan Gabrys, "An in-depth comparison of methods handling mixed-attribute data for general fuzzy min-max neural network," *Neurocomputing*, vol. 464, pp. 175-202, 2021.
3. Thanh Tung Khuat, Fang Chen, and Bogdan Gabrys, "An effective multi-resolution hierarchical granular representation based classifier using general fuzzy min-max neural network," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 2, pp. 427- 441, 2021 (**IEEE CIM Publication Spotlight paper**).
4. Thanh Tung Khuat, and Bogdan Gabrys, "Accelerated learning algorithms of general fuzzy min-max neural network using a novel hyperbox selection rule," *Information Sciences*, vol. 547, pp. 887-909, 2021.
5. Thanh Tung Khuat, Dymitr Ruta, and Bogdan Gabrys, "Hyperbox based machine learning algorithms: A comprehensive survey," *Soft Computing*, vol. 25, pp. 1325–1363, 2021.
6. Thanh Tung Khuat, and Bogdan Gabrys, "A comparative study of general fuzzy min-max neural networks for pattern classification problems," *Neurocomputing*, vol. 386, pp. 110-125, 2020.
7. Thanh Tung Khuat, Fang Chen, and Bogdan Gabrys, "An improved online learning algorithm for general fuzzy min-max neural network," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1-9,

2020 (**IEEE CIS Outstanding Student Paper Conference Registration Grant**).

8. Thanh Tung Khuat, and Bogdan Gabrys, "Accelerated training algorithms of general fuzzy min-max neural network using gpu for very high dimensional data," in *Proceedings of the 26th International Conference on Neural Information Processing (ICONIP)*, pp. 583-595, 2019 (**Best paper award finalists**).

9. Thanh Tung Khuat, and Bogdan Gabrys, "An online learning algorithm for a neuro-fuzzy classifier with mixed-attribute data," Submitted to *IEEE Transactions on Fuzzy Systems* (Revised and Resubmit).

Appendix A

Additional Results for Chapter 3

A.1 Datasets for the Experiments in Chapter 3

Table A.1 : Datasets were used for experiments in Chapter 3

ID	Dataset	No. samples	No. features	No. classes
1	Circle	1000	3	2
2	Complex9	3031	2	9
3	Diagnostic Breast Cancer	569	30	2
4	Glass	214	9	6
5	Ionosphere	351	34	2
6	Iris	150	4	3
7	Ringnorm	7400	20	2
8	Segmentation	2310	19	7
9	Spherical_5_2	250	2	5
10	Spiral	1000	2	2
11	Thyroid	215	5	3
12	Twonorm	7400	20	2
13	Waveform	5000	21	3
14	Wine	178	13	3
15	Yeast	1484	8	10
16	Zelnik6 (Toy dataset)	238	2	3

The experiments in Chapter 3 used 16 relatively small-sized datasets. These benchmark datasets have been widely used to evaluate machine learning algorithms such as in Salvador and Chan (2004); Zelnik-Manor and Perona (2004); Kononenko et al. (1997); Kaski and Peltonen (2003); Tung et al. (2005); Breiman (2001). The detailed information of these datasets is shown in Table A.1.

Table A.2 : Average performance of different variants of FMNN

ID	Dataset	Measure	Online GFMM	AGGLO-2	FMNN	EFMNN	KNEFMNN
1	Circle	No. of hyperboxes	172	126.25	209.75	282.75	116.5
		Training time (s)	1.2473	3.5819	1.682	3.6704	1.6951
		Testing error (%)	3.4	3.6	4.3	3.1	3.7
		Parameter-tuned time (s)	9.913	29.0167	18.08	23.6851	155.4245
2	Complex 9	No. of hyperboxes	198.75	213	450.25	458.5	257.25
		Training time (s)	4.1982	3.613	7.2618	11.6016	7.1803
		Testing error (%)	0	0	0.033	0	0
		Parameter-tuned time (s)	36.7573	40.6123	57.938	75.0684	424.9958
3	Diagnostic Breast Cancer	No. of hyperboxes	62.25	80.75	383	381.25	257.75
		Training time (s)	0.3179	2.6611	0.4174	0.3406	1.2007
		Testing error (%)	4.7463	2.987	3.1668	4.3955	4.0443
		Parameter-tuned time (s)	10.0033	147.3405	6.2507	12.1373	130.3236
4	Glass	No. of hyperboxes	107.25	106.25	109	110.5	101.5
		Training time (s)	0.1327	1.06	0.1203	0.172	0.1922
		Testing error (%)	30.3985	30.3895	27.1225	27.5943	25.7338
		Parameter-tuned time (s)	2.3779	6.7931	1.8835	3.0415	27.5698
5	Ionosphere	No. of hyperboxes	191.75	113	208.5	229	226
		Training time (s)	0.3292	5.3567	0.2457	0.3203	0.3514
		Testing error (%)	12.2585	14.2435	10.828	8.8328	8.8328
		Parameter-tuned time (s)	7.5723	131.0173	5.6189	9.4672	95.3095
6	Iris	No. of hyperboxes	52.25	51.75	37.5	47.75	27.5
		Training time (s)	0.05	0.4249	0.0205	0.071	0.0515
		Testing error (%)	5.299	5.299	3.983	5.3165	5.3165
		Parameter-tuned time (s)	0.9627	2.9324	0.8019	1.0995	10.1352
7	Ringnorm	No. of hyperboxes	507.25	1,415.25	1,899.75	2,263.25	1,217.50
		Training time (s)	15.0971	1,276.87	15.4478	25.4722	25.0148
		Testing error (%)	13.0405	9.311	16.027	25.4188	18.2705
		Parameter-tuned time (s)	621.682	117,532.42	412.5195	555.8019	5013.0365
8	Segmentation	No. of hyperboxes	803.5	809.75	906	1205.25	994.5
		Training time (s)	14.8696	192.1328	11.7049	17.0457	19.5409
		Testing error (%)	4.1558	3.9825	3.506	2.2075	2.2508
		Parameter-tuned time (s)	130.2684	736.4805	64.8881	261.9439	1691.2003
9	Spherical_5_2	No. of hyperboxes	22	23.25	21.25	24.5	14.75
		Training time (s)	0.0593	0.1074	0.038	0.0688	0.059
		Testing error (%)	1.2033	0.8	1.1905	1.197	1.197

Table A.2 : Average performance of different variants of FMNN

		Parameter-tuned time (s)	1.771	2.4714	1.6866	2.1349	18.0612
10	Spiral	No. of hyperboxes	121.5	115.75	102.75	137.5	121.5
		Training time (s)	0.4895	1.851	0.4994	0.9892	0.9478
		Testing error (%)	0	0	0	0	0
		Parameter-tuned time (s)	7.7798	16.4694	8.3901	13.0277	99.7823
11	Thyroid	No. of hyperboxes	68.5	48	95.25	96.5	108.5
		Training time (s)	0.0863	0.4432	0.1249	0.1393	0.1866
		Testing error (%)	2.315	3.7215	3.2408	3.7125	2.778
		Parameter-tuned time (s)	1.4175	5.2984	1.2786	1.885	16.3993
12	Twonorm	No. of hyperboxes	823.75	1,134.75	5,448.50	5,531.75	5,384.25
		Training time (s)	27.9473	463.9801	13.87	7.2354	13.8077
		Testing error (%)	4.527	4.3378	5.1213	5.3108	4.1623
		Parameter-tuned time (s)	615.0026	109,086.28	371.7325	549.3722	4,787.5467
13	Waveform	No. of hyperboxes	322.75	838	3220	3749.75	2757.25
		Training time (s)	11.1249	177.3769	5.6624	1.8178	31.3935
		Testing error (%)	17.88	17.76	22.52	21.36	19.88
		Parameter-tuned time (s)	305.3155	28,641.43	160.9124	312.5685	2,867.3944
14	Wine	No. of hyperboxes	46.25	25.75	39.25	74.5	27
		Training time (s)	0.0457	0.141	0.0368	0.0732	0.0824
		Testing error (%)	3.952	4.5073	2.8155	5.6313	2.8283
		Parameter-tuned time (s)	1.8072	9.3405	1.2843	1.8737	19.5112
15	Yeast	No. of hyperboxes	738.75	537.75	859.5	913.5	663
		Training time (s)	5.4222	54.0145	4.613	5.1744	8.0922
		Testing error (%)	49.3938	49.2588	49.7978	47.17	46.2265
		Parameter-tuned time (s)	44.0484	386.3297	33.0291	63.2031	580.8328
16	Zelnik6	No. of hyperboxes	26	40.75	59	45.25	34.5
		Training time (s)	0.0426	0.3498	0.0933	0.0976	0.0789
		Testing error (%)	0.4238	0.4238	0.4238	0.4238	0.4238
		Parameter-tuned time (s)	1.156	3.4367	1.1349	1.2665	10.8995

Table A.3 : Average performance of variants of FMNN using a pruning procedure

ID	Dataset	Measure	Online GFMM	AGGLO-2	FMNN	EFMNN	KNEFMNN
1	Circle	No. of hyperboxes before pruning	146.75	106.75	164	226.5	99.75
		No. of hyperboxes after pruning	124.75	90	87.75	184.75	78.25
		Training time	0.7504	2.3950	0.9043	1.8106	1.0028
		Testing error before pruning (%)	3.3	3.2	3.9	3	3.6
		Testing error after pruning (%)	3.3	3.8	4.1	3.3	3.8
2	Complex 9	No. of hyperboxes before pruning	183	196	320.75	345.5	221.5
		No. of hyperboxes after pruning	156	195.25	160.5	191.75	156.75
		Training time	2.9040	2.4693	4.3682	6.7983	4.5382
		Testing error before pruning (%)	0	0	0.033	0	0
		Testing error after pruning (%)	0	0	0.033	0	0
3	Diagnostic Breast Cancer	No. of hyperboxes before pruning	48.5	57.75	254	254	173.5
		No. of hyperboxes after pruning	24	35.75	43.5	56	32.5
		Training time	0.2235	1.3171	0.2754	0.2481	0.5979
		Testing error before pruning (%)	5.273	5.8013	4.2215	4.5735	4.3965
		Testing error after pruning (%)	5.2718	5.9760	4.223	5.4528	4.3965
4	Glass	No. of hyperboxes before pruning	78.25	77.75	72.5	79.75	73.75
		No. of hyperboxes after pruning	42.5	41.75	32.75	63.25	47.5
		Training time	0.0718	0.5802	0.0707	0.0806	0.091
		Testing error before pruning (%)	30.407	30.3985	27.1318	26.66	28.066
		Testing error after pruning (%)	35.045	34.5735	30.381	25.725	29.446
5	ionosphere	No. of hyperboxes before pruning	131.75	78.75	141.25	159	156.25
		No. of hyperboxes after pruning	35	26.75	33.5	73	72
		Training time	0.1991	2.2911	0.1522	0.1646	0.1818
		Testing error before pruning (%)	14.5343	14.2373	11.9645	9.401	9.117

Table A.3 : Average performance of variants of FMNN using a pruning procedure

		Testing error after pruning (%)	14.8185	14.2373	14.2405	11.3898	11.1055
6	Iris	No. of hyperboxes before pruning	39.5	38.25	23.75	37	21
		No. of hyperboxes after pruning	21.25	22	6.5	15.25	11.75
		Training time	0.0373	0.2035	0.0219	0.0449	0.0361
		Testing error before pruning (%)	5.9745	4.6585	3.983	5.9745	3.9833
		Testing error after pruning (%)	5.3165	4.641	3.983	5.9745	4.6588
7	Ringnorm	No. of hyperboxes before pruning	372.75	976.25	1132	1482.25	789.5
		No. of hyperboxes after pruning	207.25	716	2	855.5	10
		Training time	14.0247	488.1716	9.71927	18.79225	19.04846739
		Testing error before pruning (%)	12.6758	9.9595	18.0135	26.2163	17.4188
		Testing error after pruning (%)	12.6215	9.811	18.0135	25.7028	17.2568
8	Segmentation	No. of hyperboxes before pruning	624.75	631.75	635.25	885.25	744.5
		No. of hyperboxes after pruning	530.75	545.5	190.5	506.25	447.5
		Training time	7.0460	84.3958	5.6199	7.3270	8.4242
		Testing error before pruning (%)	4.8918	4.935	3.723	2.857	3.073
		Testing error after pruning (%)	5.7575	5.6278	4.632	3.7663	3.8528
9	Spherical_5_2	No. of hyperboxes before pruning	19.75	19.5	15.25	18.75	13.25
		No. of hyperboxes after pruning	17.25	15.25	9.5	11.25	10.75
		Training time	0.0567	0.0934	0.0358	0.0546	0.0485
		Testing error before pruning (%)	1.197	1.6003	1.5875	2.0035	2.4068
		Testing error after pruning (%)	1.197	1.197	1.58725	2.4003	2.4068
10	Spiral	No. of hyperboxes before pruning	103	105	81.5	109.25	103
		No. of hyperboxes after pruning	92	105	69.75	95.25	94.5
		Training time	0.3895	1.3104	0.3818	0.6423	0.6513
		Testing error before pruning (%)	0	0	0	0	0

Table A.3 : Average performance of variants of FMNN using a pruning procedure

		Testing error after pruning (%)	0	0	0	0	0
11	Thyroid	No. of hyperboxes before pruning	53	35	65.25	68.75	77.75
		No. of hyperboxes after pruning	36	21.25	18.75	24.75	31.5
		Training time	0.0547	0.2263	0.0694	0.0627	0.0838
		Testing error before pruning (%)	3.241	4.6475	5.5643	2.7868	2.315
		Testing error after pruning (%)	3.2408	6.036	6.0273	4.6475	3.7128
12	Twonorm	No. of hyperboxes before pruning	609.75	776.25	3655	3694.5	3563.75
		No. of hyperboxes after pruning	315.25	610.25	2864.25	3048	27
		Training time	23.5492	215.1739	13.3195	10.2797	15.0206
		Testing error before pruning (%)	4.7703	4.108	5.297	5.3648	4.4865
		Testing error after pruning (%)	4.8378	4.2973	5.4728	5.189	4.4865
13	Waveform	No. of hyperboxes before pruning	247.25	565.75	2153.75	2500	1751.5
		No. of hyperboxes after pruning	208.25	402.25	603	2354.75	46.25
		Training time	10.0276	85.5954	6.3840	4.4676	23.4186
		Testing error before pruning (%)	19.48	18.84	22.82	20.48	20
		Testing error after pruning (%)	19.36	18.4	22.6	19.7	19.66
14	Wine	No. of hyperboxes before pruning	31.5	20.5	28	51	20
		No. of hyperboxes after pruning	28	13.75	5.25	8.25	6.75
		Training time	0.0373	0.1165	0.0312	0.0480	0.0483
		Testing error before pruning (%)	3.9268	3.9268	2.8155	5.0883	3.9268
		Testing error after pruning (%)	3.9268	3.9268	2.8155	5.0883	3.9268
15	Yeast	No. of hyperboxes before pruning	522	387	582.5	618	461.25
		No. of hyperboxes after pruning	267.25	220.25	416.5	443.25	350.25
		Training time	2.3044	21.8444	1.9543	2.1734	3.3138
		Testing error before pruning (%)	49.7305	49.8655	51.1455	47.6415	46.5633

Table A.3 : Average performance of variants of FMNN using a pruning procedure

		Testing error after pruning (%)	49.5283	47.5068	47.9783	44.6765	45.3505
16	Zelnik6	No. of hyperboxes before pruning	23	35.25	42	36.25	29.75
		No. of hyperboxes after pruning	16.5	24.25	23	25.25	20
		Training time	0.0417	0.2616	0.0565	0.0557	0.0539
		Testing error before pruning (%)	0.8475	1.2643	0.8475	0.8475	0.8475
		Testing error after pruning (%)	1.695	3.3618	2.1045	2.0975	1.688

Appendix B

Additional Results for Chapter 4

B.1 Datasets for Experiments in Chapter 4

Several descriptions of the used datasets are shown in Table B.1. It is desirable to assess the effectiveness of the proposed algorithm on real-world datasets for practical usefulness. Therefore, several synthetic datasets used in Chapter 3 such as *Circle*, *Complex9*, *Ringnorm*, *Twonorm*, *Spiral*, and *Zelnik6* were not employed for the experiments in this chapter. Some of the real world datasets used in Chapter 3 were combined with other real datasets, which are show a small number of samples but high dimensionality (e.g., *Breast Cancer Coimbra*) or high numbers of samples and dimensions (e.g., *Page blocks*, *Landsat Satellite*), to create a set of experimental datasets in Table B.1.

Table B.1 : Descriptions of datasets for experiments in Chapter 4

ID	**Dataset**	**# samples**	**# features**	**# classes**
1	Blood transfusion	748	4	2
2	Breast Cancer Coimbra	116	9	2
3	Haberman	306	3	2
4	Heart	270	13	2
5	Page blocks	5473	10	5
6	Landsat Satellite	6435	36	6
7	Waveform	5000	21	3
8	Yeast	1484	8	10
9	Spherical_5_2	250	2	5

Appendix C

Additional Results for Chapter 5

C.1 Proof of Property 5.1 in Chapter 5

Proof. From the main paper, the current entropy of categorical values for the categorical feature j is computed as follows:

$$H_j(B_i) = -\sum_{l=1}^{N_j} \mathsf{P}_j(a_l \in d_{ij}) \cdot \log_2 \mathsf{P}_j(a_l \in d_{ij})$$

We have:

$$\mathsf{P}_j(a_l \in d_{ij}) = \frac{|\{a \in d_{ij} | a = a_l\}|}{|d_{ij}|} = \frac{\varphi(a_l)}{n_i}$$

where $\varphi(a_l)$ is a function returning the number of elements of a_l in the categorical dimension d_{ij} of B_i and n_i is the number of samples included in the hyperbox B_i. Therefore,

$$H_j(B_i) = -\sum_{l=1}^{N_j} \frac{\varphi(a_l)}{n_i} \cdot \log_2 \frac{\varphi(a_l)}{n_i}$$

Case 1: The *j-th* categorical feature includes a new categorical value x_j^d which does not exist in the current list of categorical values of d_{ij}.

In this case, the number of categorical values in the categorical attribute j after including x_j^d is $N_j + 1$ and $\varphi(a_l)(l \in [1, N_j]$ is unchanged. The number of samples contained in B_i is $n_i + 1$. As a result, it leads to: $\mathsf{P}_j(x_j^d \in d_{ij}) = \dfrac{1}{n_i + 1}$

Now, the new entropy on the categorical feature j when including x_j^d is:

$$H_j^{(1)}(B_i \cup \{X\}) = -\sum_{l=1}^{N_j} \mathsf{P}_j(a_l \in d_{ij}) \cdot \log_2 \mathsf{P}_j(a_l \in d_{ij}) - \mathsf{P}_j(x_j^d \in d_{ij}) \cdot \log_2 \mathsf{P}_j(x_j^d \in d_{ij})$$

$$= -\sum_{l=1}^{N_j} \frac{\varphi(a_l)}{n_i + 1} \cdot \log_2 \frac{\varphi(a_l)}{n_i + 1} - \frac{1}{n_i + 1} \cdot \log_2 \frac{1}{n_i + 1}$$

Case 2: The *j-th* categorical feature includes a categorical value x_j^d existed in the current list of categorical values of d_{ij}.

In this case, the number of distinct categorical values, N_j, storied on each of the *j-th* categorical dimension is unchanged, while the number of samples included in B_i increases by 1. Without loss of generality, assuming that $x_j^d = a_{N_j}$, then $\varphi(a_l)$ $(l \in [1, N_j - 1]$ is unchanged, while $\varphi(x_j^d) = \varphi^{new}(a_{N_j}) = \varphi^{old}(a_{N_j}) + 1$. Therefore, $\mathsf{P}_j(x_j^d \in d_{ij}) = \dfrac{\varphi^{old}(a_{N_j}) + 1}{n_i + 1}$

The new entropy on the categorical feature j when including x_j^d is:

$$
\begin{aligned}
H_j^{(2)}(B_i \cup \{X\}) &= -\sum_{l=1}^{N_j-1} \mathsf{P}_j(a_l \in d_{ij}) \cdot \log_2 \mathsf{P}_j(a_l \in d_{ij}) - \mathsf{P}_j(x_j^d \in d_{ij}) \cdot \log_2 \mathsf{P}_j(x_j^d \in d_{ij}) \\
&= -\sum_{l=1}^{N_j-1} \frac{\varphi(a_l)}{n_i+1} \cdot \log_2 \frac{\varphi(a_l)}{n_i+1} - \frac{\varphi^{old}(a_{N_j})+1}{n_i+1} \cdot \log_2 \frac{\varphi^{old}(a_{N_j})+1}{n_i+1} \\
&= -\sum_{l=1}^{N_j-1} \frac{\varphi(a_l)}{n_i+1} \cdot \log_2 \frac{\varphi(a_l)}{n_i+1} - \frac{\varphi^{old}(a_{N_j})}{n_i+1} \cdot \log_2 \frac{\varphi^{old}(a_{N_j})+1}{n_i+1} \\
&\quad - \frac{1}{n_i+1} \cdot \log_2 \frac{\varphi^{old}(a_{N_j})+1}{n_i+1}
\end{aligned}
$$

Because $1 \leq \varphi^{old}(a_{N_j})$, it leads to $-\log_2 \dfrac{\varphi^{old}(a_{N_j})+1}{n_i+1} < -\log_2 \dfrac{\varphi^{old}(a_{N_j})}{n_i+1}$ and $-\log_2 \dfrac{\varphi^{old}(a_{N_j})+1}{n_i+1} < -\log_2 \dfrac{1}{n_i+1}$ Hence,

$$
\begin{aligned}
H_j^{(2)}(B_j \cup \{X\}) &< -\sum_{l=1}^{N_j-1} \frac{\varphi(a_l)}{n_i+1} \cdot \log_2 \frac{\varphi(a_l)}{n_i+1} - \frac{\varphi^{old}(a_{N_j})}{n_i+1} \cdot \log_2 \frac{\varphi^{old}(a_{N_j})}{n_i+1} \\
&\quad - \frac{1}{n_i+1} \cdot \log_2 \frac{1}{n_i+1} \\
&= -\sum_{l=1}^{N_j} \frac{\varphi(a_l)}{n_i+1} \cdot \log_2 \frac{\varphi(a_l)}{n_i+1} - \frac{1}{n_i+1} \cdot \log_2 \frac{1}{n_i+1} = H_j^{(1)}(B_j \cup \{X\})
\end{aligned}
$$

As a result, the change in the entropy for the hyperbox B_i on the *j-th* categorical attribute:

$$
Z_j^{(2)} = H_j^{(2)}(B_j \cup \{X\}) - H_j(B_j) < H_j^{(1)}(B_j \cup \{X\}) - H_j(B_j) = Z_j^{(1)}
$$

Therefore, the change in the entropy for each categorical dimension of the hyperbox B_i will takes its maximum value if the categorical value on that categorical dimension of the input pattern does not appear in the current list of categorical values stored on that categorical dimension. This property is proved. □

C.2 Proof of Property 5.2 in Chapter 5

Proof. Based on the Property 5.1, it can be seen that the maximum entropy change occurs on the *j-th* categorical dimension when it accommodates a new categorical value which does not exist in its current values. Hence, we obtain:

$$
\begin{aligned}
Z_j \leq Z_j^{(1)} &= H_j^{(1)}(B_i \cup \{X\}) - \frac{n_i}{n_i+1} H_j(B_i) \\
&= -\sum_{l=1}^{N_j} \frac{\varphi(a_l)}{n_i+1} \cdot \log_2 \frac{\varphi(a_l)}{n_i+1} - \frac{1}{n_i+1} \cdot \log_2 \frac{1}{n_i+1} + \frac{n_i}{n_i+1} \cdot \sum_{l=1}^{N_j} \frac{\varphi(a_l)}{n_i} \cdot \log_2 \frac{\varphi(a_l)}{n_i} \\
&= -\frac{1}{n_i+1} \sum_{l=1}^{N_j} [\varphi(a_l) \cdot [\log_2 \varphi(a_l) - \log_2(n_i+1)]] + \frac{\log_2(n_i+1)}{n_i+1} \\
&+ \frac{1}{n_i+1} \cdot \sum_{l=1}^{N_j} \varphi(a_l) \cdot [\log_2 \varphi(a_l) - \log_2 n_i] \\
&= \frac{\log_2(n_i+1)}{n_i+1} \sum_{l=1}^{N_j} \varphi(a_l) + \frac{\log_2(n_i+1)}{n_i+1} - \frac{\log_2 n_i}{n_i+1} \cdot \sum_{l=1}^{N_j} \varphi(a_l) \\
&= \frac{\log_2(n_i+1)}{n_i+1} \left[\sum_{l=1}^{N_j} \varphi(a_l) + 1 \right] - \frac{\log_2 n_i}{n_i+1} \cdot \sum_{l=1}^{N_j} \varphi(a_l)
\end{aligned}
$$

We have: $\sum_{l=1}^{N_j} \varphi(a_l) = n_i$. Hence,

$$
Z_j^{(1)} = \frac{\log_2(n_i+1)}{n_i+1}(n_i+1) - \frac{\log_2 n_i}{n_i+1} \cdot n_i = \log_2(n_i+1) - \frac{n_i}{n_i+1} \cdot \log_2 n_i
$$

As a result,

$$
Z_j \leq \log_2(n_i+1) - \frac{n_i}{n_i+1} \cdot \log_2 n_i
$$

The property is proved. □

C.3 Proof of Property 5.3 in Chapter 5

Proof. First of all, it is necessary to prove that $Z_j \leq 1;\ \forall j \in [1, r]$

Let $f(n) = \log_2(n+1) - \dfrac{n}{n+1} \cdot \log_2 n;\ (n \geq 1)$. The first order derivative of $f(n)$ is:

$$f^{'}(n) = \frac{1}{(n+1)\ln 2} - \frac{\log_2 n}{(n+1)^2} - \frac{1}{(n+1)\ln 2} = -\frac{\log_2 n}{(n+1)^2}$$

Therefore, $f^{'}(n) \leq 0,\ \forall n \geq 1$. As a result, $f(n)$ is decreasing on the interval $[1, +\infty]$. This leads to $f(n) \leq f(1) = 1;\ \forall n \geq 1$. Hence, $\log_2(n_i+1) - \dfrac{n_i}{n_i+1} \cdot \log_2 n_i \leq 1;\ \forall n_i \geq 1$. From the Property 5.2, it can obtain

$$Z_j \leq \log_2(n_i+1) - \frac{n_i}{n_i+1} \cdot \log_2 n_i;\ \forall j \in [1, r]$$

As a result, $Z_j \leq 1;\ \forall j \in [1, r]$ is proved.

Next, it is necessary to prove that $Z_j \geq 0;\ \forall j \in [1, r]$. From the Property 5.1, it can be seen that the change in the entropy on each categorical dimension j takes a smaller value if the included value x_j^d has already been in the current categorical values of d_{ij} in the hyperbox B_i. Hence, it is sufficient to only prove $Z_j \geq 0$ for this case. We obtain:

$$\begin{aligned}
Z_j &= H_j^{(2)}(B_i \cup \{X\}) - \frac{n_i}{n_i+1} \cdot H_j(B_i) \\
&= -\sum_{l=1}^{N_j} \mathsf{P}_j(a_l \in d_{ij} | B_i \cup \{X\}) \cdot \log_2 \mathsf{P}_j(a_l \in d_{ij} | B_i \cup \{X\}) \\
&+ \frac{n_i}{n_i+1} \cdot \sum_{l=1}^{N_j} \mathsf{P}_j(a_l \in d_{ij} | B_i) \cdot \log_2 \mathsf{P}_j(a_l \in d_{ij} | B_i) \\
&= -\sum_{l=1}^{N_j} \frac{\varphi(a_l | B_i \cup \{X\})}{n_i+1} \cdot \log_2 \frac{\varphi(a_l | B_i \cup \{X\})}{n_i+1} + \frac{n_i}{n_i+1} \cdot \sum_{l=1}^{N_j} \frac{\varphi(a_l | B_i)}{n_i} \cdot \log_2 \frac{\varphi(a_l | B_i)}{n_i}
\end{aligned}$$

To prove $Z_j \geq 0$, we need to prove that:

$$n_i \cdot \frac{\varphi(a_l|B_i)}{n_i} \cdot \log_2 \frac{\varphi(a_l|B_i)}{n_i} \geq (n_i+1) \cdot \frac{\varphi(a_l|B_i \cup \{X\})}{n_i+1} \cdot \log_2 \frac{\varphi(a_l|B_i \cup \{X\})}{n_i+1}, \forall l \in [1, N_j]$$

$$\Leftrightarrow \varphi(a_l|B_i) \cdot \log_2 \frac{\varphi(a_l|B_i)}{n_i} \geq \varphi(a_l|B_i \cup \{X\}) \cdot \log_2 \frac{\varphi(a_l|B_i \cup \{X\})}{n_i+1}, \forall l \in [1, N_j] \tag{C.1}$$

Let $x = \varphi(a_l|B_i) \geq 1$, then $\varphi(a_l|B_i \cup \{X\}) = x+1$ or $\varphi(a_l|B_i \cup \{X\}) = x$ because the number of elements of a_l on the *j-th* categorical feature can remain unchanged or increase by 1 when B_i includes X. In the case that $\varphi(a_l|B_i \cup \{X\}) = x$, we have:

$$(\text{C.1}) \Leftrightarrow x \cdot \log_2 \frac{x}{n_i} \geq x \cdot \log_2 \frac{x}{n_i+1}$$

This inequality is obviously true because $x \geq 1$ and $\frac{x}{n_i} \geq \frac{x}{n_i+1} \Leftrightarrow \log_2 \frac{x}{n_i} \geq \log_2 \frac{x}{n_i+1}$.

In the case that $\varphi(a_l|B_i \cup \{X\}) = x+1$, we need to prove that:

$$x \cdot \log_2 \frac{x}{n_i} \geq (x+1) \cdot \log_2 \frac{x+1}{n_i+1} \tag{C.2}$$

Let $a_1, \ldots, a_m$ and $b_1, \ldots, b_m$ be non-negative numbers, and $a = \sum_{k=1}^{m} a_k$, $b = \sum_{k=1}^{m} b_k$, the log-sum inequality states that

$$\sum_{k=1}^{m} a_i \cdot \log_2 \frac{a_i}{b_i} \geq a \cdot \log_2 \frac{a}{b}$$

Based on this log-sum inequality, we obtain:

$$x \cdot \log_2 \frac{x}{n_i} + 1 \cdot \log_2 \frac{1}{1} \geq (x+1) \cdot \log_2 \frac{x+1}{n_i+1}$$

We have $1 \cdot \log_2 \frac{1}{1} = 0$, thus the inequality (C.2) is true.

From all above proofs, we obtain: $0 \leq Z_j \leq 1;\ \forall j \in [1, r]$. The property is proved. □

C.4 Proof of Property 5.4 in Chapter 5

Proof. From the properties 5.2 and 5.3, we have:

$$0 \leq Z_j \leq \log_2(n_i + 1) - \frac{n_i}{n_i + 1} \cdot \log_2 n_i$$

To prove this property, therefore, we need to show that:

$$\lim_{n_i \to +\infty} \log_2(n_i + 1) - \frac{n_i}{n_i + 1} \cdot \log_2 n_i = 0$$

Let $f(n) = \log_2(n + 1) - \frac{n}{n+1} \cdot \log_2 n = \frac{(n+1) \cdot \log_2(n+1) - n \cdot \log_2 n}{n+1}$, then $\lim_{n \to +\infty} f(n)$ is the indeterminate form $\frac{\infty}{\infty}$. Let's apply L'Hospital's rule:

$$\begin{aligned} \lim_{n \to +\infty} f(n) &= \lim_{n \to +\infty} \frac{(n+1) \cdot \log_2(n+1) - n \cdot \log_2 n}{n+1} \\ &= \lim_{n \to +\infty} \left[\log_2(n+1) + \frac{1}{\ln 2} - \log_2 n - \frac{1}{\ln 2} \right] = \lim_{n \to +\infty} \log_2(1 + \frac{1}{n}) = 0 \end{aligned}$$

The property is proved. □

C.5 Datasets

Table C.1 shows the detailed information about the datasets used for the experiments in this study. These datasets were used in the previous research (Khuat and Gabrys 2021b) regarding different solutions to learn from mixed-attribute data for the general fuzzy min-max neural network. These datasets show the diversity in the numbers of samples, classes, and features. All datasets are class-imbalanced.

C.6 Additional Results

C.6.1 Additional Experimental Results for Chapter 5

This section provides the interested readers with the detailed results for the experiments presented in Chapter 5.

Table C.1 : Experimental dataset used in Chapter 5

ID	Dataset	# Patterns	# Classes	# Continuous attributes	# categorical attributes
1	abalone	4177	28	7	1
2	australian	690	2	6	8
3	cmc	1473	3	2	7
4	dermatology	358	6	1	33
5	flag	194	8	10	18
6	german	1000	2	7	13
7	heart	270	2	7	6
8	japanese credit	653	2	6	9
9	molecular biology	3190	3	0	60
10	nursery	12960	5	0	8
11	post operative	87	3	1	7
12	tae	151	3	1	4
13	tic tac toe	958	2	0	9
14	zoo	101	7	1	15

Fig C.1 provides a graphical illustration for two ways of estimating the value of α shown in subsection 5.3.3. Figure C.2 to Figure C.4 show the change in the CBA scores for different values of α. These results support the contents in subsection 5.3.1. Fig C.5 presents the change in the CBA values for different values of δ using the IOL-GFMM-v1 algorithm. Meanwhile, Figure C.6 shows similar results for the IOL-GFMM-v2 algorithm. These figures consolidate the comments in subsection 5.3.1.

Figures from C.7 to C.9 show the distribution of the obtained values of α using different methods to find the suitable values for α. The best results in each row of the tables in this document are highlighted in bold. These figures support the explanation in subsection 5.3.3 related to the effectiveness of each method of finding the trade-off parameter α.

Table C.2 presents the average class balanced accuracy for different learning algorithms with mixed-attribute handling ability over 40 iterations (10 times repeated stratified 4-fold cross-validation). The number of generated hyperboxes from these algorithms is shown in Table C.3. These are the results for the experiment mentioned in subsection 5.3.2.

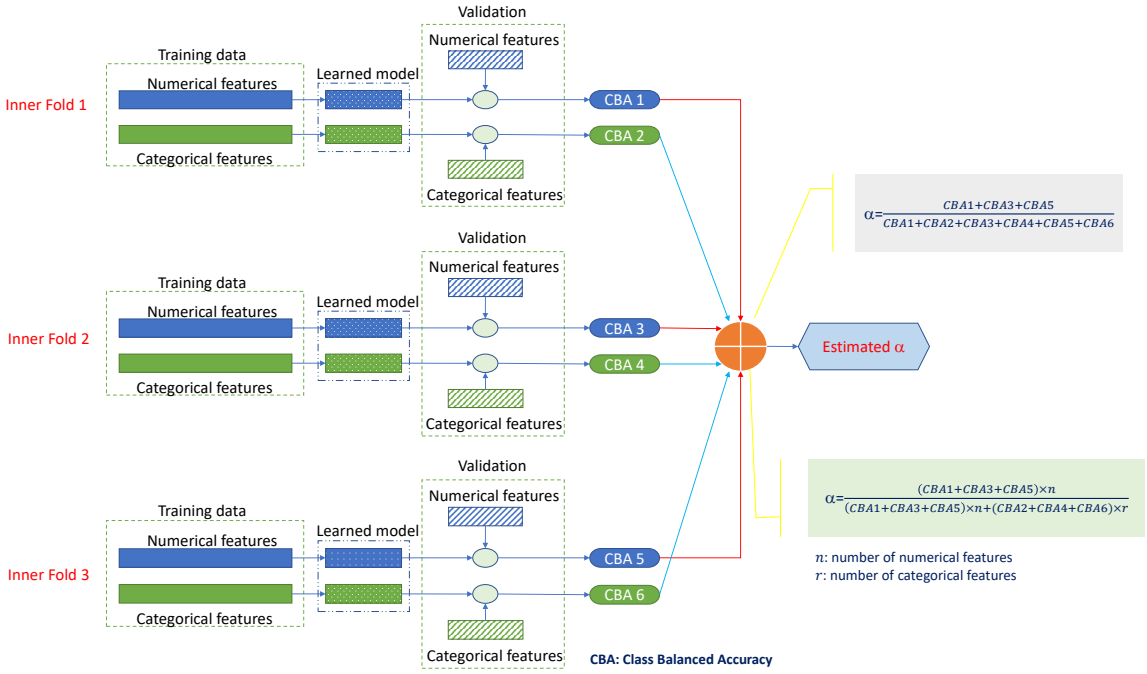
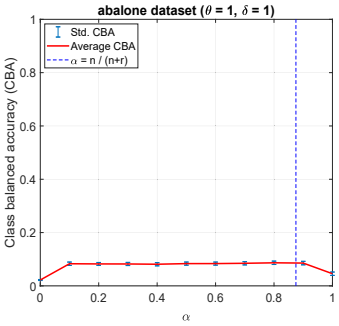


Figure C.1 : A demonstration for the methods used to estimate the values for parameter α.

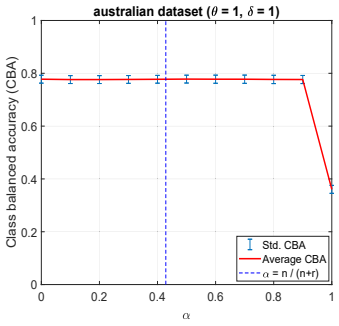
Table C.4 shows the average CBA for the proposed methods and the original improved online learning algorithm using different encoding techniques to transform the categorical features into numerical features. These results supplement to the claims in subsection 5.3.2.

Table C.5 describes the average CBA values for the different methods to find the appropriate values for α using the proposed learning algorithms. These are the detailed empirical results for the experiment presented in subsection 5.3.3.

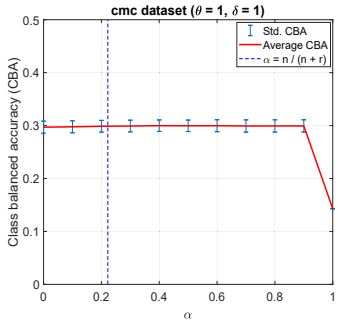
Table C.6 shows the average CBA values for four learning algorithms using the hyper-parameter tuning approach. The ranks for these results are presented in Table C.7. These results add more details to the content in subsection 5.3.3.



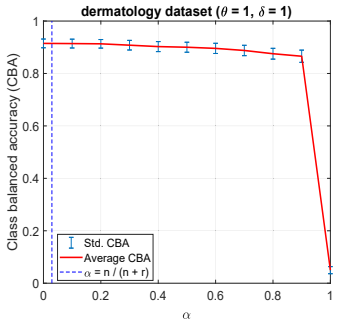
(a) abalone



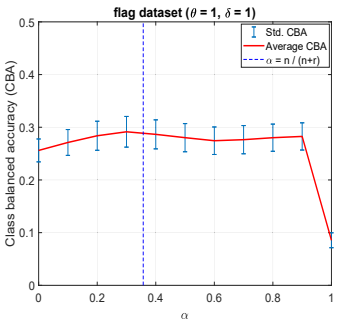
(b) australian



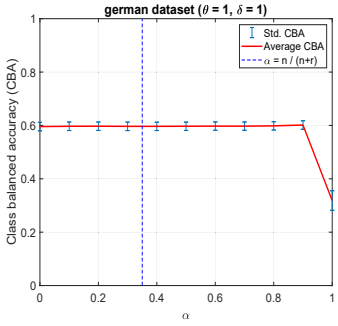
(c) cmc



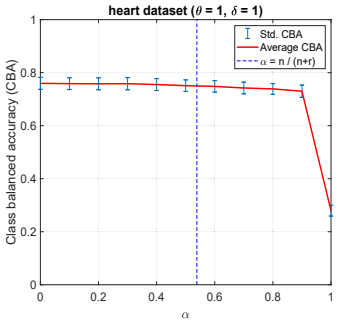
(d) dermatology



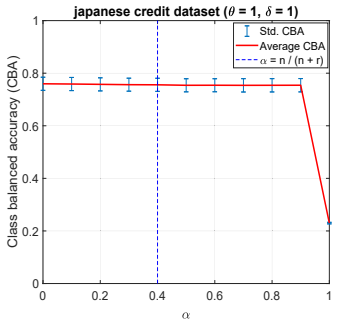
(e) flag



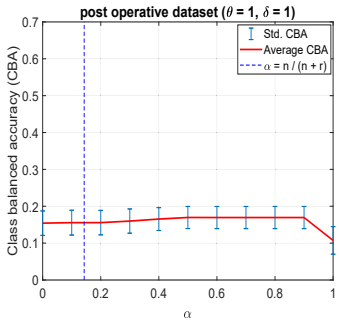
(f) german



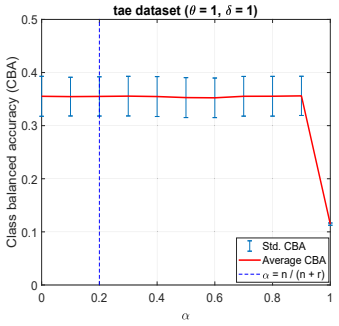
(g) heart



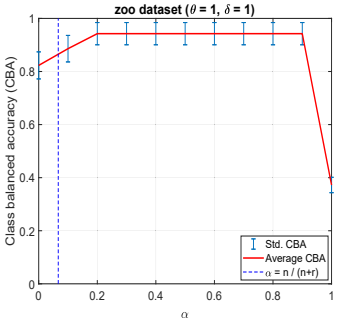
(h) japanese credit



(i) post operative

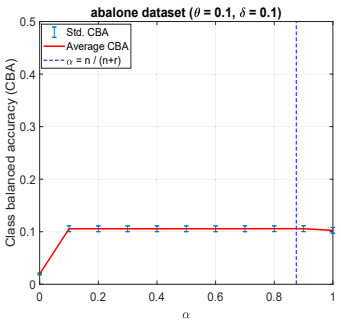


(j) tae

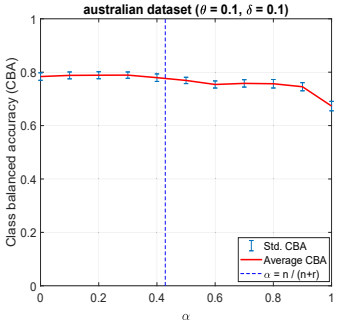


(k) zoo

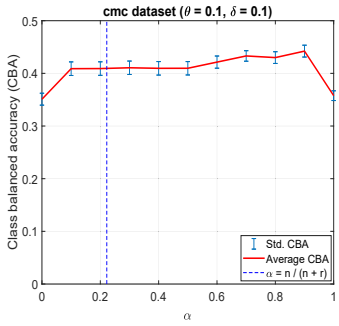
Figure C.2 : Class balanced accuracy values for different values of α ($\theta = 1, \delta = 1$).



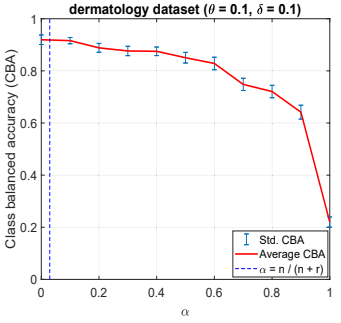
(a) abalone



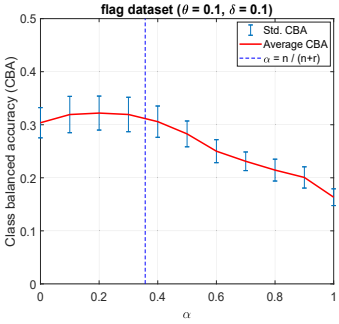
(b) australian



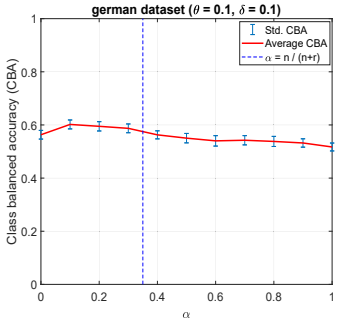
(c) cmc



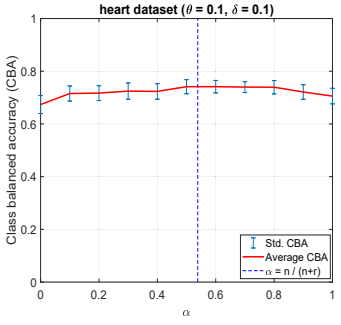
(d) dermatology



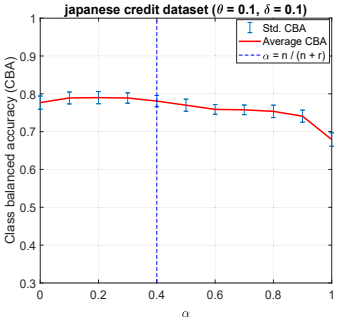
(e) flag



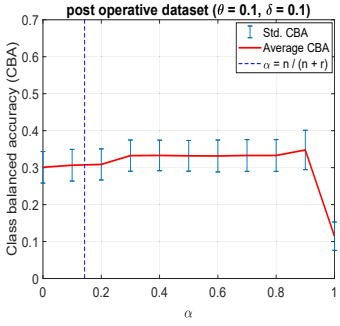
(f) german



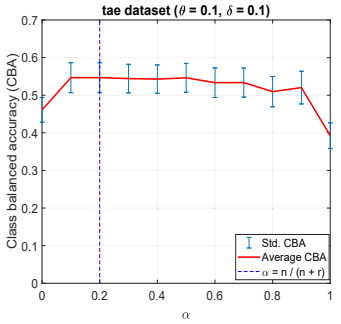
(g) heart



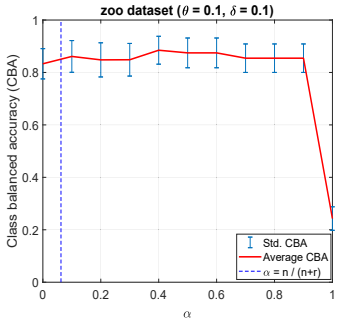
(h) japanese credit



(i) post operative

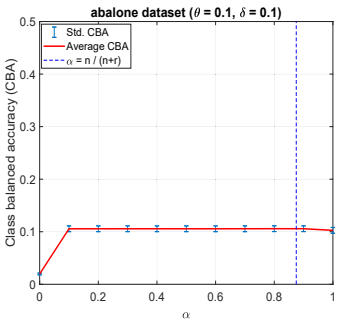


(j) tae

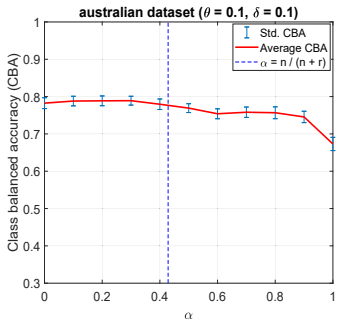


(k) zoo

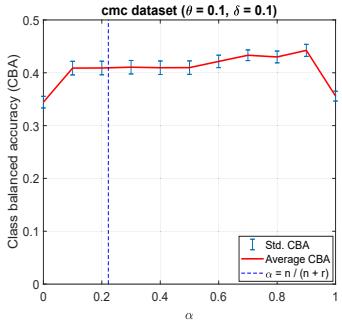
Figure C.3 : Class balanced accuracy values for different values of α using the IOL-GFMM-v1 algorithm ($\theta = 0.1, \delta = 0.1$).



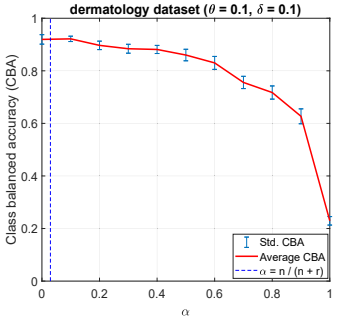
(a) abalone



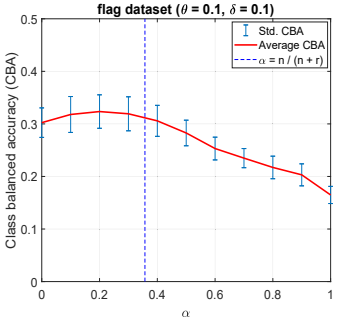
(b) australian



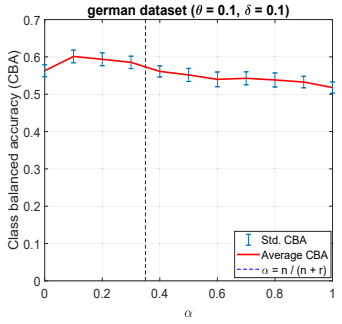
(c) cmc



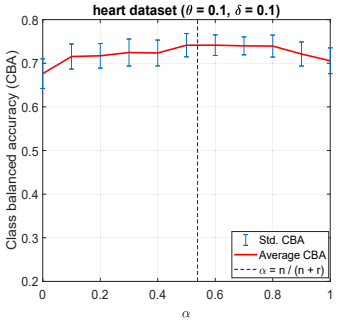
(d) dermatology



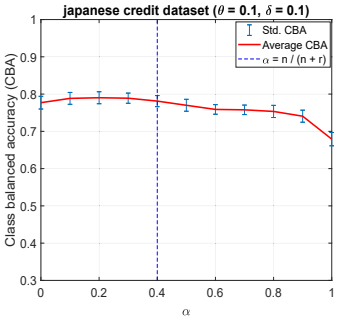
(e) flag



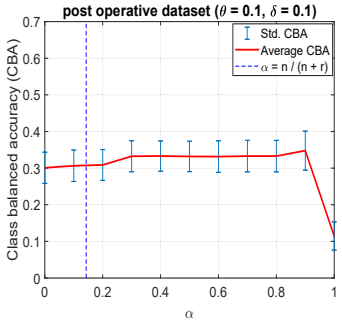
(f) german



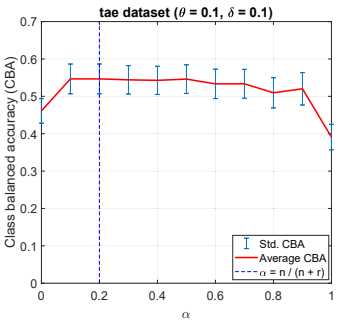
(g) heart



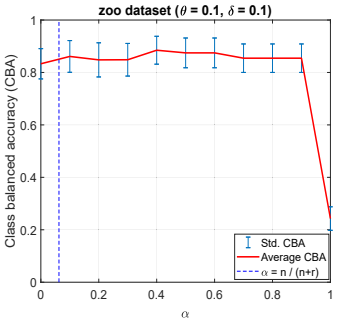
(h) japanese credit



(i) post operative

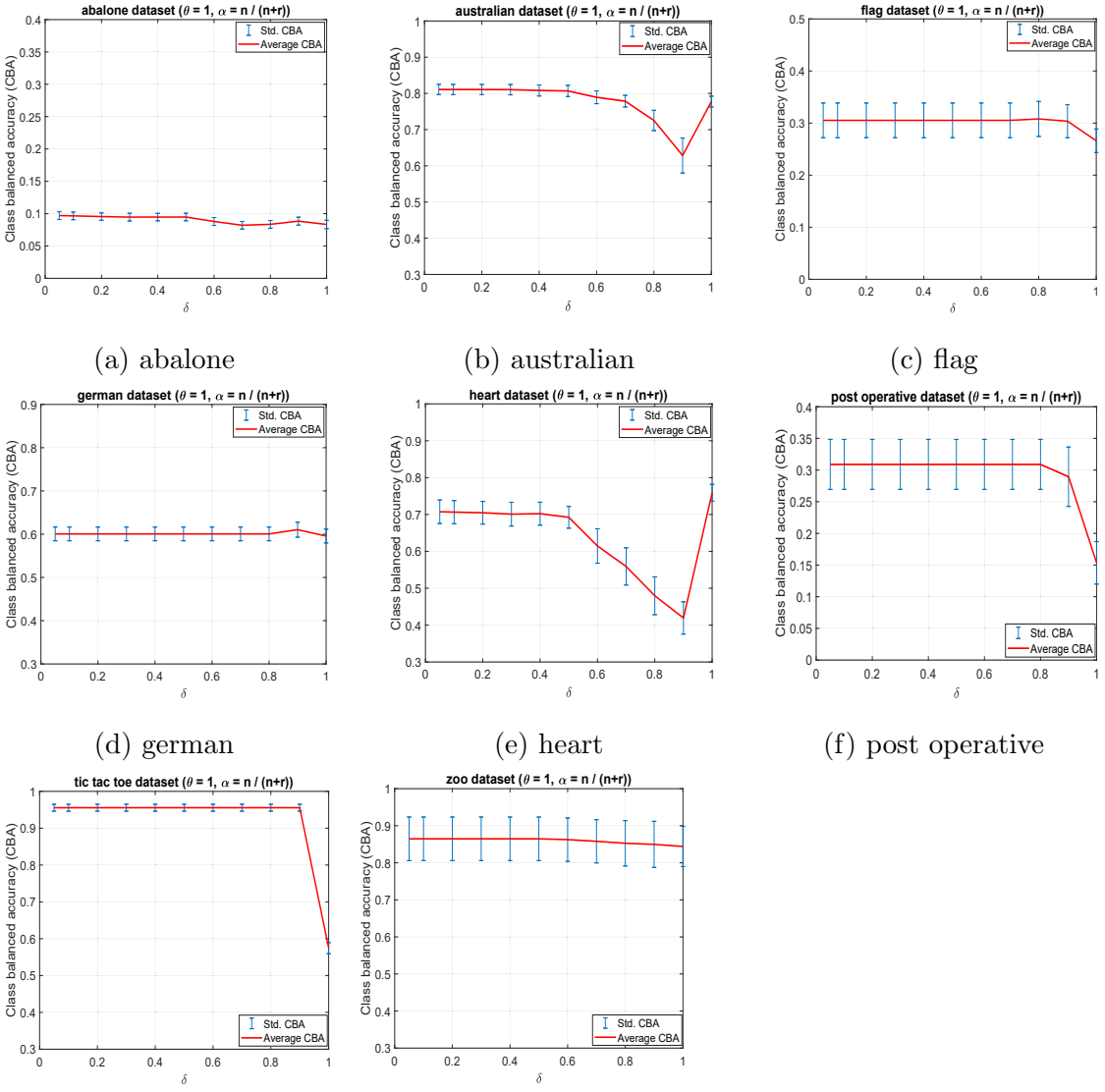


(j) tae



(k) zoo

Figure C.4 : Class balanced accuracy values for different values of α using the IOL-GFMM-v2 algorithm ($\theta = 0.1, \delta = 0.1$).

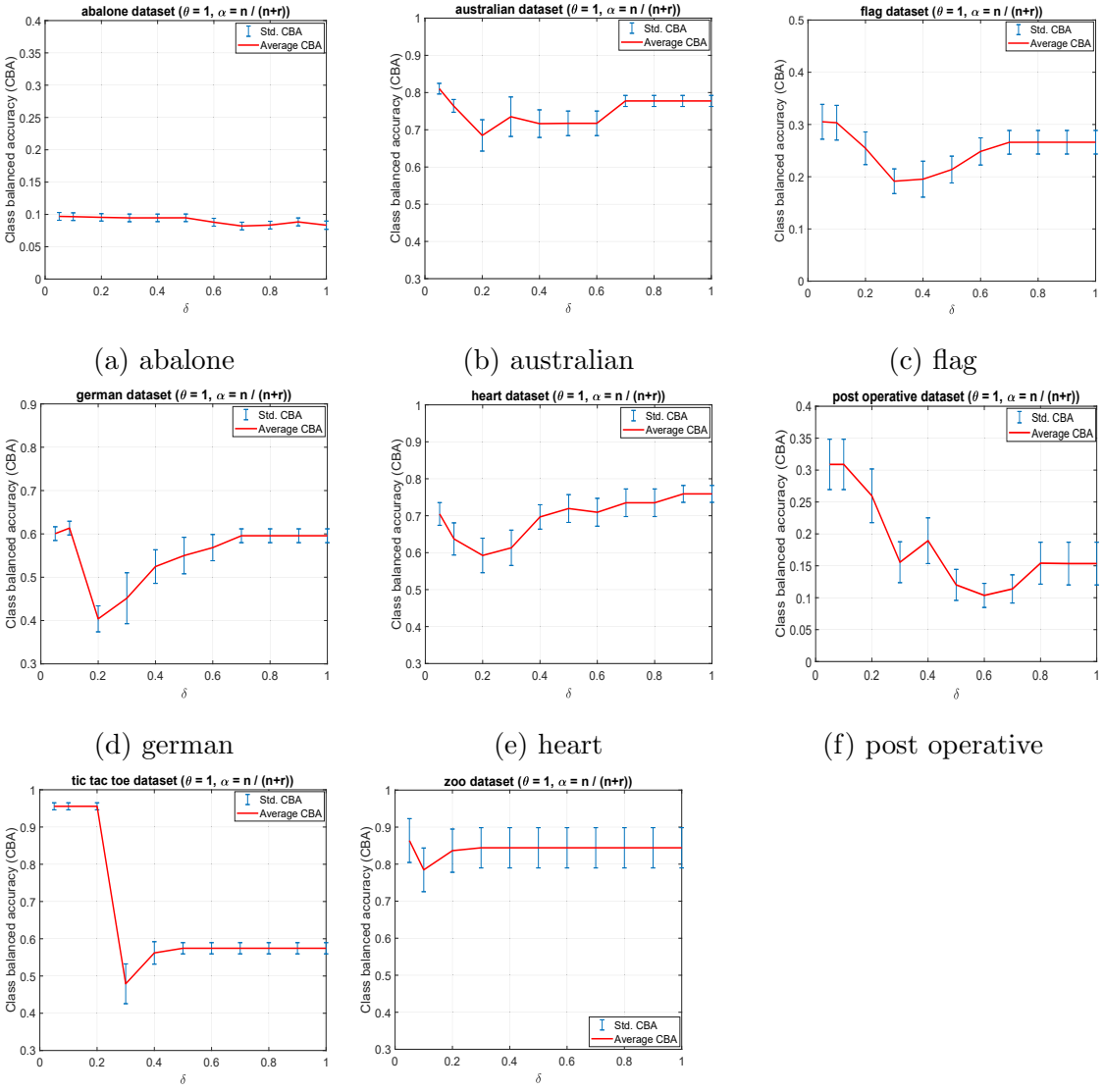


(a) abalone (b) australian (c) flag

(d) german (e) heart (f) post operative

(g) tic tac toe (h) zoo

Figure C.5 : Class balanced accuracy values for different values of δ using the IOL-GFMM-v1 algorithm ($\theta = 1, \alpha = n/(n+r)$).



(a) abalone (b) australian (c) flag

(d) german (e) heart (f) post operative

(g) tic tac toe (h) zoo

Figure C.6 : Class balanced accuracy values for different values of δ using the IOL-GFMM-v2 algorithm ($\theta = 1, \alpha = n/(n+r)$).

Table C.2 : Average class balanced accuracy for the proposed EIOL-GFMM algorithms and two existing algorithms with the mixed-attribute learning ability

Dataset	θ	Onln-GFMM-M1			Onln-GFMM-M2			EIOL-GFMM-v1			EIOL-GFMM-v2		
		$\eta = 0.1$	$\eta = 0.7$	$\eta = 1$	$\beta = 0.25r$	$\beta = 0.5r$	$\beta = 0.75r$	$\delta = 0.1$	$\delta = 0.7$	$\delta = 1$	$\delta = 0.1$	$\delta = 0.7$	$\delta = 1$
abalone	0.1	0.10182	0.10156	0.1039	0.10195	0.10195	0.10195	**0.10592**	0.09893	0.0951	**0.10592**	0.09893	0.0951
	0.7	0.09182	0.08919	0.07923	0.08662	0.08662	0.08662	**0.09384**	0.07846	0.08748	**0.09384**	0.07846	0.08748
	1	0.08831	0.08357	0.0729	0.08455	0.08455	0.08455	**0.094**	0.07766	0.08545	**0.094**	0.07766	0.08545
australian	0.1	0.77345	0.76536	0.75532	0.62766	0.6276	0.6425	**0.77871**	**0.77871**	0.74137	**0.77871**	0.73931	0.74137
	0.7	0.7939	0.69939	0.62978	0.54578	0.53535	0.50845	**0.80259**	0.78758	0.77546	0.78005	0.77846	0.77546
	1	0.79382	0.69821	0.60577	0.55546	0.55456	0.50156	**0.80242**	0.78793	0.7771	0.78057	0.7771	0.7771
cmc	0.1	0.4003	0.39828	0.37507	0.39889	0.40054	**0.41933**	0.40938	0.41002	0.2987	0.40936	0.30153	0.2987
	0.7	**0.39879**	0.31526	0.28304	0.24961	0.24822	0.31226	0.36564	0.32006	0.32407	0.34705	0.33095	0.32407
	1	**0.39603**	0.31665	0.29373	0.28774	0.28759	0.19917	0.35431	0.31151	0.29882	0.32979	0.29882	0.29882
dermatology	0.1	0.89702	0.88468	0.87987	0.80026	0.80026	0.80822	0.91871	0.91871	0.81362	**0.92217**	0.81362	0.81362
	0.7	0.897	0.87034	0.86026	0.51967	0.51967	0.53411	**0.91871**	**0.91871**	0.86642	0.9095	0.86642	0.86642
	1	0.897	0.87097	0.85932	0.45787	0.45787	0.51649	**0.91871**	**0.91871**	0.91473	0.91284	0.91473	0.91473
flag	0.1	0.20758	0.2098	0.21106	0.27048	0.27417	0.27848	0.31378	0.31378	0.25518	**0.31398**	0.25518	0.25518
	0.7	0.20707	0.21187	0.22017	0.28746	0.2849	0.25168	**0.31232**	**0.31232**	0.27507	0.30981	0.27569	0.27507
	1	0.20722	0.21272	0.21607	0.25298	0.25797	0.2513	**0.31301**	**0.31301**	0.28709	0.31044	0.28691	0.28709
german	0.1	**0.59853**	0.5947	0.5906	0.49704	0.50612	0.53338	0.57323	0.57323	0.58901	0.57247	0.58943	0.58901
	0.7	**0.59975**	0.56437	0.53981	0.51698	0.51199	0.52676	0.57175	0.57175	0.57531	0.5759	0.57426	0.57531
	1	**0.60161**	0.54852	0.51234	0.37304	0.37304	0.43286	0.56632	0.56632	0.5969	0.58602	0.5969	0.5969
heart	0.1	0.72107	0.72131	0.73083	**0.7632**	0.75282	0.74892	0.74995	0.74995	0.73615	0.74995	0.73522	0.73615
	0.7	0.72325	0.71335	0.64509	0.59566	0.55006	0.54566	**0.7445**	0.73701	0.73827	0.73341	0.72268	0.73827
	1	0.72291	0.71955	0.60935	0.56718	0.53851	0.53506	0.74169	**0.76615**	0.74998	0.74213	0.74415	0.74998
japanese credit	0.1	0.77503	0.7741	0.75015	0.65456	0.64753	0.67543	0.78084	0.78084	0.70476	**0.78116**	0.70675	0.70476
	0.7	0.798	0.7099	0.60187	0.52062	0.51952	0.51664	**0.80483**	0.78195	0.75976	0.74552	0.76258	0.75976
	1	0.79766	0.7024	0.59146	0.50578	0.50578	0.50519	**0.80554**	0.78266	0.75584	0.74095	0.75584	0.75584
molecular biology	-	0.54925	0.53621	0.4805	0.43242	0.43218	0.17394	0.62653	0.63410	0.47090	**0.63435**	0.45035	0.47090
nursery	-	**0.84465**	0.38292	0.21174	0.50402	0.50402	0.50402	0.78223	0.78223	0.33203	0.78223	0.33203	0.33203
post operative	0.1	0.33633	0.31583	0.28579	0.27155	0.31368	**0.35604**	0.30645	0.30645	0.12997	0.30645	0.11659	0.12997
	0.7	**0.33491**	0.28111	0.2672	0.23059	0.23783	0.24881	0.30748	0.30748	0.1589	0.30748	0.15175	0.1589
	1	**0.33491**	0.28598	0.27953	0.24144	0.23603	0.27637	0.30888	0.30888	0.15504	0.30888	0.11313	0.15504
tae	0.1	0.54089	0.49741	0.43901	0.48868	0.52062	0.53408	**0.54668**	**0.54668**	0.43773	**0.54668**	0.39556	0.43773
	0.7	0.54286	0.42037	0.3789	0.32134	0.35273	0.45906	**0.55159**	**0.55159**	0.36061	**0.55159**	0.35133	0.36061
	1	0.53996	0.39701	0.3698	0.31717	0.3535	0.44079	**0.55181**	**0.55181**	0.35503	**0.55181**	0.31224	0.35503
tic tac toe	-	0.85823	0.8238	0.49501	0.44417	0.44417	0.44417	**0.95561**	**0.95561**	0.57448	**0.95561**	0.57448	0.57448
zoo	0.1	0.55787	0.55787	0.67455	0.86093	0.86093	0.86093	0.8647	0.86375	**0.90143**	0.80119	**0.90143**	**0.90143**
	0.7	0.6146	0.6146	0.61193	0.7827	0.7827	0.78776	0.8647	0.85801	**0.86369**	0.79869	**0.86369**	**0.86369**
	1	0.6146	0.6146	0.57089	0.7396	0.7396	0.73202	**0.8647**	0.85801	0.85744	0.79869	0.85744	0.85744

Table C.3 : The Average Number of Generated Hyperboxes from the Proposed Method and Other Existing Algorithms with Mixed-Attribute Learning Ability

Dataset	θ	Onln-GFMM-M1			Onln-GFMM-M2			EIOL-GFMM-v1			EIOL-GFMM-v2		
		$\eta = 0.1$	$\eta = 0.7$	$\eta = 1$	$\beta = 0.25r$	$\beta = 0.5r$	$\beta = 0.75r$	$\delta = 0.1$	$\delta = 0.7$	$\delta = 1$	$\delta = 0.1$	$\delta = 0.7$	$\delta = 1$
abalone	0.1	693.95	591.725	522.825	704.15	704.15	704.15	1675.2	1318.925	919.45	1675.2	1318.925	919.45
	0.7	77.225	56.875	46.75	79.375	79.375	79.375	1665.075	605.25	934.1	1665.075	605.25	934.1
	1	63.3	45.275	37.375	65.275	65.275	65.275	1662.125	599.225	932.375	1662.125	599.225	932.375
australian	0.1	472.15	348.225	222.6	198.775	206.05	279.025	493.225	493.225	203.2	493.225	204.7	203.2
	0.7	260.275	95.825	42.7	11.675	12.875	21.875	334.45	207.5	11.85	250.55	12	11.85
	1	253.25	86.4	35.725	2	2.675	6.95	329.225	198.475	2	243.35	2	2
cmc	0.1	880.95	447.85	202.325	118.05	127.05	240	925.925	924.9	131.625	925.9	137.6	131.625
	0.7	496.325	117.2	38.675	10.85	11.775	24.475	736.475	184.975	14.25	341.1	14.5	14.25
	1	471.1	99	28.8	3	3.025	8.275	734.7	130.725	3	267.6	3	3
dermatology	0.1	268.5	212.65	105.975	38.65	38.675	71.2	268.5	268.5	46.225	255.1	46.225	46.225
	0.7	267.375	164.825	68.025	7.8	7.8	20	267.425	267.425	12.125	220.825	12.125	12.125
	1	267.375	163.85	67.1	6	6	19.05	267.425	267.425	6	220.35	6	6
flag	0.1	143.2	137.5	107.825	101.95	105.075	124.35	143.2	143.2	101.95	141.175	101.95	101.95
	0.7	140.975	123.3	44.725	23.55	25.475	49.5	140.975	140.975	23.55	132.675	23.825	23.55
	1	140.675	116.625	34.2	8	9.65	31.55	140.675	140.675	8	131.5	8.025	8
german	0.1	747.925	711.3	549.55	524.225	547.45	669.3	748.225	748.225	523.7	744.975	526.225	523.7
	0.7	726.225	299.575	115.225	32.25	36.375	87.725	738.825	738.825	32.6	666.725	32.975	32.6
	1	714.925	201.1	69.575	2	2.125	10.3	735.425	735.425	2	516	2	2
heart	0.1	198.675	197.475	181.375	181.225	183.525	188.5	199.1	199.1	181.425	199.1	182.3	181.425
	0.7	92.05	71.5	16.9	12.5	13.4	17.425	103.45	73.175	12.725	97.625	13.1	12.725
	1	74.925	53.625	7.25	2	3.375	4.85	89.85	50.8	2	76.925	2.2	2
japanese_credit	0.1	451.175	340.15	219.775	198.65	202	233.25	468.875	468.875	202.55	468.3	203.725	202.55
	0.7	252.775	95.675	42.475	12.35	12.95	15.775	320.425	210.3	12.725	197.325	12.95	12.725
	1	246.7	87.475	35.275	2	2	4.425	315.675	198.4	2	183.9	2	2
molecular_biology	-	2271.125	2034.025	1060.925	3.9	178.425	1743.325	2279.6	2279.6	3	2195.1	5.55	3
nursery	-	5434.325	156.4	33.075	5	5	5.175	9720	9720	5	9720	5	5
post_operative	0.1	58.575	38.9	13.2	7	9.05	19.1	59.25	59.25	7	59.25	8.7	7
	0.7	55.1	30.075	8.7	4.825	5.175	11.15	56.125	56.125	4.45	56.125	5.05	4.45
	1	55.05	29.375	7.475	2.75	3.95	9.425	56.05	56.05	2.75	56.05	3.625	2.75
tae	0.1	80.75	47.15	35.125	22.125	32.325	68.775	84.15	84.05	23.325	84.05	26.45	23.325
	0.7	70.55	29	21.55	5.025	6.95	37.125	77.35	77.225	6.225	77.225	6.775	6.225
	1	69.95	28.425	20.575	3.025	4.625	33.325	76.75	76.625	3	76.625	3.8	3
tic_tac_toe	-	639.125	239.275	18.175	2	2	2.05	718.5	718.5	2	718.5	2	2
zoo	0.1	48.4	48.4	13.375	13.375	13.375	14.5	49.35	45.625	13.375	26.65	13.375	13.375
	0.7	44.975	44.975	8	8	8	8.55	46	39.95	8	19.275	8	8
	1	44.975	44.975	7	7	7	7.25	46	39.95	7	18.15	7	7

Table C.4 : The average class balanced accuracy for the proposed method and the IOL-GFMM using encoding techniques

Dataset	$\theta(=\delta)$	CatBoost	One-hot	LOO	Label	Target	James-Stein	Helmert	Sum	EIOL-GFMM-v1	EIOL-GFMM-v2
abalone	0.1	0.1077	0.11216	0.08367	**0.11217**	0.11161	0.11161	0.11216	0.11215	0.10592	0.10592
	0.7	**0.11117**	0.04521	0.04599	0.07214	0.0544	0.0544	0.04521	0.03942	0.07846	0.07846
	1	**0.11041**	0.03651	0.05247	0.07843	0.07027	0.07027	0.02491	0.02491	0.08545	0.08545
australian	0.1	0.79148	0.6283	0.70876	0.77303	0.77283	0.77321	0.62855	**0.79163**	0.77871	0.77871
	0.7	**0.81259**	0.63919	0.60169	0.79844	0.80302	0.80712	0.63943	0.80088	0.78758	0.77846
	1	0.72686	0.36155	0.45375	0.38134	0.63626	0.64024	0.36384	0.36405	**0.7771**	**0.7771**
cmc	0.1	0.27015	0.39891	0.39403	**0.4124**	0.40797	0.40849	0.39891	0.4095	0.40938	0.40936
	0.7	0.24563	0.38885	0.23629	0.37042	0.37329	0.3739	0.38885	**0.40109**	0.32006	0.33095
	1	0.28563	0.14279	0.13443	0.14279	0.14482	0.14482	0.14301	0.14301	**0.29882**	**0.29882**
dermatology	0.1	0.7371	0.05674	0.87801	0.55851	0.87721	0.85747	0.05674	0.28197	0.91871	**0.92217**
	0.7	0.75532	0.05674	0.87791	0.60397	0.88038	0.86116	0.05674	0.34796	**0.91871**	0.86642
	1	0.74119	0.75364	0.81876	0.80716	0.82864	0.81965	0.82665	0.24084	**0.91473**	**0.91473**
flag	0.1	0.2597	0.06655	0.2218	0.13868	0.14268	0.14573	0.06655	0.12138	0.31378	**0.31398**
	0.7	0.21977	0.06655	0.2205	0.13571	0.1382	0.14176	0.06655	0.12031	**0.31232**	0.27569
	1	0.12787	0.2391	0.15268	0.13549	0.17076	0.17001	0.24299	0.24084	**0.28709**	**0.28709**
german	0.1	0.57516	0.35944	0.57602	0.5219	**0.58811**	0.5854	0.35944	0.46208	0.57323	0.57247
	0.7	0.54573	0.35944	0.57281	0.51857	**0.59267**	0.59057	0.35944	0.43172	0.57175	0.57426
	1	0.3895	0.38493	0.40407	0.37334	0.36717	0.36736	0.35404	0.35404	**0.5969**	**0.5969**
heart	0.1	**0.76571**	0.6405	0.71338	0.69879	0.67343	0.69832	0.64175	0.68189	0.74995	0.74995
	0.7	**0.74487**	0.66551	0.72538	0.69216	0.69781	0.72236	0.66635	0.6776	0.73701	0.72268
	1	0.64107	0.32977	0.42858	0.47927	0.55031	0.54678	0.32641	0.32641	**0.74998**	**0.74998**
japanese credit	0.1	0.77653	0.62878	0.74217	0.77063	0.7745	0.77262	0.62878	**0.80086**	0.78084	0.78116
	0.7	**0.81413**	0.63782	0.70741	0.80831	0.80203	0.80616	0.63782	0.81319	0.78195	0.76258
	1	0.70607	0.36353	0.44061	0.49254	0.5359	0.54993	0.36467	0.36554	**0.75584**	**0.75584**
molecular biology	0.1	0.45006	0.30092	**0.68158**	0.54743	0.54642	0.51235	0.30092	0.40823	0.62742	0.63435
	0.7	0.18284	0.30092	**0.68161**	0.52217	0.54689	0.54447	0.30092	0.364	0.62742	0.47625
	1	0.38381	0.17398	0.39789	**0.47431**	0.27719	0.22743	0.17405	0.17398	0.46948	0.46948
nursery	0.1	0.34878	0.075	0.773	0.70924	0.7625	0.76726	0.075	0.45355	**0.78033**	**0.78033**
	0.7	0.34219	0.075	0.70922	0.63043	0.58198	0.58197	0.075	0.55267	**0.78033**	0.33111
	1	0.34404	0.3361	0.41684	0.42149	0.49159	**0.49174**	0.3361	0.3361	0.33111	0.33111
post operative	0.1	**0.37532**	0.31386	0.34674	0.34483	0.32674	0.32082	0.31386	0.29886	0.30645	0.30645
	0.7	**0.37125**	0.31386	0.34063	0.35297	0.33419	0.32633	0.31386	0.3535	0.30748	0.15175
	1	0.34648	0.3423	0.3623	0.36426	0.36205	**0.36685**	0.34087	0.34087	0.15504	0.15504
tae	0.1	0.27737	0.54896	0.41787	0.54959	0.53362	0.5414	0.57858	**0.58822**	0.54668	0.54668
	0.7	0.29342	0.54896	0.32724	0.40225	0.42256	0.4233	**0.57922**	0.51246	0.55159	0.35133
	1	0.29217	0.28113	0.17504	0.343	0.22983	0.28051	0.31731	0.28113	**0.35503**	**0.35503**
tic tac toe	0.1	0.40253	0.32672	0.86118	0.49864	0.84898	0.84898	0.32672	0.50738	**0.95561**	**0.95561**
	0.7	0.21604	0.32672	0.85408	0.52528	0.85278	0.85278	0.32672	0.61887	**0.95561**	0.57448
	1	0.21598	0.32672	0.34482	0.32672	0.32672	0.32672	0.32672	0.32672	**0.57448**	**0.57448**
zoo	0.1	0.73326	0.47723	**0.89851**	0.47723	0.47723	0.47723	0.47723	0.47723	0.8647	0.80119
	0.7	0.72461	0.47723	**0.89851**	0.47723	0.47723	0.47723	0.47723	0.47723	0.85801	0.86369
	1	0.66473	0.65154	**0.87506**	0.65154	0.65154	0.65154	0.65154	0.65154	0.85744	0.85744

Table C.5 : The Average Class Balanced Accuracy for Different Methods Used to Find the Values for Parameter α

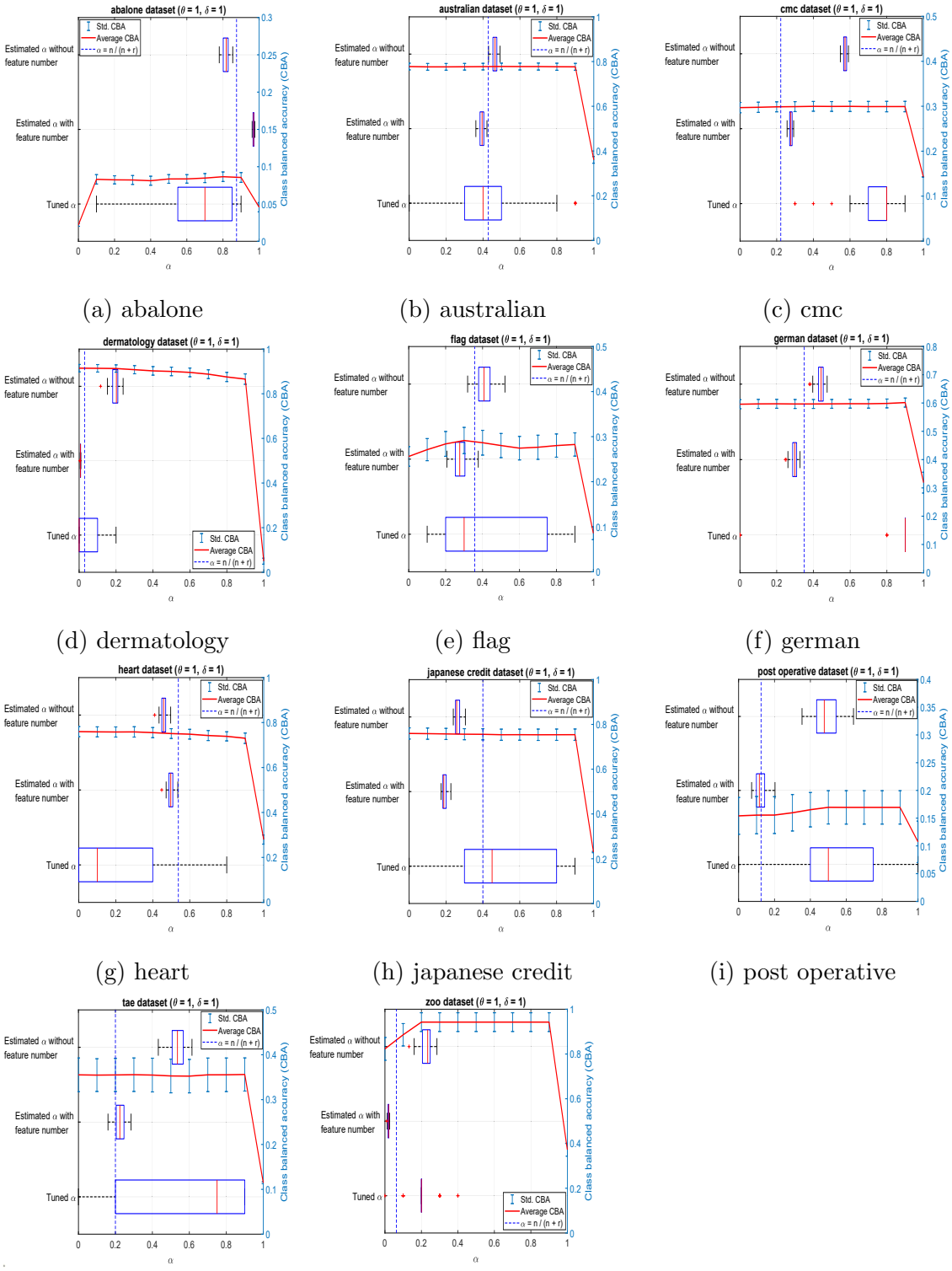
Dataset	Algorithm	$\theta = \delta$	Tuning α	Est-α-v1	Est-α-v2	$\alpha = n/(n+r)$
abalone	EIOL-GFMM-v1	0.1	0.10452	0.10571	0.10589	0.10592
	EIOL-GFMM-v2	0.1	0.10452	0.10571	0.10589	0.10592
	Both	1	0.08795	0.08498	0.08307	0.08545
australian	EIOL-GFMM-v1	0.1	0.7806	0.77979	0.77416	0.77871
	EIOL-GFMM-v2	0.1	0.78038	0.77991	0.77406	0.77871
	Both	1	0.77654	0.77734	0.77784	0.7771
cmc	EIOL-GFMM-v1	0.1	0.43148	0.41052	0.41074	0.40938
	EIOL-GFMM-v2	0.1	0.43168	0.41074	0.41414	0.40936
	Both	1	0.29941	0.29907	0.29967	0.29882
dermatology	EIOL-GFMM-v1	0.1	0.91653	0.91871	0.88963	0.91871
	EIOL-GFMM-v2	0.1	0.91763	0.92217	0.89801	0.92217
	Both	1	0.91446	0.91473	0.91315	0.91473
flag	EIOL-GFMM-v1	0.1	0.3206	0.32098	0.30551	0.31378
	EIOL-GFMM-v2	0.1	0.32063	0.32267	0.30648	0.31398
	Both	1	0.28035	0.28494	0.28583	0.28709
german	EIOL-GFMM-v1	0.1	0.59599	0.57887	0.55533	0.57323
	EIOL-GFMM-v2	0.1	0.59563	0.5794	0.5568	0.57247
	Both	1	0.5998	0.59663	0.59677	0.5969
heart	EIOL-GFMM-v1	0.1	0.73854	0.74783	0.7465	0.74995
	EIOL-GFMM-v2	0.1	0.73854	0.7462	0.74909	0.74995
	Both	1	0.75397	0.75177	0.75383	0.74998
japanese credit	EIOL-GFMM-v1	0.1	0.76721	0.78298	0.77622	0.78084
	EIOL-GFMM-v2	0.1	0.76642	0.78392	0.77529	0.78116
	Both	1	0.75572	0.75755	0.7563	0.75584
post operative	EIOL-GFMM-v1	0.1	0.30712	0.30645	0.33384	0.30645
	EIOL-GFMM-v2	0.1	0.30712	0.30645	0.33384	0.30645
	Both	1	0.1627	0.1555	0.16801	0.15504
tae	EIOL-GFMM-v1	0.1	0.512	0.54668	0.54421	0.54668
	EIOL-GFMM-v2	0.1	0.512	0.54668	0.54421	0.54668
	Both	1	0.35715	0.35499	0.35216	0.35503
zoo	EIOL-GFMM-v1	0.1	0.84196	0.8647	0.85161	0.8647
	EIOL-GFMM-v2	0.1	0.82726	0.79839	0.85708	0.80119
	Both	1	0.91994	0.84655	0.94256	0.85744

Table C.6 : The average class balanced accuracy for the learning algorithms using the hyper-parameter tuning method

Dataset	Onln-GFMM-M1	Onln-GFMM-M2	EIOL-GFMM-v1	EIOL-GFMM-v2
abalone	0.10072	0.09932	**0.10431**	**0.10431**
australian	0.78501	0.79201	**0.79484**	0.78588
cmc	0.39265	0.40692	**0.42634**	0.42522
dermatology	0.87378	0.83522	**0.91563**	0.91497
flag	0.21735	0.27828	**0.30107**	0.28806
german	0.58233	0.55034	**0.60345**	0.59929
heart	0.72481	**0.7772**	0.76922	0.75861
japanese credit	0.763	0.76685	**0.79294**	0.79211
post operative	**0.32856**	0.29945	0.3582	0.31890
tae	**0.48682**	0.4853	0.47482	0.44618
zoo	0.67941	0.8648	**0.87179**	0.85685

Table C.7 : The rank for the learning algorithms using the hyper-parameter tuning method

Dataset	Onln-GFMM-M1	Onln-GFMM-M2	EIOL-GFMM-v1	EIOL-GFMM-v2
abalone	3	4	**1.5**	**1.5**
australian	4	2	**1**	3
cmc	4	3	**1**	2
dermatology	3	4	**1**	2
flag	4	3	**1**	2
german	3	4	**1**	2
heart	4	**1**	2	3
japanese_credit	4	3	**1**	2
post_operative	**1**	4	3	2
tae	**1**	2	3	4
zoo	4	2	**1**	3
Average	3.182	2.909	**1.5**	2.409



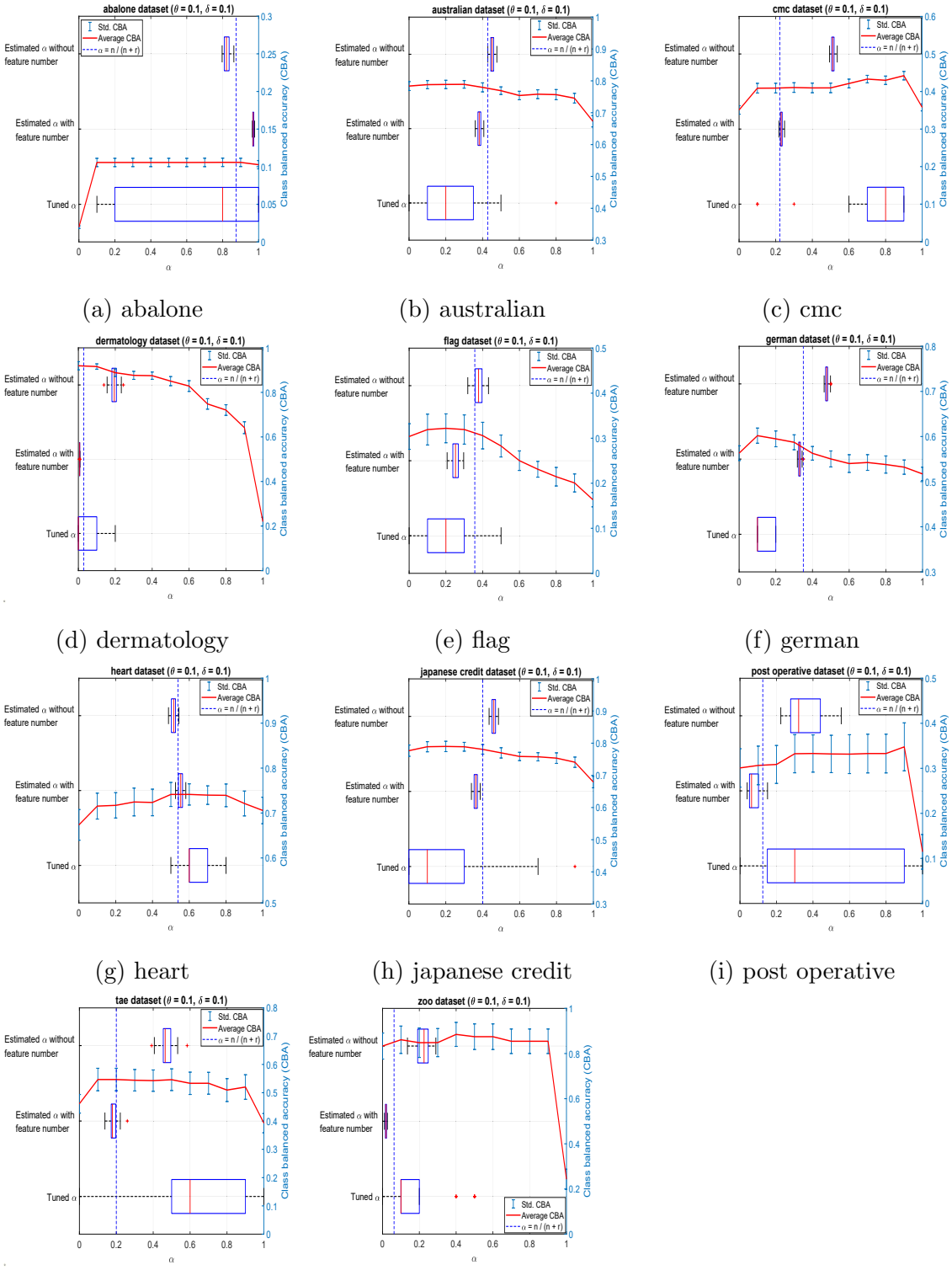
(a) abalone (b) australian (c) cmc

(d) dermatology (e) flag (f) german

(g) heart (h) japanese credit (i) post operative

(j) tae (k) zoo

Figure C.7 : Class balanced accuracy values and range of obtained α for different ways of estimating α ($\theta = 1, \delta = 1$).



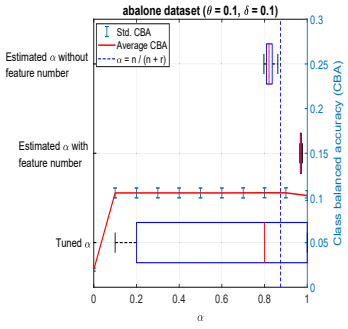
(a) abalone (b) australian (c) cmc

(d) dermatology (e) flag (f) german

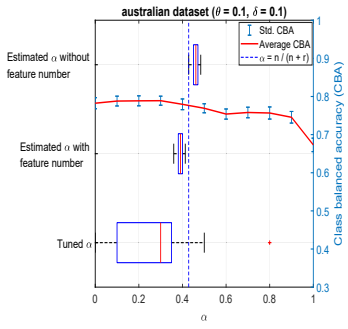
(g) heart (h) japanese credit (i) post operative

(j) tae (k) zoo

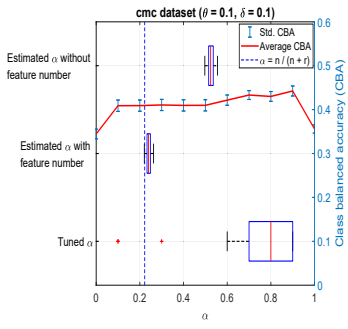
Figure C.8 : Class balanced accuracy values and range of obtained α for different ways of estimating α (using EIOL-GFMM-v1 with $\theta = 0.1, \delta = 0.1$).



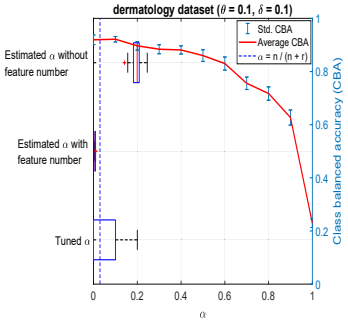
(a) abalone



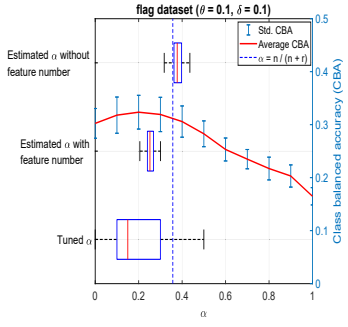
(b) australian



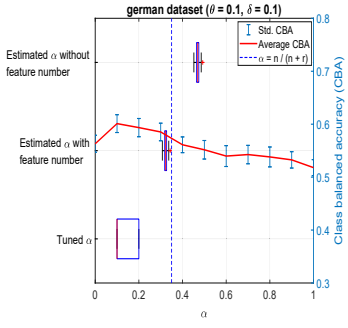
(c) cmc



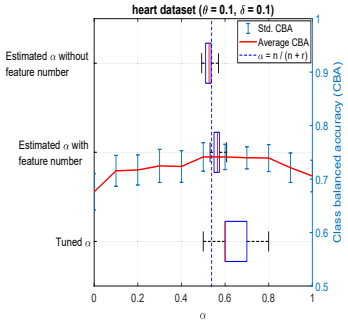
(d) dermatology



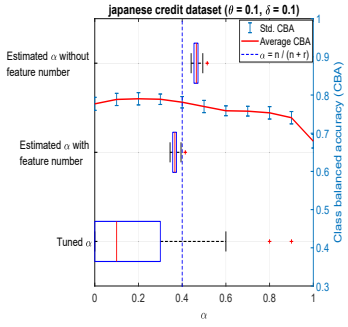
(e) flag



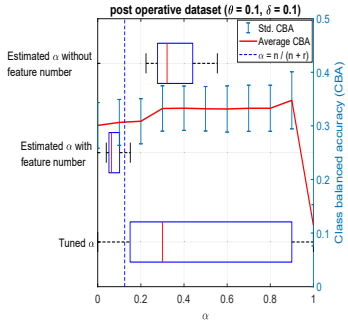
(f) german



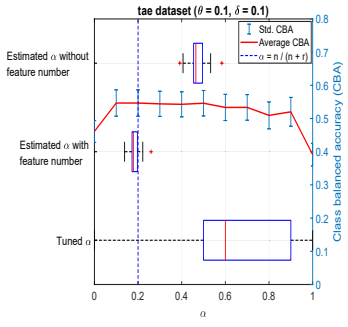
(g) heart



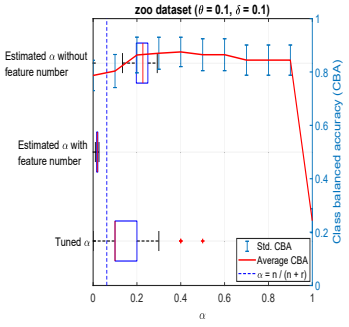
(h) japanese credit



(i) post operative



(j) tae



(k) zoo

Figure C.9 : Class balanced accuracy values and range of obtained α for different ways of estimating α (using EIOL-GFMM-v2 with $\theta = 0.1, \delta = 0.1$).

Appendix D

Additional Results for Chapter 6

D.1 Proof of Lemma 6.1

Proof. It is expected to prove that if the membership degree $b_h(X)$ is below $1-\theta\cdot\gamma_{max}$ then the maximum hyperbox size condition is not satisfied for at least one of the dimensions. First of all, it is desirable to prove that if the membership value for the *j-th* dimension $b_h(x_j) < 1 - \theta \cdot \gamma_j$, then the maximum hyperbox size condition is not met for the *j-th* dimension, i.e., $w_j^{new} - v_j^{new} > \theta$. An assumption for this lemma is that all the dimensions of the input hyperbox $X = [X^l, X^u]$ must satisfy the maximum hyperbox size condition $x_j^u - x_j^l \leq \theta$ and $x_j^u,\ x_j^l \in [0,1]\ \forall j \in [1,n]$. For each *j-th* dimension, there are six cases concerning the positions of the input pattern $X = [X^l, X^u]$ and the hyperbox $h = [V, W]$ as follows:

Case 1: $x_j^l \leq v_j \leq x_j^u \leq w_j$. The membership value along the *j-th* dimension is:

$$b_{hj} = b_h(x_j) = \min([1 - f(x_j^u - w_j, \gamma_j)], [1 - f(v_j - x_j^l, \gamma_j)]) = 1 - \min((v_j - x_j^l)\cdot\gamma_j, 1)$$

Only $(v_j - x_j^l)\cdot\gamma_j \leq 1$ is considered, because in case of $(v_j - x_j^l)\cdot\gamma_j > 1 \Rightarrow b_{hj} = 0$ and $1 < (v_j - x_j^l)\cdot\gamma_j \leq (x_j^u - x_j^l)\cdot\gamma_j \leq \theta\cdot\gamma_j$, thus $1 - \theta\cdot\gamma_j < 0 \Rightarrow b_{hj} = 0 > 1 - \theta\cdot\gamma_j$. Therefore, if $(v_j - x_j^l)\cdot\gamma_j > 1$, the case $b_{hj} < 1 - \theta\cdot\gamma_j$ will never occur. For $(v_j - x_j^l)\cdot\gamma_j \leq 1$, we have:

$$b_{hj} = 1 - (v_j - x_j^l)\cdot\gamma_j$$

If the hyperbox h is expanded, then:

$$v_j^{new} = \min(v_j, x_j^l) = x_j^l; \quad w_j^{new} = \max(w_j, x_j^u) = w_j$$

It is obtained:

$$b_{hj} < 1 - \theta \cdot \gamma_j \Rightarrow 1 - (v_j - x_j^l) \cdot \gamma_j < 1 - \theta \cdot \gamma_j \Rightarrow v_j - x_j^l > \theta \text{ (because of } \gamma_j > 0)$$
$$\Rightarrow w_j - x_j^l > \theta \text{ (because of } w_j \geq v_j) \Rightarrow w_j^{new} - v_j^{new} > \theta$$

Hence, if $b_{hj} < 1 - \theta \cdot \gamma_j$, then $w_j^{new} - v_j^{new} > \theta$ in this case.

Case 2: $v_j \leq x_j^l \leq w_j \leq x_j^u$. The membership value along the *j-th* dimension is:

$$b_{hj} = b_h(x_j) = \min([1 - f(x_j^u - w_j, \gamma_j)], [1 - f(v_j - x_j^l, \gamma_j)]) = 1 - \min((x_j^u - w_j) \cdot \gamma_j, 1)$$

Similarly to case 1, Only $(x_j^u - w_j) \cdot \gamma_j \leq 1$ is considered, thus it can get:

$$b_{hj} = 1 - (x_j^u - w_j) \cdot \gamma_j$$

If the hyperbox h is expanded, then:

$$v_j^{new} = \min(v_j, x_j^l) = v_j; \quad w_j^{new} = \max(w_j, x_j^u) = x_j^u$$

We have:

$$b_{hj} < 1 - \theta \cdot \gamma_j \Rightarrow 1 - (x_j^u - w_j) \cdot \gamma_j < 1 - \theta \cdot \gamma_j \Rightarrow x_j^u - w_j > \theta \text{ (because of } \gamma_j > 0)$$
$$\Rightarrow x_j^u - v_j > \theta \text{ (because of } v_j \leq w_j) \Rightarrow w_j^{new} - v_j^{new} > \theta$$

Hence, if $b_{hj} < 1 - \theta \cdot \gamma_j$, then $w_j^{new} - v_j^{new} > \theta$ in this case.

Case 3: $x_j^l \leq v_j \leq w_j \leq x_j^u$. The membership value along the *j-th* dimension is:

$$\begin{aligned} b_{hj} = b_h(x_j) &= \min([1 - f(x_j^u - w_j, \gamma_j)], [1 - f(v_j - x_j^l, \gamma_j)]) \\ &= \min([1 - (x_j^u - w_j) \cdot \gamma_j], [1 - (v_j - x_j^l) \cdot \gamma_j]) \end{aligned}$$

According to the assumption of the lemma with regard to the input hyperbox, $x_j^u - x_j^l \leq \theta$, thus, $x_j^u - w_j \leq x_j^u - x_j^l \leq \theta$ and $v_j - x_j^l \leq x_j^u - x_j^l \leq \theta$. These lead to $(x_j^u - w_j) \cdot \gamma_j \leq \theta \cdot \gamma_j$ and $(v_j - x_j^l) \cdot \gamma_j \leq \theta \cdot \gamma_j$ (because of $\gamma_j > 0$)

$\Rightarrow 1-(x_j^u - w_j)\cdot\gamma_j \geq 1-\theta\cdot\gamma_j$ and $1-(v_j - x_j^l)\cdot\gamma_j \geq 1-\theta\cdot\gamma_j$. Therefore, $b_{hj} \geq 1-\theta\cdot\gamma_j$. As a result, in this case, $b_{hj} < 1-\theta\cdot\gamma_j$ never happens.

Case 4: $v_j \leq x_j^l \leq x_j^u \leq w_j$, the membership value is computed as: $b_{hj} = 1 \geq 1-\theta\cdot\gamma_j$, and so $b_{hj} < 1-\theta\cdot\gamma_j$ never occurs in this case.

Case 5: $v_j \leq w_j \leq x_j^l \leq x_j^u$. The membership value along the *j-th* dimension is:

$$b_{hj} = b_h(x_j) = \min([1-f(x_j^u - w_j, \gamma_j)], [1-f(v_j - x_j^l, \gamma_j)]) = 1-\min((x_j^u - w_j)\cdot\gamma_j, 1)$$

If the hyperbox h is expanded, then:

$$v_j^{new} = \min(v_j, x_j^l) = v_j; \quad w_j^{new} = \max(w_j, x_j^u) = x_j^u$$

Case 5.1: $(x_j^u - w_j)\cdot\gamma_j > 1 \Leftrightarrow x_j^u - w_j > 1/\gamma_j$ due to $\gamma_j > 0$. In addition, $b_{hj} = 0$. We have:

$$\begin{aligned} b_{hj} < 1-\theta\cdot\gamma_j &\Rightarrow 0 < 1-\theta\cdot\gamma_j \Rightarrow \theta\cdot\gamma_j < 1 \\ &\Rightarrow \theta < 1/\gamma_j < x_j^u - w_j \leq x_j^u - v_j \text{ (due to } v_j \leq w_j \text{ and } \gamma_j > 0) \\ &\Rightarrow \theta < w_j^{new} - v_j^{new} \end{aligned}$$

Hence, if $b_{hj} < 1-\theta\cdot\gamma_j$, then $w_j^{new} - v_j^{new} > \theta$ in this case.

Case 5.2: $(x_j^u - w_j)\cdot\gamma_j \leq 1$, it can lead to:

$$b_{hj} = 1-(x_j^u - w_j)\cdot\gamma_j$$

and

$$\begin{aligned} b_{hj} < 1-\theta\cdot\gamma_j &\Rightarrow 1-(x_j^u - w_j)\cdot\gamma_j < 1-\theta\cdot\gamma_j \Rightarrow x_j^u - w_j > \theta \text{ (due to } \gamma_j > 0) \\ &\Rightarrow x_j^u - v_j > \theta \text{ (because of } v_j \leq w_j) \Rightarrow w_j^{new} - v_j^{new} > \theta \end{aligned}$$

As a result, if $b_{hj} < 1-\theta\cdot\gamma_j$, then $w_j^{new} - v_j^{new} > \theta$ in this case as well.

Case 6: $x_j^l \leq x_j^u \leq v_j \leq w_j$. The membership value along the *j-th* dimension is:

$$b_{hj} = b_h(x_j) = \min([1-f(x_j^u - w_j, \gamma_j)], [1-f(v_j - x_j^l, \gamma_j)]) = 1-\min((v_j - x_j^l)\cdot\gamma_j, 1)$$

If the hyperbox h is expanded, then:

$$v_j^{new} = \min(v_j, x_j^l) = x_j^l; \quad w_j^{new} = \max(w_j, x_j^u) = w_j$$

Case 6.1: $(v_j - x_j^l) \cdot \gamma_j > 1 \Leftrightarrow v_j - x_j^l > 1/\gamma_j$ due to $\gamma_j > 0$. In addition, $b_{hj} = 0$. It is obtained:

$$b_{hj} < 1 - \theta \cdot \gamma_j \Rightarrow 0 < 1 - \theta \cdot \gamma_j \Rightarrow \theta \cdot \gamma_j < 1 \Rightarrow \theta < 1/\gamma_j < v_j - x_j^l \leq w_j - x_j^l \text{ (due to } v_j \leq w_j \text{ an}$$
$$\Rightarrow \theta < w_j^{new} - v_j^{new}$$

Hence, if $b_{hj} < 1 - \theta \cdot \gamma_j$, then $w_j^{new} - v_j^{new} > \theta$ in this case.

Case 6.2: $(v_j - x_j^l) \cdot \gamma_j \leq 1$, it can obtained:

$$b_{hj} = 1 - (v_j - x_j^l) \cdot \gamma_j$$

and:

$$b_{hj} < 1 - \theta \cdot \gamma_j \Rightarrow 1 - (v_j - x_j^l) \cdot \gamma_j < 1 - \theta \cdot \gamma_j \Rightarrow v_j - x_j^l > \theta \text{ (due to } \gamma_j > 0)$$
$$\Rightarrow w_j - x_j^l > \theta \text{ (because of } v_j \leq w_j) \Rightarrow w_j^{new} - v_j^{new} > \theta$$

As a result, if $b_{hj} < 1 - \theta \cdot \gamma_j$, then $w_j^{new} - v_j^{new} > \theta$ in this case as well.

From the six above cases, it can be seen that for each dimension j if $b_{hj} < 1 - \theta \cdot \gamma_j$, then $w_j^{new} - v_j^{new} > \theta$. Given that the membership function for the input hyperbox X is $b_h(X) = \min_{j=1}^{n} b_{hj}$ and $0 \leq \theta \cdot \gamma_j \leq \theta \cdot \max_{j=1}^{n} \gamma_j = \theta \cdot \gamma_{max} \Rightarrow 1 - \theta \cdot \gamma_j \geq 1 - \theta \cdot \gamma_{max}$. Therefore, if $b_{hj} < 1 - \theta \cdot \gamma_{max}$, then $w_j^{new} - v_j^{new} > \theta$ for every dimension j. As a result, if $b_h(X) < 1 - \theta \cdot \gamma_{max}$, then the maximum hyperbox size condition is not satisfied for the expanded hyperbox. The lemma is proved. □

D.2 Proof of Lemma 6.2

Proof. An underlying assumption in this lemma is that all input hyperboxes and aggregatable hyperbox candidates have sizes less than θ. If not, the aggregation process will not be possible. First of all, it is required to prove that if the similarity value $s_{ik} = s(B_i, B_k) < 1 - \theta \cdot \gamma_{max}$, then the maximum hyperbox size constraint is not satisfied for at least one of the dimensions of the hyperbox aggregated from

B_i and B_k. Therefore, it is expected to prove that if $s_{ik}^j < 1 - \theta \cdot \gamma_j$ for any dimension $j \in [1, n]$, then the maximum hyperbox size condition is not met for that $j\text{-}th$ dimension, i.e., $w_j^{new} - v_j^{new} > \theta$. It can be seen that the similarity measure using middle distance between two hyperboxes is the same as the membership value between a hyperbox and an input pattern. Therefore, the proof is the same as in the Appendix D.1. Here, the above condition is only proved for the longest and shortest distance measures.

Using the shortest distance based similarity measure

The shortest distance based similarity value for the $j\text{-}th$ dimension is computed as follows:

$$\widetilde{s}_{ik}^j = \min[1 - f(v_{kj} - w_{ij}, \gamma_j), 1 - f(v_{ij} - w_{kj}, \gamma_j)]$$

For each $j\text{-}th$ dimension, there are six cases concerning the positions of the hyperbox $B_i = [V_i, W_i]$ and the hyperbox $B_k = [V_k, W_k]$ as follows:

Case 1: $v_{ij} \leq v_{kj} \leq w_{ij} \leq w_{kj}$. The similarity value: $\widetilde{s}_{ik}^j = 1 \geq 1 - \theta \cdot \gamma_j$ (because of $\theta \cdot \gamma_j > 0$). Therefore, $\widetilde{s}_{ik}^j < 1 - \theta \cdot \gamma_j$ will never happen in this case.

Case 2: $v_{kj} \leq v_{ij} \leq w_{kj} \leq w_{ij}$. The similarity value: $\widetilde{s}_{ik}^j = 1 \geq 1 - \theta \cdot \gamma_j$. Therefore, $\widetilde{s}_{ik}^j < 1 - \theta \cdot \gamma_j$ will never happen in this case as well.

Case 3: $v_{kj} \leq w_{kj} \leq v_{ij} \leq w_{ij}$. The coordinate at the $j\text{-}th$ dimension of the hyperbox aggregated from B_i and B_k is:

$$v_j^{new} = \min(v_{ij}, v_{kj}) = v_{kj}; \quad w_j^{new} = \max(w_{ij}, w_{kj}) = w_{ij}$$

The similarity value: $\widetilde{s}_{ik}^j = 1 - \min[1, (v_{ij} - w_{kj}) \cdot \gamma_j]$

Case 3.1: $(v_{ij} - w_{kj}) \cdot \gamma_j > 1 \Leftrightarrow v_{ij} - w_{kj} > 1/\gamma_j$ (because of $\gamma_j > 0$). In this case: $\widetilde{s}_{ik}^j = 0$. It leads to:

$\widetilde{s}_{ik}^j < 1 - \theta \cdot \gamma_j \Rightarrow 0 < 1 - \theta \cdot \gamma_j \Rightarrow \theta \cdot \gamma_j < 1$

$\Rightarrow \theta < 1/\gamma_j < v_{ij} - w_{kj} \leq w_{ij} - w_{kj} \leq w_{ij} - v_{kj}$(due to $v_{kj} \leq w_{kj}, v_{ij} \leq w_{ij}, \gamma_j > 0$)

$\Rightarrow \theta < w_j^{new} - v_j^{new}$

Therefore, if $\widetilde{s}_{ik}^{j} < 1 - \theta \cdot \gamma_j$, then $w_j^{new} - v_j^{new} > \theta$ in this case.

Case 3.2: $(v_{ij} - w_{kj}) \cdot \gamma_j \leq 1 \Rightarrow \widetilde{s}_{ik}^{j} = 1 - (v_{ij} - w_{kj}) \cdot \gamma_j$. It can be obtained that:

$$\widetilde{s}_{ik}^{j} < 1 - \theta \cdot \gamma_j \Rightarrow 1 - (v_{ij} - w_{kj}) \cdot \gamma_j < 1 - \theta \cdot \gamma_j \Rightarrow \theta < v_{ij} - w_{kj} \text{ (due to } \gamma_j > 0)$$
$$\Rightarrow \theta < w_{ij} - w_{kj} \leq w_{ij} - v_{kj} = w_j^{new} - v_j^{new} \text{ (because of } v_{kj} \leq w_{kj}; v_{ij} \leq w_{ij})$$

Therefore, if $\widetilde{s}_{ik}^{j} < 1 - \theta \cdot \gamma_j$, then $w_j^{new} - v_j^{new} > \theta$ in this case as well.

Case 4: $v_{ij} \leq w_{ij} \leq v_{kj} \leq w_{kj}$. This case is proved similarly to case 3.

Case 5: $v_{kj} \leq v_{ij} \leq w_{ij} \leq w_{kj}$. The similarity value: $\widetilde{s}_{ik}^{j} = 1 \geq 1 - \theta \cdot \gamma_j$. Therefore, $\widetilde{s}_{ik}^{j} < 1 - \theta \cdot \gamma_j$ will never happen in this case.

Case 6: $v_{ij} \leq v_{kj} \leq w_{kj} \leq w_{ij}$. The similarity value: $\widetilde{s}_{ik}^{j} = 1 \geq 1 - \theta \cdot \gamma_j$. Hence, $\widetilde{s}_{ik}^{j} < 1 - \theta \cdot \gamma_j$ will never happen in this case as well.

Using the longest distance based similarity measure

The longest distance based similarity value for the *j-th* is calculated as follows:

$$\widehat{s}_{ik}^{j} = \min[1 - f(w_{kj} - v_{ij}, \gamma_j), 1 - f(w_{ij} - v_{kj}, \gamma_j)]$$

For each *j-th* dimension, it is also expected to consider in turn six cases relevant to the positions of the hyperbox $B_i = [V_i, W_i]$ and the hyperbox $B_k = [V_k, W_k]$ as follows:

Case 1: $v_{ij} \leq v_{kj} \leq w_{ij} \leq w_{kj}$. The coordinate at the *j-th* dimension of the hyperbox aggregated from B_i and B_k is:

$$v_j^{new} = \min(v_{ij}, v_{kj}) = v_{ij}; \quad w_j^{new} = \max(w_{ij}, w_{kj}) = w_{kj}$$

The similarity value: $\widehat{s}_{ik}^{j} = \min[1 - \min((w_{kj} - v_{ij}) \cdot \gamma_j, 1), 1 - \min((w_{ij} - v_{kj}) \cdot \gamma_j, 1)]$. In this case, we have $w_{kj} - v_{ij} \geq w_{kj} - v_{kj} \geq w_{ij} - v_{kj} \Rightarrow \min((w_{kj} - v_{ij}) \cdot \gamma_j, 1) \geq \min((w_{ij} - v_{kj}) \cdot \gamma_j, 1)$. Therefore, $\widehat{s}_{ik}^{j} = 1 - \min((w_{kj} - v_{ij}) \cdot \gamma_j, 1)$

Case 1.1: $(w_{kj} - v_{ij}) \cdot \gamma_j > 1 \Leftrightarrow w_{kj} - v_{ij} > 1/\gamma_j$ (because of $\gamma_j > 0$). In this

case: $\widehat{s}_{ik}^{j} = 0$. We have:

$$\widehat{s}_{ik}^{j} < 1 - \theta \cdot \gamma_j \Rightarrow 0 < 1 - \theta \cdot \gamma_j \Rightarrow \theta < 1/\gamma_j \text{ (due to } \gamma_j > 0)$$
$$\Rightarrow \theta < w_{kj} - v_{ij} = w_j^{new} - v_j^{new}$$

Therefore, if $\widehat{s}_{ik}^{j} < 1 - \theta \cdot \gamma_j$, then $w_j^{new} - v_j^{new} > \theta$.

Case 1.2: $(w_{kj} - v_{ij}) \cdot \gamma_j \leq 1 \Rightarrow \widehat{s}_{ik}^{j} = 1 - (w_{kj} - v_{ij}) \cdot \gamma_j$. We have:

$$\widehat{s}_{ik}^{j} < 1 - \theta \cdot \gamma_j \Rightarrow 1 - (w_{kj} - v_{ij}) \cdot \gamma_j < 1 - \theta \cdot \gamma_j \Rightarrow \theta < w_{kj} - v_{ij} \text{ (due to } \gamma_j > 0)$$
$$\Rightarrow \theta < w_j^{new} - v_j^{new}$$

Therefore, if $\widehat{s}_{ik}^{j} < 1 - \theta \cdot \gamma_j$, then $w_j^{new} - v_j^{new} > \theta$ in this case.

Case 2: $v_{kj} \leq v_{ij} \leq w_{kj} \leq w_{ij}$. This case is proved similarity to case 1.

Case 3: $v_{kj} \leq w_{kj} \leq v_{ij} \leq w_{ij}$. The coordinate at the *j-th* dimension of the hyperbox aggregated from B_i and B_k is:

$$v_j^{new} = \min(v_{ij}, v_{kj}) = v_{kj}; \quad w_j^{new} = \max(w_{ij}, w_{kj}) = w_{ij}$$

The similarity value: $\widehat{s}_{ik}^{j} = 1 - \min((w_{ij} - v_{kj}) \cdot \gamma_j, 1)$.

Case 3.1: $(w_{ij} - v_{kj}) \cdot \gamma_j > 1 \Leftrightarrow w_{ij} - v_{kj} > 1/\gamma_j$ (because of $\gamma_j > 0$). In this case: $\widehat{s}_{ik}^{j} = 0$. We have:

$$\widehat{s}_{ik}^{j} < 1 - \theta \cdot \gamma_j \Rightarrow 0 < 1 - \theta \cdot \gamma_j \Rightarrow \theta < 1/\gamma_j \text{ (due to } \gamma_j > 0)$$
$$\Rightarrow \theta < w_{ij} - v_{kj} = w_j^{new} - v_j^{new}$$

Therefore, if $\widehat{s}_{ik}^{j} < 1 - \theta \cdot \gamma_j$, then $w_j^{new} - v_j^{new} > \theta$ in this case.

Case 3.2: $(w_{ij} - v_{kj}) \cdot \gamma_j \leq 1 \Rightarrow \widehat{s}_{ik}^{j} = 1 - (w_{ij} - v_{kj}) \cdot \gamma_j$. It can be obtained:

$$\widehat{s}_{ik}^{j} < 1 - \theta \cdot \gamma_j \Rightarrow 1 - (w_{ij} - v_{kj}) \cdot \gamma_j < 1 - \theta \cdot \gamma_j \Rightarrow \theta < w_{ij} - v_{kj} \text{ (due to } \gamma_j > 0)$$
$$\Rightarrow \theta < w_j^{new} - v_j^{new}$$

Therefore, if $\widehat{s}_{ik}^{j} < 1 - \theta \cdot \gamma_j$, then $w_j^{new} - v_j^{new} > \theta$ in this case.

Case 4: $v_{ij} \leq w_{ij} \leq v_{kj} \leq w_{kj}$. This case is proved similarly to case 3.

Case 5: $v_{kj} \leq v_{ij} \leq w_{ij} \leq w_{kj}$. The coordinate at the j-th dimension of the hyperbox aggregated from B_i and B_k is:

$$v_j^{new} = \min(v_{ij}, v_{kj}) = v_{kj}; \quad w_j^{new} = \max(w_{ij}, w_{kj}) = w_{kj}$$

The similarity value: $\widehat{s}_{ik}^{j} = \min[1 - \min((w_{kj} - v_{ij}) \cdot \gamma_j, 1), 1 - \min((w_{ij} - v_{kj}) \cdot \gamma_j, 1)]$.

According to the assumption of the lemma the sizes of the input hyperboxes for the aggregation process must be below θ along each of their n dimensions. Therefore, in this case:

$$\begin{aligned}
w_{kj} - v_{kj} \leq \theta &\Rightarrow 0 \leq w_{kj} - v_{ij} \leq \theta \text{ and } 0 \leq w_{ij} - v_{kj} \leq \theta \ (v_{kj} \leq v_{ij} \text{ and } w_{ij} \leq w_{kj}) \\
&\Rightarrow (w_{kj} - v_{ij}) \cdot \gamma_j \leq \theta \cdot \gamma_j \text{ and } (w_{ij} - v_{kj}) \cdot \gamma_j \leq \theta \cdot \gamma_j \text{ (because of } \gamma_j > 0) \\
&\Rightarrow \min((w_{kj} - v_{ij}) \cdot \gamma_j, 1) \leq \min(\theta \cdot \gamma_j, 1) \\
&\text{and } \min((w_{ij} - v_{kj}) \cdot \gamma_j, 1) \leq \min(\theta \cdot \gamma_j, 1) \\
&\Rightarrow 1 - \min((w_{kj} - v_{ij}) \cdot \gamma_j, 1) \geq 1 - \min(\theta \cdot \gamma_j, 1) \\
&\text{and } 1 - \min((w_{ij} - v_{kj}) \cdot \gamma_j, 1) \geq 1 - \min(\theta \cdot \gamma_j, 1) \\
&\Rightarrow \widehat{s}_{ik}^{j} \geq 1 - \min(\theta \cdot \gamma_j, 1) \geq 1 - \theta \cdot \gamma_j
\end{aligned}$$

Therefore, the input hyperboxes size assumption always leads to $\widehat{s}_{ik}^{j} \geq 1 - \theta \cdot \gamma_j$. As a result, $\widehat{s}_{ik}^{j} < 1 - \theta \cdot \gamma_j$ will never occur in this case.

Case 6: $v_{ij} \leq v_{kj} \leq w_{kj} \leq w_{ij}$. This case is proved similarly to case 5.

From the above proofs, it can be seen that if the similarity value $s_{ik}^{j} < 1 - \theta \cdot \gamma_j$; $\forall j \in [1, n]$, then the hyperbox aggregated from two hyperboxes B_i and B_k does not satisfy the maximum hyperbox size condition on the j-th dimension. We also have $0 \leq \theta \cdot \gamma_j \leq \theta \cdot \gamma_{max} \Rightarrow 1 - \theta \cdot \gamma_j \geq 1 - \theta \cdot \gamma_{max}; \forall j \in [1, n]$. Therefore, if the similarity score between two hyperboxes $s_{ik} < 1 - \theta \cdot \gamma_{max}$, then the maximum hyperbox size condition is not satisfied for at least one of the dimensions of the aggregated hyperbox. In addition, in the agglomerative learning, two hyperboxes B_i and B_k are aggregated if their similarity value $s_{ik} \geq \sigma$, where σ is a given minimum similarity threshold. From these two conditions, it is sufficient to consider only pairs of hyperboxes with similarity values $s_{ik} \geq \max(\sigma, 1 - \theta \cdot \gamma_{max})$ when selecting hyperbox canditates for the aggregation process. Lemma 6.2 is proved. □

D.3 Experimental Datasets and Parameter Settings

To evaluate the effectiveness of the proposed method, the experiments on 24 datasets taken from the UCI machine learning repository (Dua and Graff 2019) were conducted. A summary of these datasets related to the numbers of classes, features, and samples is shown in Table D.1. As discussed in the complexity analysis parts, the complexity of all learning algorithms depends on both the number of features and the number of hyperboxes considered during the training process. The proposed acceleration methods in section 6.2 target reducing only the number of hyperboxes considered. Therefore, it is appropriate to keep a high number of dimensions when assessing the impact of number of considered hyperboxes on the acceleration of the proposed methods in comparison to the original algorithms. As a result, several high dimensional datasets used in Chapters 3 and 4 were combined with other high dimensional datasets to build a set of experimental datasets shown in Table D.1. It is noted that the proposed method in this Chapter only fits for datasets with only numerical features, thus the datasets in this Chapter are different from those used in Chapter 5.

For each dataset, 5 times 2-fold cross-validation were carried out, and then the average values of the training time and the number of hyperbox candidates considered during the training process are reported in this chapter. Experiments were executed on a Intel Xeon Gold 6150 2.7GHz computer with 32GB RAM running Red Hat Enterprise Linux. The algorithms were implemented using Python programming language.

The sensitivity parameter γ_j impacts the decreasing speed of the membership function on the j -th dimension. If a large value of γ_j is set, there may be cases that samples are not classified correctly as the membership values for all classes are zero. Therefore, to avoid this situation, the sensitivity parameter $\gamma_j = 1; \forall j \in [1, n]$ was used as recommended in Abe (2001) when all the input data were normalized to the range of [0, 1]. If the maximum hyperbox size parameter θ is assigned a large value, the classification accuracy of the GFMMNN is negatively affected. A high classification accuracy is usually achieved for a small value of θ but it significantly increases

Table D.1 : The summary of the used datasets for the second solution of accelerating learning algorithms

ID	Dataset	# samples	# features	# classes
1	blance scale	625	4	3
2	banknote authentication	1372	4	2
3	blood transfusion	748	4	2
4	breast cancer wisconsin	699	9	2
5	breast cancer coimbra	116	9	2
6	climate model crashes	540	18	2
7	connectionist bench sonar	208	60	2
8	glass	214	9	6
9	haberman	306	3	2
10	heart	270	13	2
11	ionosphere	351	33	2
12	movement libras	360	90	15
13	optical digit	5620	62	10
14	page blocks	5473	10	2
15	pendigits	10992	16	10
16	pima diabetes	768	8	2
17	plant species leaves margin	1600	64	100
18	plant species leaves texture	1600	64	100
19	ringnorm	7400	20	2
20	seeds	210	7	3
21	image segmentation	2310	19	7
22	spambase	4601	57	2
23	spectf heart	267	44	2
24	landsat satellite	6435	36	6

the training time (Khuat and Gabrys 2020). Therefore, to show the efficiency of the proposed method, a small value of $\theta = 0.1$ was used for learning algorithms in this experiment. In the agglomerative learning algorithms, the minimum similarity threshold $\sigma = 0$ was set to assess the impact of the lower bound related to θ on the training time of algorithms. For $\sigma = 0$, the hyperbox aggregation step depends only on the maximum hyperbox size. The analysis of the impacts of different parameters on the classification performance of GFMM learning algorithms was presented in Chapter 3.

D.4 Training Time of Algorithms with and without using Lemmas

This appendix subsection shows the training time of online and agglomerative learning algorithms from the experiments in section 6.2.4 in Chapter 6. Table D.2 shows the training time of the IOL-GFMM and original online learning algorithms. Table D.3 presents the training time of the AGGLO-2 algorithm, while Table D.4 decribes the learning time of the AGGLO-SM algorithm.

Table D.2 : Training time of online learning algorithms in Chapter 6

Dataset	**IOL-GFMM**		**Onln-GFMM**	
	w/o. lemma	**w. lemma**	**w/o. lemma**	**w. lemma**
blance scale	0.1465	0.0256	0.1479	0.0274
banknote authentication	0.0984	0.0756	0.471	0.4459
blood transfusion	0.0497	0.0398	0.1322	0.1251
breast cancer wisconsin	0.1205	0.0311	0.121	0.0331
breast cancer coimbra	0.0099	0.0042	0.0101	0.0051
climate model crashes	0.2093	0.0335	0.2168	0.0358
connectionist bench sonar	0.03	0.0106	0.029	0.0113
glass	0.0141	0.0107	0.0382	0.0347
haberman	0.0212	0.0158	0.0469	0.0418
heart	0.0402	0.0113	0.0407	0.0131
ionosphere	0.0584	0.0211	0.091	0.0563
movement libras	0.0271	0.0174	0.0578	0.0526
optical digit	4.9332	1.0854	3.7477	1.2472
page blocks	0.8802	0.6761	4.8526	4.64
pendigits	14.4605	3.3734	179.4647	168.4146
pima diabetes	0.406	0.0821	0.6995	0.3686
plant species leaves margin	0.2702	0.1327	0.1768	0.1357
plant species leaves texture	4.3093	0.5983	4.5103	0.6081
ringnorm	38.1094	2.9893	54.7914	18.037
seeds	0.0315	0.0183	0.0714	0.0595
image segmentation	0.7204	0.3984	10.2108	9.8831
spambase	5.567	1.6777	17.4908	13.4731
spectf heart	0.1039	0.027	0.1033	0.0275
landsat satellite	6.4317	2.3303	58.5758	54.5749
Average	**3.210358**	**0.570238**	**14.00407**	**11.34798**

Table D.3 : Training time of the AGGLO-2 algorithm in Chapter 6

Dataset	**Longest distance**		**Shortest distance**		**Mid-max distance**		**Mid-min distance**	
	w/o lemma	**w/ lemma**	**w/o lemma**	**w/ lemma**	**w/o lemma**	**w/ lemma**	**w/o lemma**	**w/ lemma**
blance scale	0.883	0.0278	0.8819	0.0276	0.8949	0.0423	0.8928	0.0421
banknote authentication	0.6161	0.174	0.6153	0.2315	0.6686	0.2542	0.6675	0.243
blood transfusion	0.4673	0.1078	0.3892	0.1421	0.417	0.154	0.4842	0.1499
breast cancer wisconsin	2.74	0.1284	2.7136	0.1282	2.8078	0.191	2.7892	0.1914
breast cancer coimbra	0.0712	0.008	0.0713	0.0079	0.0764	0.0126	0.0761	0.0126
climate model crashes	1.4255	0.0438	1.4104	0.0438	1.4442	0.069	1.4435	0.0691
connectionist bench sonar	0.1423	0.0117	0.1421	0.0118	0.1511	0.0193	0.1509	0.0196
glass	0.0859	0.0205	0.0849	0.0231	0.0942	0.0318	0.0946	0.0299
haberman	0.1477	0.0291	0.1249	0.0356	0.1351	0.0429	0.1562	0.0414
heart	0.3037	0.0173	0.3014	0.0172	0.3149	0.0275	0.3127	0.0275
ionosphere	0.832	0.0572	0.6701	0.0524	0.6898	0.0771	0.8597	0.0887
movement libras	0.1728	0.045	0.142	0.0406	0.1612	0.0575	0.1963	0.0661
optical digit	323.1978	1.184	324.0414	1.2508	324.5103	1.8357	331.7654	1.8095
page blocks	14.5953	2.6135	10.8555	3.1821	13.2088	3.7672	15.3054	3.7015
pendigits	597.5753	7.7973	596.2356	10.0636	599.6309	11.7277	599.684	10.5186
pima diabetes	6.3228	0.1788	4.9712	0.1767	5.0455	0.2371	6.3992	0.269
plant species leaves margin	0.6798	0.117	0.6807	0.1189	0.7446	0.1805	0.7451	0.1806
plant species leaves texture	119.9449	1.9998	120.5406	2.0027	122.1148	3.2466	121.6378	3.2535
ringnorm	1350.8695	8.7521	1352.427	12.2458	1357.874	15.1354	1354.012	13.4766
seeds	0.174	0.0279	0.1727	0.0331	0.1849	0.0442	0.187	0.0406
image segmentation	11.882	0.8685	10.3135	1.0592	10.6223	1.2304	12.1874	1.204
spambase	441.0555	8.4443	396.2303	13.5323	400.5978	13.9315	446.2259	12.7619
spectf heart	0.6437	0.0311	0.6436	0.0311	0.6632	0.0502	0.6646	0.0502
landsat satellite	255.4462	5.5045	242.8039	15.0677	243.0923	11.9613	307.8871	9.4819
Average	**130.428**	**1.591**	**127.811**	**2.480**	**128.589**	**2.680**	**133.534**	**2.405**

Table D.4 : Training time of the AGGLO-SM algorithm in Chapter 6

Dataset	**Longest distance**		**Shortest distance**		**Mid-max distance**		**Mid-min distance**	
	w/o lemma	**w/ lemma**	**w/o lemma**	**w/ lemma**	**w/o lemma**	**w/ lemma**	**w/o lemma**	**w/ lemma**
blance scale	0.3724	0.0207	0.3702	0.0206	0.3704	0.0207	0.3702	0.0207
banknote authentication	20.4138	18.0293	20.9532	18.1422	20.5455	17.8608	20.6175	17.722
blood transfusion	0.5674	0.4925	0.8059	0.7081	0.7212	0.6478	0.6057	0.5409
breast cancer wisconsin	0.3999	0.1324	0.4103	0.1335	0.3982	0.1358	0.3998	0.1319
breast cancer coimbra	0.0233	0.0087	0.0234	0.0087	0.0233	0.0087	0.0233	0.0087
climate model crashes	0.6912	0.0319	0.6951	0.0324	0.6944	0.032	0.6914	0.0323
connectionist bench sonar	0.0678	0.0091	0.0676	0.0091	0.0678	0.0091	0.0679	0.0091
glass	0.1718	0.1578	0.1733	0.1623	0.1686	0.1585	0.1738	0.1613
haberman	0.1577	0.1375	0.1893	0.1689	0.1893	0.1674	0.1598	0.1387
heart	0.0212	0.0123	0.021	0.0123	0.0233	0.0139	0.0208	0.0123
ionosphere	0.4828	0.3937	0.5086	0.3863	0.4907	0.3896	0.4833	0.3781
movement libras	0.2215	0.2043	0.225	0.2049	0.2218	0.2046	0.2208	0.2036
optical digit	105.8615	1.2225	103.9479	1.2399	98.2648	1.2936	104.8599	1.249
page blocks	494.1239	422.8161	1409.174	1206.817	1156.35	1015.346	511.3471	439.3924
pendigits	130.6623	38.7196	267.1413	158.6442	250.7539	157.6118	140.068	46.2601
pima diabetes	3.7218	2.3679	3.9888	2.6727	3.932	2.5885	3.7252	2.3731
plant species leaves margin	0.3564	0.0967	0.3577	0.0966	0.3562	0.0966	0.3562	0.0967
plant species leaves texture	40.4664	12.0927	40.5373	12.1132	40.6338	12.1003	40.4593	12.0958
ringnorm	1327.968	872.3743	1442.132	986.0059	1382.036	929.1679	1332.649	874.2278
seeds	0.3235	0.279	0.3286	0.2895	0.3259	0.2859	0.3197	0.279
image segmentation	62.3063	58.843	72.7162	69.1137	71.0587	67.3851	63.6872	60.2109
spambase	1506.906	1412.081	1642.779	1543.002	1621.651	1521.519	1516.833	1427.837
spectf heart	0.3101	0.0246	0.3102	0.0246	0.3101	0.0247	0.3102	0.0247
landsat satellite	237.7967	184.5571	965.5685	884.7285	908.0397	826.8151	291.1204	236.4152
Average	**163.933**	**126.046**	**248.893**	**203.531**	**231.568**	**189.745**	**167.899**	**129.993**

Appendix E

Additional Results for Chapter 7

E.1 Implementation of the MRHGRC

E.1.1 An Example of the MRHGRC

This sub-section is to provide an example for the proposed method in Chapter 7. The illustration of the proposed approach can be found in Figure E.1. Two-class datasets with 400 training samples, 200 testing samples, and 100 validation patterns were generated based on Gaussian distribution. The dotted points in respective figures surrounding the hyperboxes are testing points. Based on probability density functions used to generate the data, the lowest classification error on the testing set is 11%. Assuming using four processes to build hyperboxes in parallel. The training set is split into four parts and delivered to four processes to form four hyperbox-based classifiers independently. For each model, it is easy to observe that the decision boundary is complex, in which small hyperboxes can occupy relatively large influencing regions such as A, B, C, and D zones in the figure. These regions cause a misclassification for each model. However, these influencing regions are narrowed down in size after the merging step because evidence from the surrounding correct hyperboxes is sufficient to form new decision boundaries as well as reducing the impact of hyperboxes causing misclassification. Therefore, the error rate of the merged classifier decreases. Nonetheless, the complexity of the aggregated model increases and many hyperboxes may contribute to the predictive results; even some noisy hyperboxes can lead to the decline in the classification accuracy of the model. As a result, a pruning step is performed using a validation set, and hyperboxes with the accuracy lower than 50% are eliminated. It can be observed that the error rate of the model decreases. In phase 2, the minimum hyperbox sizes are increased to reduce the complexity of the classifier, and overlapping regions among hyperboxes

representing different classes are resolved. As shown, the accuracy of the model only changes a little, while the complexity is significantly reduced. This example demonstrates the efficiency of the proposed method.

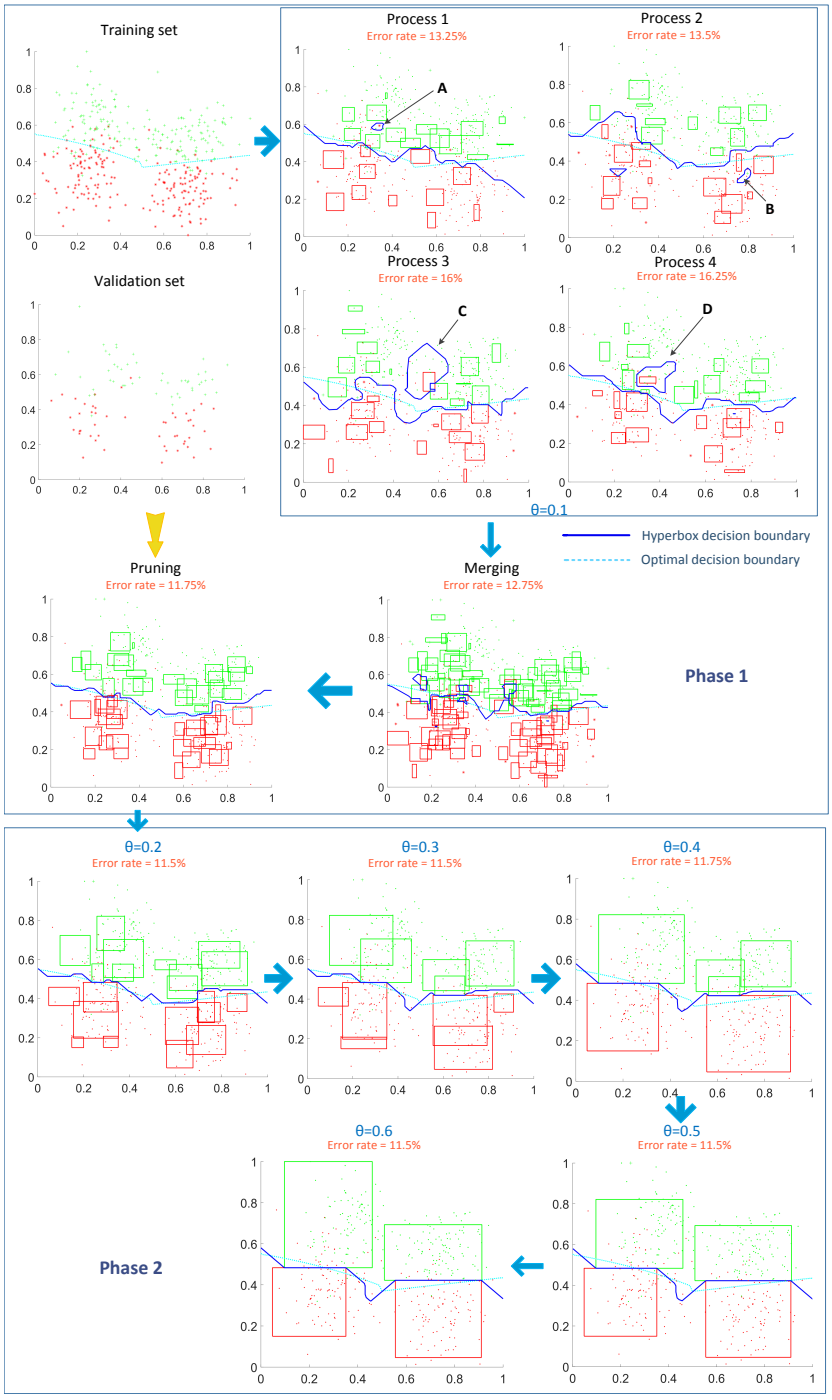


Figure E.1 : Demonstration of the training process proposed in Chapter 7

Figure E.2 shows the aggregation process of hyperboxes through different levels of granularity for the above example. The numbers in the right-hand side of the figure show the numbers of generated hyperboxes for each granularity level specified by the maximum hyperbox size threshold θ. The value of $\theta \leq 0.6$ in this experiment is only used to illustrate the proposed method. If the value of θ larger than 0.6 is passed to the algorithm, phase 2 will continue to be performed to aggregate and merge hyperboxes when the constraints are still met. When the value of θ is close to 1, and the minimum membership value between hyperbox candidates is close to 0, it is possible that all samples in the same class will be covered by only one hyperbox.

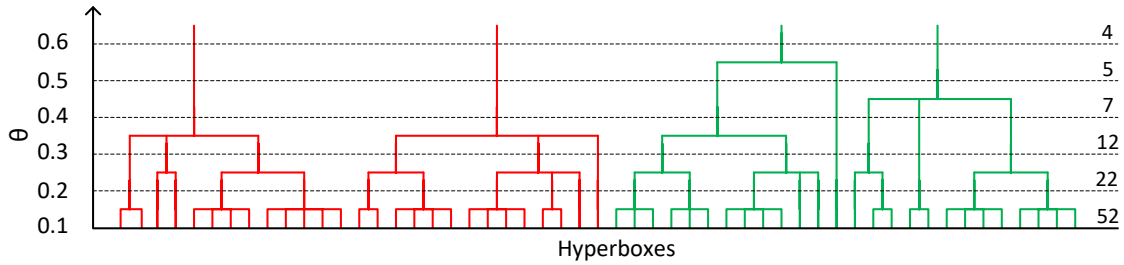


Figure E.2 : An dendrogram showing the changes in the number of hyperboxes through different levels of granularity for the example in Figure E.1

The next subsections present the detailed steps for implementation of the proposed method. The readers can find the source code at: https://github.com/UTS-AAi/MRHGRC.

E.1.2 An Implementation of the MRHGRC

This sub-section is relevant to section 7.2 in Chapter 7. It provides the readers with some of the implementational aspects including the free text description, pseudo-codes, and implementation pipeline.

Phase 1

The steps of phase 1 are described in Algorithm E.1. First of all, the training set which is loaded from the input file using heterogeneous or homogeneous mode is separated into many disjoint stratified groups. The number of groups is equal to the

Algorithm E.1 Phase 1 of the proposed method

Input:

ckS: the size of each data chunk loaded from the input file; $path$: the path to the file containing the input data; $pathVal$: the path to the file containing the validation data; $dType$: type of distributing data to each worker (homogeneous or heterogeneous mode); α: minimum accuracy of each hyperbox to be trained after pruning; n_w: number of workers (processes); θ_0: maximum hyperbox size in phase 1

Output:

A list $\mathbf{H}_0$ of hyperbox fuzzy sets with minimum and maximum coordinates and classes

1: $\mathbf{X} \leftarrow$ **ReadInputData**$(path, ckS)$ and group data by classes if $dType$ is the homogeneous mode
2: **for** parallel worker p in $[1, n_w]$ **do**
3: **if** the first iteration **then**
4: **if** $dType$ is heterogeneous mode **then**
5: Initialize an empty list of hyperboxes for each worker: min-max values $V[p] = W[p] = \varnothing$, hyperbox classes: $L[p] = \varnothing$
6: **else**
7: Initialize a hashtable containing the empty lists for each class: $V\{c\}[p] = W\{c\}[p] = L\{c\}[p] = \varnothing$ for c in the list of classes $\mathcal{C}$ in $\mathbf{X}$
8: **end if**
9: **end if**
10: **if** $dType$ is the heterogeneous mode **then**
11: $\mathbf{X}[p] \leftarrow$ Get data for worker p from $\mathbf{X}$
12: $V[p], W[p], L[p] \leftarrow$ **HyperboxExpansionOrCreateNewAndCentroidConstruction** $(p, \mathbf{X}[p], V[p], W[p], L[p], \theta_0)$
13: **else**
14: $\mathbf{X}\{c\}[p] \leftarrow$ Get data for worker p from $\mathbf{X}$ for each class c in the list of classes in $\mathbf{X}$
15: $V\{c\}[p], W[p], L[p] \leftarrow$ **HyperboxExpansionOrCreateNewAndCentroidConstruction** $(p, \mathbf{X}\{c\}[p], V\{c\}[p], W\{c\}[p], L\{c\}[p], \theta_0), \forall c \in \mathcal{C}$
16: **end if**
17: **end for**
18: go to step 1 if there are remaining data in the input data file
19: **if** $dType$ is the heterogeneous mode **then**
20: $[\mathcal{V}, \mathcal{W}, \mathcal{L}] \leftarrow$ **Merge**$(V[p], W[p], L[p])$ $\forall p \in [1, n_w]$ using Eq. (7.2)
21: **else**
22: $[\mathcal{V}, \mathcal{W}, \mathcal{L}] \leftarrow$ **Merge**$(V\{c\}[p], W\{c\}[p], L\{c\}[p])$, $\forall p \in [1, n_w], \forall c \in \mathcal{C}$ using Eq. (7.2)
23: **end if**
24: $[\mathcal{V}, \mathcal{W}, \mathcal{L}] \leftarrow$ **RemoveContainedHyperboxesAndUpdateCentroid**$(\mathcal{V}, \mathcal{W}, \mathcal{L})$ using Eq. (7.3)
25: $\mathbf{X}_V \leftarrow$ **ReadValidationData**$(pathVal)$
26: $[\mathcal{V}, \mathcal{W}, \mathcal{L}] \leftarrow$ **Pruning**$(\mathcal{V}, \mathcal{W}, \mathcal{L}, \alpha, \mathbf{X}_V)$ using Eqs. (7.4) and (7.5)
27: **return** $\mathbf{H}_0 = [\mathcal{V}, \mathcal{W}, \mathcal{L}]$

number of initialized processes (CPU cores). In the case that the size of the training set is large, all data should not be fetched into the main memory. In this case, it is expected to load data in chunks with the pre-determined size. The disjoint stratified

groups are formed from these chunks. After that, these groups will be distributed to workers on separated CPU cores. Each process executes an incremental learning procedure associated with determining the centroid of patterns of each constructed hyperbox (*lines 10-16* in Algorithm E.1). If the homogeneous mode is used, then the process of building hyperboxes only includes the verification of maximum hyperbox size, expansion of the hyperbox with the highest membership grade (if the condition is met) or generation of new hyperboxes. Otherwise, if the heterogeneous mode is deployed, firstly it is necessary to filter hyperboxes representing the same class as the input sample before performing the process of hyperbox expansion/generation. The operations of checking the expansion criterion and choosing the suitable hyperboxes are only conducted on the obtained hyperboxes of the filtering process. The expansion criterion and steps of expanding/generating hyperboxes are the same as those in the learning algorithm of the GFMMNN. When a new pattern X is presented to the classifier, the operation of building the pattern centroid for each hyperbox is performed according to Eq. (7.1) in Chapter 7.

After entire data in the training file are read, and the process of building hyperboxes on workers finished, all constructed hyperboxes are joint into a data structure on the main thread to form a single model (*lines 19-23* in Algorithm E.1). Then, it is required to merge all hyperboxes contained in other hyperboxes with the same label, and update the pattern centroid of larger hyperbox using Eq. (7.3).

The process continues with the pruning step using a validation set. The validation patterns are put through the constructed model to compute the numbers of samples which are accurately predicted and misclassified by each hyperbox fuzzy set. Next, hyperboxes with the value of prediction accuracy being lower than a predefined threshold (50% in this work) are eliminated. It is noted that there are some hyperboxes which do not join the classification process on the validation set (their accuracy is zero). There are two solutions for these hyperboxes, i.e., eliminating or keeping them. If the removal of these hyperboxes leads to a better value of averaged accuracy over all classifiers on the validation set compared to the case of retaining them, these hyperboxes will be pruned; otherwise, they are retained. The final set

of hyperboxes is called phase-1 hyperbox fuzzy sets.

Phase 2

Algorithm E.2 describes the steps of phase 2 in the proposed method. For each given level of granularity, previously generated hyperboxes are pushed sequentially through the process of hyperbox aggregation. Firstly, it is required to filter hyperboxes representing the same label as the input hyperbox X_h among all hyperboxes at the current level of granularity, and then computing their membership values with the input hyperbox. Next, the values of membership are sorted in descending order (*lines 5-7* in Algorithm E.2). The hyperbox forming the highest membership degree with the input pattern is selected to verify the maximum hyperbox size and the minimum value of membership m_s. If the expansion criteria are met, the selected hyperbox will be expanded to cover the input sample. After that, the overlap between the expanded hyperbox and the hyperboxes belonging to other classes is checked. If no overlap occurs, the expanded hyperbox will replace its previous versions, the centroid of newly aggregated hyperbox is updated using Eq. (7.3) in Chapter 7, and the process continues with another input hyperbox (*lines 9-21*). Otherwise, the hyperbox with the second highest membership is chosen, and the steps of expansion, overlap test, and replacing hyperbox are repeated until there is a satisfied hyperbox or no hyperbox to be selected. If no hyperbox in the current set of hyperboxes can expand to cover the input hyperbox X_h, hyperbox X_h will be added to the current list of hyperboxes. Then, the overlap test is conducted, and if there exists any overlapping region, the contraction process is deployed to deal with overlap (*lines 22-28*). The steps of the contraction process are the same as those in the general fuzzy min-max neural network shown in section 3.2.2 in Chapter 3.

E.2 Experimental results on real datasets

This part is relevant to the subsection 7.3.2 in Chapter 7. Several experimental results of the proposed method on the real datasets will be shown in this part.

Twelve datasets with diverse ranges of numbers of sizes, dimensions, and classes

Algorithm E.2 Phase 2 of the proposed method

Input:

$\mathbf{H}_0$: a set of hyperboxes created in phase 1; Θ: a list of maximum hyperbox sizes; m_s: the minimum membership degree of two aggregated hyperboxes

Output:

A list $\mathbf{H}_H(\Theta)$ of higher-level hyperbox fuzzy sets with minimum-maximum points and classes

$i \leftarrow 1$
for each $\theta \in \Theta$ **do**
  Initialize an empty list of hyperboxes: $\mathbf{H}_i \leftarrow [\mathcal{V} = \varnothing, \mathcal{W} = \varnothing, \mathcal{L} = \varnothing]$
  for each hyperbox $X_h \in \mathbf{H}_{i-1}$ **do**
    $H_s \leftarrow$ Filter hyperboxes in $\mathbf{H}_i$ representing the same class with X_h
    $F \leftarrow$ **ComputeMembershipValue**(H_s, X_h) using Eq. (3.1)
    $Index(H_s) \leftarrow$ **SortDescending**(F)
    $isAdjust \leftarrow$ **false**
    for each $i \in Index(H_s)$ **do**
      $e \leftarrow$ **ExpansionConditionChecking** $(H_s[i], X_h, \theta, m_s)$ Eqs. (3.4) and (7.8)
      if e is **true then**
        $H_1 = [V_1, W_1, L_1] \leftarrow$ Expand hyperbox $H_s[i]$ to cover X_h using Eqs. (3.5) and (3.5)
        $o \leftarrow$ **IsOverlap**$(H_1, \mathbf{H}_i \setminus H_s)$
        if o is **false then**
          $isAdjust =$ **true**
          Update the sample centroids of newly expanded hyperbox using Eq. (7.3)
          Replace $H_s[i]$ by H_1
          break
        end if
      end if
    end for
    if $isAdjust =$ **false then**
      $\mathbf{H}_i \leftarrow \mathbf{H}_i \cup X_h$
      $O \leftarrow$ **OverlapTest**$(X_h, \mathbf{H}_i \setminus H_s)$
      if $|O| > 0$ **then**
        Contraction(X_h, O)
      end if
    end if
  end for
  $\mathbf{H}_H(\Theta) \leftarrow \mathbf{H}_H(\Theta) \cup \mathbf{H}_i$
  $i \leftarrow i + 1$
end for
return $\mathbf{H}_H(\Theta)$

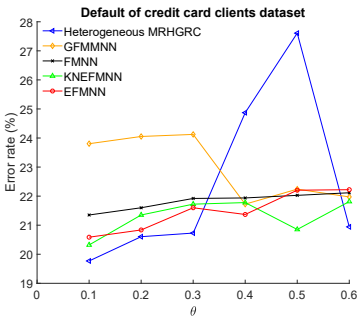
were used. These datasets were taken from the LIBSVM (Chang and Lin 2011), Kaggle (Kaggle 2019), and UCI repositories (Dua and Graff 2019) and their properties are described in Table E.1. For the *susy* dataset, the last 500,000 patterns were

used for the test set as shown in Baldi et al. (2014).

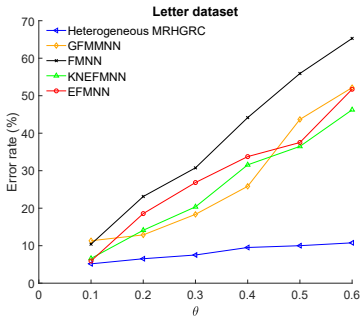
Table E.1 : The real datasets and their statistics for experiments in Chapter 7

Dataset	**#Dimensions**	**#Classes**	**#Training**	**#Validation**	**#Testing**	**Source**
Poker Hand	10	10	25,010	50,000	950000	LIBSVM
SensIT Vehicle	100	3	68,970	9,852	19,706	LIBSVM
Skin_NonSkin	3	2	171,540	24,260	49,257	LIBSVM
Covtype	54	7	406,709	58,095	116,208	LIBSVM
White wine quality	11	7	2,449	1,224	1,225	Kaggle
PhysioNet MIT-BIH Arrhythmia	187	5	74,421	13,133	21,892	Kaggle
MAGIC Gamma Telescope	10	2	11,887	3,567	3,566	UCI
Letter	16	26	15,312	2,188	2,500	UCI
Default of credit card clients	23	2	18,750	5,625	5,625	UCI
MoCap Hand Postures	36	5	53,104	9,371	15,620	UCI
MiniBooNE	50	2	91,044	12,877	26,143	UCI
SUSY	18	2	4,400,000	100,000	500,000	UCI

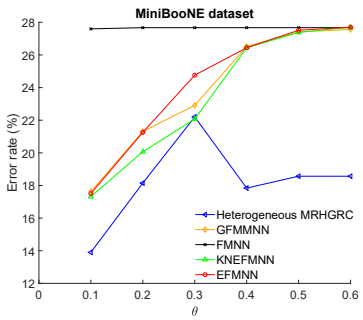
Figure E.3 shows the error rate of classifiers on several real testing datasets with the change in the data abstraction levels. Table E.2 presents the minimum error rates of classifiers on the validation (E_V) and testing (E_T) sets as well as the total running time through six levels of granularity. Moreover, the granularity level which results in the minimum error rate for each dataset will be shown. In addition, the error on the testing set if the model is trained using the best granularity level on the validation set is also mentioned.



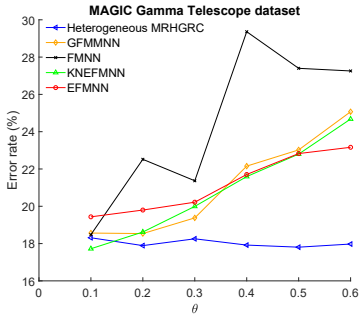
(a) Default of credit card clients



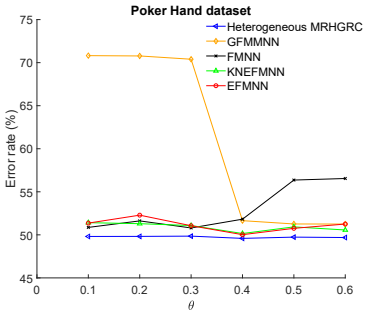
(b) Letter



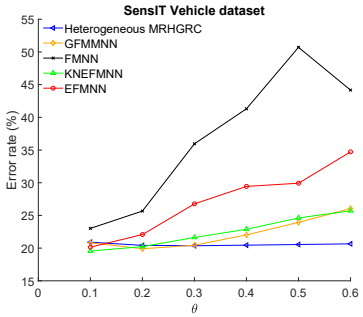
(c) MiniBooNE



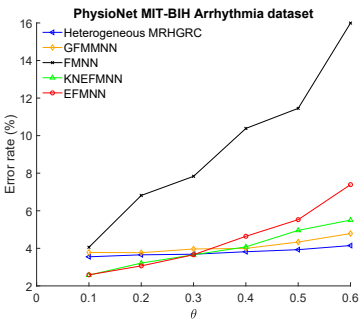
(d) MAGIC Gamma Telescope



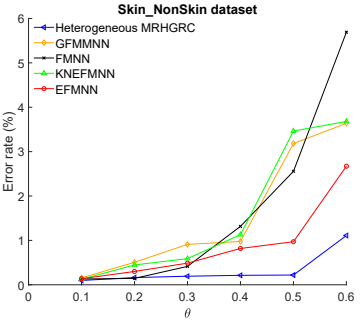
(e) Poker Hand



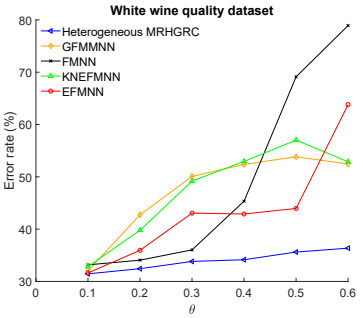
(f) SensIT Vehicle



(g) PhysioNet MIT-BIH Arrhythmia



(h) Skin_Nonskin



(i) White wine quality

Figure E.3 : The error rate of classifiers on several real testing datasets with the change in the data abstraction levels

Table E.2 : The lowest error rates and training time of classifiers on real datasets

Dataset	Algorithm	min E_V	min E_T	$E_T(\theta = \theta_V)$	θ_V	θ_T	Total training time (s)
covtype	He-MRHGRC	**5.312**	7.536	7.536	0.1	0.1	35,097.1298
	GFMM	6.426	8.690	8.690	0.1	0.1	171,511.1511
	FMNN	36.905	37.27	37.27	0.1	0.1	89,060.0745
	KNEFMNN	7.016	8.777	8.777	0.1	0.1	469,703.8682
	EFMNN	7.736	9.514	9.514	0.1	0.1	1,134,505.2210
	GNB	90.986	90.992	-	-	-	8.3349
	SVM	27.510	27.480	-	-	-	30,823.2404
	Decision tree	6.492	**6.479**	-	-	-	16.6013
Poker Hand	He-MRHGRC	**39.112**	49.589	49.813	0.1	0.4	2,170.2065
	GFMM	47.078	51.251	51.647	0.4	0.6	6,849.3993
	FMNN	42.586	50.801	50.865	0.1	0.3	9,159.2247
	KNEFMNN	40.286	50.153	51.412	0.1	0.4	9,953.7574
	EFMNN	40.292	50.02	51.363	0.1	0.4	9,831.7911
	GNB	49.880	49.879	-	-	-	0.1685
	SVM	46.838	**46.573**	-	-	-	32.2543
	Decision tree	51.866	52.162	-	-	-	0.2762
Skin_NonSkin	He-MRHGRC	**0.070**	0.097	0.097	0.1	0.1	11.9259
	GFMM	0.128	0.156	0.156	0.1	0.1	116.1890
	FMNN	0.12	0.13	0.144	0.2	0.1	1,709.5918
	KNEFMNN	0.107	0.124	0.124	0.1	0.1	163.9299
	EFMNN	0.12	0.136	0.136	0.1	0.1	846.6562
	GNB	7.6876	7.467	-	-	-	1.1434
	SVM	1.1047	1.1288	-	-	-	79.7543
	Decision tree	0.071	**0.070**	-	-	-	1.3032
SensIT Vehicle	He-MRHGRC	**14.921**	20.364	20.907	0.1	0.3	15,658.6847
	GFMM	14.961	19.903	20.816	0.1	0.2	35,675.7565
	FMNN	22.381	23.013	23.013	0.1	0.1	76,412.8685
	KNEFMNN	17.428	**19.557**	19.557	0.1	0.1	92,118.9432
	EFMNN	18.321	20.146	20.146	0.1	0.1	158,362.4286
	GNB	24.208	24.911	-	-	-	2.8469
	SVM	20.270	20.258	-	-	-	846.4761
	Decision tree	23.603	23.869	-	-	-	19.5091
MiniBooNE	He-MRHGRC	13.590	13.904	13.904	0.1	0.1	24.461218
	GFMM	16.999	17.622	17.622	0.1	0.1	370.7756
	FMNN	27.669	27.598	27.598	0.1	0.1	63.8761
	KNEFMNN	17.046	17.316	17.316	0.1	0.1	8,073.123
	EFMNN	17.287	17.504	17.504	0.1	0.1	24,572.8257
	GNB	71.469	71.656	-	-	-	1.8561
	SVM	17.551	17.358	-	-	-	1493.6944
	Decision tree	**10.549**	**10.821**	-	-	-	9.0240
Letter	He-MRHGRC	**3.336**	**5.160**	5.160	0.1	0.1	69.033658
	GFMM	8.958	11.320	11.320	0.1	0.1	888.5603
	FMNN	8.821	10.400	10.400	0.1	0.1	1,835.5864
	KNEFMNN	4.890	6.600	6.600	0.1	0.1	1,312.8839

Letter	EFMNN	4.616	5.920	5.920	0.1	0.1	2,684.6123
	GNB	35.421	35.160	-	-	-	0.1418
	SVM	26.463	25.640	-	-	-	9.5060
	Decision tree	11.731	11.080	-	-	-	0.2699
MAGIC Gamma Telescope	He-MRHGRC	**13.429**	17.807	18.312	0.1	0.5	21.1161
	GFMM	13.569	18.536	18.564	0.1	0.2	242.1811
	FMNN	17.550	18.480	18.480	0.1	0.1	2,052.5925
	KNEFMNN	14.382	**17.723**	17.723	0.1	0.1	1,099.5460
	EFMNN	16.428	19.434	19.434	0.1	0.1	2,261.7551
	GNB	27.082	27.734	-	-	-	0.1333
	SVM	17.886	18.116	-	-	-	4.0418
	Decision tree	18.466	18.146	-	-	-	0.3372
SUSY	He-MRHGRC	**25.601**	**26.148**	26.148	0.4	0.4	26,356.3185
	GFMM	26.430	26.965	26.965	0.4	0.4	254,181.1993
	FMNN	N/A	N/A	N/A	N/A	N/A	N/A
	KNEFMNN	N/A	N/A	N/A	N/A	N/A	N/A
	EFMNN	N/A	N/A	N/A	N/A	N/A	N/A
	GNB	26.433	26.533	-	-	-	41.9785
	SVM	N/A	N/A	-	-	-	N/A
	Decision tree	28.427	28.415	-	-	-	381.5020
PhysioNet MIT-BIH Arrhythmia	He-MRHGRC	3.076	3.549	3.549	0.1	0.1	53,100.2895
	GFMM	3.373	3.768	3.782	0.1	0.2	85,499.5178
	FMNN	3.982	4.056	4.056	0.1	0.1	100,232.7024
	KNEFMNN	2.33	**2.572**	2.572	0.1	0.1	164,408.8073
	EFMNN	**2.307**	2.590	2.590	0.1	0.1	186,775.4619
	GNB	87.1469	86.8491	-	-	-	4.9943
	SVM	8.170	7.820	-	-	-	601.4092
	Decision tree	5.358	4.892	-	-	-	37.7381
Default of credit card clients	He-MRHGRC	**14.827**	**19.769**	19.769	0.1	0.1	96.6555
	GFMM	17.511	21.724	23.804	0.1	0.4	550.3491
	FMNN	20.018	21.351	21.351	0.1	0.1	1,691.3673
	KNEFMNN	15.502	20.32	20.32	0.1	0.1	2,743.4832
	EFMNN	15.200	20.587	20.587	0.1	0.1	4,349.6698
	GNB	34.524	34.222	-	-	-	0.1748
	SVM	21.582	21.636	-	-	-	13.0480
	Decision tree	26.619	26.892	-	-	-	0.4684
White wine quality	He-MRHGRC	**24.898**	**31.454**	31.454	0.1	0.1	30.9282
	GFMM	27.265	32.026	32.026	0.1	0.1	33.1445
	FMNN	29.878	33.170	33.170	0.1	0.1	45.0146
	KNEFMNN	27.592	32.843	32.843	0.1	0.1	72.1374
	EFMNN	27.592	31.618	31.618	0.1	0.1	100.6138
	GNB	54.694	55.392	-	-	-	0.0151
	SVM	49.143	49.020	-	-	-	0.3169
	Decision tree	31.918	35.594	-	-	-	0.0309

Appendix F

Additional Results for Chapter 8

F.1 Proof of Lemma 8.1 in Chapter 8

This section provides the readers with the proof of Lemma 8.1 in Chapter 8.

Proof. Supposing that $\Phi = (\Phi_1, \ldots, \Phi_M)$ is a set of M random variables with given covariances $\sigma_{ij} = \texttt{Cov}(\Phi_i, \Phi_j)$, it is required to find variance of an average variable $\mathcal{L}(\Phi_1, \ldots, \Phi_M)$ obtained as a linear combination of M random variables, i.e.,

$$\mathcal{L}(\Phi_1, \ldots, \Phi_M) = \sum_{i=1}^{M} (\lambda_i \cdot \Phi_i)$$

this formula can be rewritten in a compact way using matrix and vector notations as follows:

$$\mathcal{L}(\Phi) = \mathbf{\Lambda}^T \cdot \Phi$$

where $\mathbf{\Lambda}^T = (\lambda_1, \ldots, \lambda_M)$. And then, one has the expected value:

$$\mathbb{E}(\mathcal{L}(\Phi)) = \mathbb{E}(\mathbf{\Lambda}^T \cdot \Phi) = \mathbf{\Lambda}^T \cdot \mathbb{E}(\Phi)$$

and the variance:

$$\begin{aligned}
\texttt{Var}(\mathcal{L}(\Phi)) &= \mathbb{E}(\mathcal{L}^2(\Phi)) - [\mathbb{E}(\mathcal{L}(\Phi))]^2 = \mathbb{E}(\mathbf{\Lambda}^T \Phi \Phi^T \mathbf{\Lambda}) - \mathbb{E}(\mathbf{\Lambda}^T \Phi)[\mathbb{E}(\mathbf{\Lambda}^T \Phi)]^T \\
&= \mathbf{\Lambda}^T \mathbb{E}(\Phi \Phi^T) \mathbf{\Lambda} - \mathbf{\Lambda}^T \mathbb{E}(\Phi)(\mathbb{E}(\Phi))^T \mathbf{\Lambda} = \mathbf{\Lambda}^T [\mathbb{E}(\Phi \Phi^T) - \mathbb{E}(\Phi)(\mathbb{E}(\Phi))^T] \mathbf{\Lambda} \\
&= \mathbf{\Lambda}^T \texttt{Cov}(\Phi) \mathbf{\Lambda} = \mathbf{\Lambda}^T \Sigma \mathbf{\Lambda}
\end{aligned}$$

where $\Sigma = (\sigma_{ij})$ is the covariance of Φ

In this lemma, $\sigma_{ij} = \rho \cdot \sigma^2$ when $i \neq j$. One also has $\sigma_{ii} = \texttt{Cov}(\Phi_i, \Phi_i) = \sigma^2 = [\rho + (1 - \rho)]\sigma^2$. Hence, the co-variance matrix Σ may be decomposed into the sum of two matrices, i.e., one includes ρ in every entry and the other includes $(1 - \rho)$ on

the main diagonal and zeros for the rest. Formally, it can achieve:

$$\Sigma = \sigma^2[\rho \mathbf{1}_M \mathbf{1}_M^T + (1-\rho)\mathbf{I}_M]$$

where $\mathbf{1}_M$ is a column vector containing M 1's and $\mathbf{I}_M$ is an identity matrix with size $M \times M$. Then, we get:

$$\texttt{Var}(\mathcal{L}(\Phi)) = \mathbf{\Lambda}^T \sigma^2[\rho \mathbf{1}_M \mathbf{1}_M^T + (1-\rho)\mathbf{I}_M]\mathbf{\Lambda} = (\mathbf{\Lambda}^T \mathbf{1}_M \mathbf{1}_M^T \mathbf{\Lambda})\rho\sigma^2 + (\mathbf{\Lambda}^T \mathbf{I}_M \mathbf{\Lambda})(1-\rho)\sigma^2$$

For $\mathbf{\Lambda}^T = (1/M, \ldots, 1/M)$, we get:

$$\mathbf{\Lambda}^T \mathbf{1}_M \mathbf{1}_M^T \mathbf{\Lambda} = (\mathbf{\Lambda}^T \mathbf{1}_M)^2 = (M \cdot 1/M)^2 = 1$$

and

$$\mathbf{\Lambda}^T \mathbf{I}_M \mathbf{\Lambda} = 1/M^2 + \ldots + 1/M^2 = M \cdot 1/M^2 = 1/M$$

Therefore,

$$\texttt{Var}(\mathcal{L}(\Phi)) = \rho\sigma^2 + \frac{1-\rho}{M}\sigma^2$$

The lemma is proved. □

F.2 Proof of Lemma 8.2 in Chapter 8

This section provides the proof of Lemma 8.2 in Chapter 8.

Proof. The margin function of the random hyperboxes model with M base learners at each input sample X can be shown as follows:

$$\mathcal{M}(X, c) = \frac{1}{M}\sum_{i=1}^{M} \mathbb{1}(h_i(X) = c) - \max_{j \neq c} \frac{1}{M}\sum_{i=1}^{M} \mathbb{1}(h_i(X) = j)$$

For random vectors $\Phi_1, \Phi_2, \ldots$ and for all input vectors X, to prove Lemma 8.2, it suffices to show

$$\frac{1}{M}\sum_{i=1}^{M} \mathbb{1}(h_i(X) = j) \xrightarrow{M \to \infty} \mathbf{P}_\Phi(h(X, \Phi) = j)$$

where $h_i(X) \equiv h(X, \Phi_i)$, and $\mathbb{1}(\cdot)$ is an indicator function.

For each hyperbox-based learner, $h(X, \Phi_i) = j$ is union of hyerboxes with class j and their neighborhood regions which generate the maximum membership value from these hyperboxes to an input X in comparison to hyperboxes representing other classes. Assuming a finite number of random vectors Φ (the finite number of sample subsets and finite number of feature subsets) from which any hyperbox-based learner $h(X, \Phi_i)$ ($\Phi_i \subset \Phi$) is constructed, then there exists a finite number K of such unions of hyperboxes and neighbourhood regions, called $S_1, \ldots, S_K$.

Let define:

$$\varphi(\Phi) = k \text{ if } \{X : h(X, \Phi) = j\} = S_k$$

Let N_k be the number of times that $\varphi(\Phi_i) = k$ in the first M trials, then it can obtain:

$$\frac{1}{M}\sum_{i=1}^{M} \mathbb{1}(h(X, \Phi_i) = j) = \frac{1}{M}\sum_{k} N_k \mathbb{1}(X \in S_k)$$

According to the strong law of large numbers when M increases,

$$N_k = \frac{1}{M}\sum_{i=1}^{M} \mathbb{1}(\varphi(\Phi_i) = k)$$

converges almost surely (a.s.) with probability 1 to

$$\mathbb{E}_{\Phi}[\mathbb{1}(\varphi(\Phi) = k)] = \mathbf{P}_{\Phi}(\varphi(\Phi) = k)$$

Therefore,

$$\begin{aligned}\frac{1}{M}\sum_{i=1}^{M} \mathbb{1}(h(X, \Phi_i) = j) &\xrightarrow{a.s.} \sum_{k} \mathbf{P}_{\Phi}(\varphi(\Phi) = k)\mathbb{1}(X \in S_k) \\ &= \mathbf{P}_{\Phi}(h(X, \Phi) = j)\end{aligned}$$

The lemma is proved. □

F.3 Proof of Theorem 8.2 in Chapter 8

This section shows the proof for Theorem 8.2 in Chapter 8.

Proof. From lemma 8.2, we have:

$$\mathcal{M}^*(X,c) = \mathbf{P}_\Phi(h(X,\Phi)=c) - \max_{j\neq c}\mathbf{P}_\Phi(h(X,\Phi)=j)$$

With the assumption of the strength $\mathcal{S} = \mathbb{E}_{\mathbf{X},\mathcal{C}}\mathcal{M}^*(X,c) > 0$, according to Chebyshev's inequality, we have:

$$\begin{aligned}\mathcal{E}^* &= \mathbf{P}_{\mathbf{X},\mathcal{C}}\left[\mathcal{M}^*(X,c) < 0\right] \leq \mathbf{P}_{\mathbf{X},\mathcal{C}}\left[\mathcal{S} - \mathcal{M}^*(X,c) \geq \mathcal{S}\right] \\ &= \mathbf{P}_{\mathbf{X},\mathcal{C}}\left[|\mathcal{M}^*(X,c) - \mathcal{S}| \geq \mathcal{S}\right] \leq \frac{\mathtt{Var}_{\mathbf{X},\mathcal{C}}(\mathcal{M}^*(X,c))}{\mathcal{S}^2}\end{aligned}$$

For any function f and two i.i.d. random variables Φ and Φ', we have:

$$\mathbb{E}_\Phi[f(\Phi)]^2 = \mathbb{E}_{\Phi,\Phi'}[f(\Phi)f(\Phi')]$$

In Chapter 8, we get $\mathcal{M}^*(X,c) = \mathbb{E}_\Phi\mathcal{R}(\Phi)$, thus

$$[\mathcal{M}^*(X,c)]^2 = \mathbb{E}_\Phi\mathcal{R}(\Phi)^2 = \mathbb{E}_{\Phi,\Phi'}[\mathcal{R}(\Phi)\mathcal{R}(\Phi')]$$

Now, $\mathtt{Var}_{X,\mathcal{C}}(\mathcal{M}^*(X,c))$ can be computed as follows:

$$\begin{aligned}\mathtt{Var}_{\mathbf{X},\mathcal{C}}(\mathcal{M}^*(X,c)) &= \mathbb{E}_{\mathbf{X},\mathcal{C}}([\mathcal{M}^*(X,c)]^2) - \Big[\mathbb{E}_{\mathbf{X},\mathcal{C}}(\mathcal{M}^*(X,c))\Big]^2 \\ &= \mathbb{E}_{\mathbf{X},\mathcal{C}}\Big[\mathbb{E}_{\Phi,\Phi'}[\mathcal{R}(\Phi)\mathcal{R}(\Phi')]\Big] - \Big[\mathbb{E}_{\mathbf{X},\mathcal{C}}(\mathbb{E}_\Phi\mathcal{R}(\Phi))\Big]^2 \\ &= \mathbb{E}_{\Phi,\Phi'}\Big[\mathbb{E}_{\mathbf{X},\mathcal{C}}[\mathcal{R}(\Phi)\mathcal{R}(\Phi')]\Big] - \Big[\mathbb{E}_\Phi(\mathbb{E}_{\mathbf{X},\mathcal{C}}\mathcal{R}(\Phi))\Big]^2 \\ &= \mathbb{E}_{\Phi,\Phi'}\Big[\mathbb{E}_{\mathbf{X},\mathcal{C}}[\mathcal{R}(\Phi)\mathcal{R}(\Phi')]\Big] - \mathbb{E}_{\Phi,\Phi'}\Big[\mathbb{E}_{\mathbf{X},\mathcal{C}}\mathcal{R}(\Phi)\mathbb{E}_{\mathbf{X},\mathcal{C}}\mathcal{R}(\Phi')\Big] \\ &= \mathbb{E}_{\Phi,\Phi'}\Big[\mathbb{E}_{\mathbf{X},\mathcal{C}}[\mathcal{R}(\Phi)\mathcal{R}(\Phi')] - \mathbb{E}_{\mathbf{X},\mathcal{C}}\mathcal{R}(\Phi)\mathbb{E}_{\mathbf{X},\mathcal{C}}\mathcal{R}(\Phi')\Big] \\ &= \mathbb{E}_{\Phi,\Phi'}\Big[\mathtt{Cov}_{\mathbf{X},\mathcal{C}}(\mathcal{R}(\Phi)\mathcal{R}(\Phi'))\Big] \\ &= \mathbb{E}_{\Phi,\Phi'}\Big[\rho_{\mathbf{X},\mathcal{C}}(\Phi,\Phi')\sigma_{\mathbf{X},\mathcal{C}}(\mathcal{R}(\Phi))\sigma_{\mathbf{X},\mathcal{C}}(\mathcal{R}(\Phi'))\Big] \\ &= \overline{\rho}\Big[\mathbb{E}_\Phi(\sigma_{\mathbf{X},\mathcal{C}}(\mathcal{R}(\Phi)))\Big]^2\end{aligned}$$

where $\overline{\rho} = \mathbb{E}_{\Phi,\Phi'}[\rho_{\mathbf{X},\mathcal{C}}(\Phi,\Phi')]$

For any random variable $\mathbf{Z}$, $\mathtt{Var}(\mathbf{Z}) \geq 0 \Rightarrow \mathbb{E}(\mathbf{Z}^2) - \mathbb{E}(\mathbf{Z})^2 \geq 0 \Rightarrow \mathbb{E}(\mathbf{Z})^2 \leq \mathbb{E}(\mathbf{Z}^2)$. Therefore,

$$\mathtt{Var}_{\mathbf{X},\mathcal{C}}(\mathcal{M}^*(X,c)) = \overline{\rho}\Big[\mathbb{E}_\Phi(\sigma_{\mathbf{X},\mathcal{C}}(\mathcal{R}(\Phi)))\Big]^2 \leq \overline{\rho}\,\mathbb{E}_\Phi(\sigma_{\mathbf{X},\mathcal{C}}(\mathcal{R}(\Phi))^2) = \overline{\rho}\,\mathbb{E}_\Phi(\mathtt{Var}_{\mathbf{X},\mathcal{C}}(\mathcal{R}(\Phi)))$$

In addition, using the definition of the variance for a random variable and inequality $\mathbb{E}(\mathbf{Z})^2 \leq \mathbb{E}(\mathbf{Z}^2)$, we can write:

$$\begin{aligned}
\mathbb{E}_{\Phi}(\mathtt{Var}_{\mathbf{X},\mathcal{C}}(\mathcal{R}(\Phi))) &= \mathbb{E}_{\Phi}\Big[\mathbb{E}_{\mathbf{X},\mathcal{C}}[\mathcal{R}(\Phi)^2] - \mathbb{E}_{\mathbf{X},\mathcal{C}}[\mathcal{R}(\Phi)]^2\Big] \\
&= \mathbb{E}_{\Phi}\Big[\mathbb{E}_{\mathbf{X},\mathcal{C}}[\mathcal{R}(\Phi)^2]\Big] - \mathbb{E}_{\Phi}\Big[[\mathbb{E}_{\mathbf{X},\mathcal{C}}\mathcal{R}(\Phi)]^2\Big] \\
&\leq \mathbb{E}_{\Phi}\Big[\mathbb{E}_{\mathbf{X},\mathcal{C}}[\mathcal{R}(\Phi)^2]\Big] - \Big[\mathbb{E}_{\Phi}(\mathbb{E}_{\mathbf{X},\mathcal{C}}[\mathcal{R}(\Phi)])\Big]^2 \\
&= \mathbb{E}_{\Phi}\Big[\mathbb{E}_{\mathbf{X},\mathcal{C}}[\mathcal{R}(\Phi)^2]\Big] - \Big[\mathbb{E}_{\mathbf{X},\mathcal{C}}(\mathbb{E}_{\Phi}[\mathcal{R}(\Phi)])\Big]^2 \\
&= \mathbb{E}_{\Phi}\Big[\mathbb{E}_{\mathbf{X},\mathcal{C}}[\mathcal{R}(\Phi)^2]\Big] - \Big[\mathbb{E}_{\mathbf{X},\mathcal{C}}\mathcal{M}^*(X,c)\Big]^2 \\
&\leq 1 - \mathcal{S}^2
\end{aligned}$$

due to $\mathcal{R}(\Phi) \leq 1$ and $\mathcal{S} = \mathbb{E}_{X,\mathcal{C}}\mathcal{M}^*(X,c)$. As a result,

$$\mathcal{E}^* \leq \frac{\mathtt{Var}_{\mathbf{X},\mathcal{C}}(\mathcal{M}^*(X,c))}{\mathcal{S}^2} \leq \frac{\overline{\rho}\ \mathbb{E}_{\Phi}(\mathtt{Var}_{\mathbf{X},\mathcal{C}}(\mathcal{R}(\Phi)))}{\mathcal{S}^2} \leq \frac{\overline{\rho}\ (1-\mathcal{S}^2)}{\mathcal{S}^2} = \overline{\rho}\ \Big(\frac{1}{\mathcal{S}^2} - 1\Big)$$

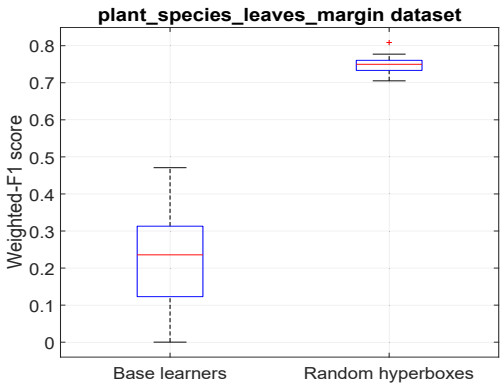
The theorem is proved. □

F.4 Additional Experimental Results

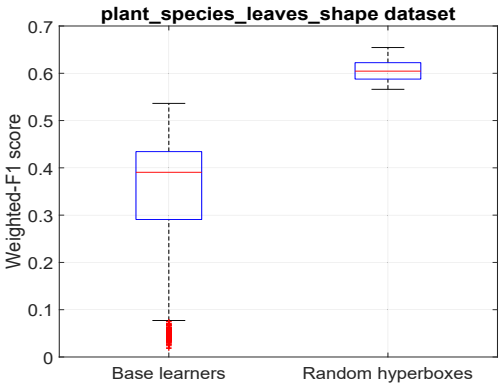
F.4.1 Supplementary Part for Analyzing the Variance of the Random Hyperboxes Classifier

This part provides some supplementary figures for subsection 8.3.1 in Chapter 8. This experiment was performed on six datasets with diversity in the numbers of samples, features, and classes, i.e., *plant species leaves margin*, *plant species leaves shape*, *heart*, *vowel*, *ringnorm*, and *connectionist bench sonar*. Figure F.1 shows the variance values in terms of weighted-F1 scores using the 10 times repeated 4-fold cross-validation of base classifiers and the random hyperboxes models over different datasets. These results confirm that the random hyperboxes model is able to reduce the variance in its base learners, and so it can achieve better performance than its base models.

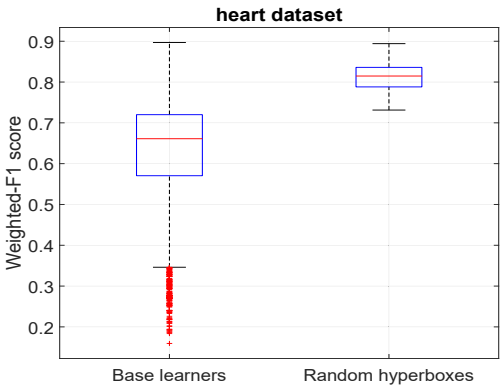
Figure F.2 shows the probability of the number of features, p, used to build the 4000 base learners for the experiment shown in subsection 8.3.1 in Chapter 8. It



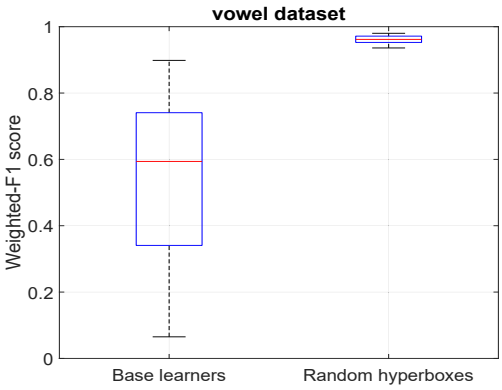
(a) Plant species leaves margin



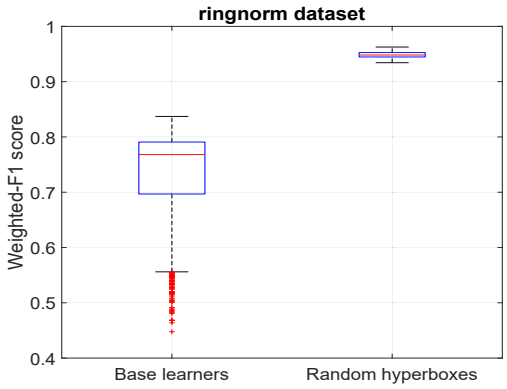
(b) Plant species leaves shape



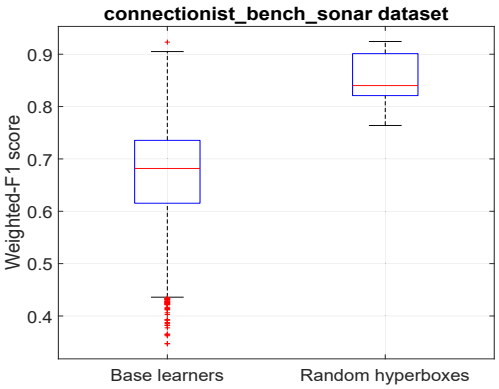
(c) Heart



(d) Vowel



(e) Ringnorm

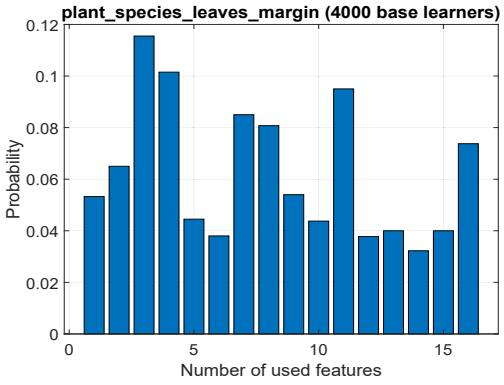


(f) Connectionist bench sonar

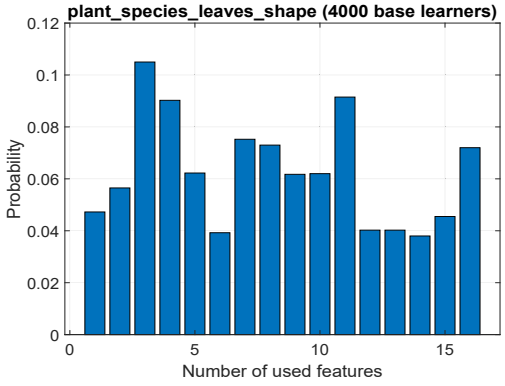
Figure F.1 : The variances of the random hyperboxes models and their base learners for different datasets.

can be observed that the probability distribution of the number of used features is nearly uniform in all 4000 base learners.

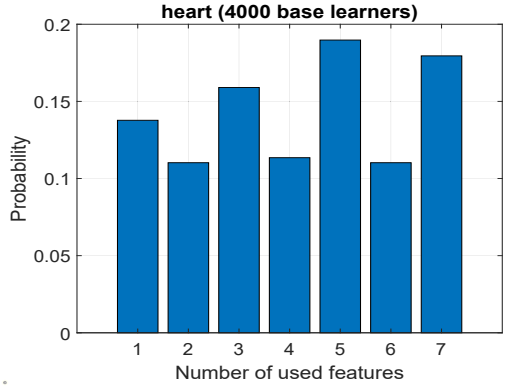
The used probability of each feature over 4000 base learners can also be identified



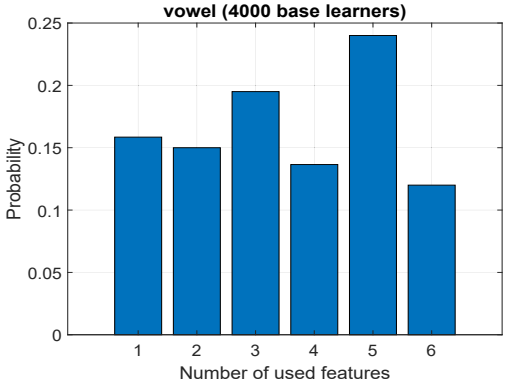
(a) Plant species leaves margin



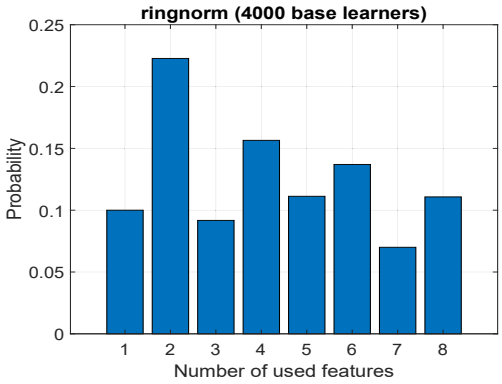
(b) Plant species leaves shape



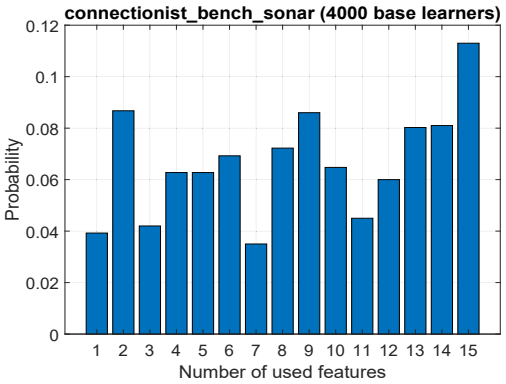
(c) Heart



(d) Vowel



(e) Ringnorm



(f) Connectionist bench sonar

Figure F.2 : The probability of the number of used features for all base learners over different datasets.

to find the importance scores of features with respect to the performance of the ensemble model. This information is given in Figure F.3. From the importance scores of features, a single model was built using top-K of the most important

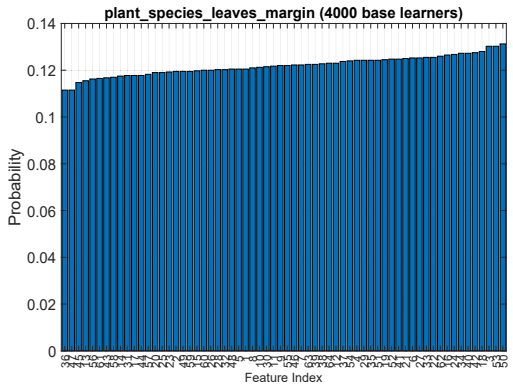
features to assess the performance of the random hyperboxes and the use of single models. It can be observed that in many datasets, the single model often achieves better performance when it is trained on more features. However, in several cases such as in *ringnorm* and *connectionist bench sonar* datasets, the best performance of the single model is obtained if it is trained on a subset of the most important features. From Figures F.1 and F.4, it is easily seen that the random hyperboxes model trained using a subset of features usually achieves higher classification accuracy than the single model trained on the same dataset using all of the available features.

F.4.2 Analyzing the Effectiveness of the Random Hyperboxes on High Dimensional Data

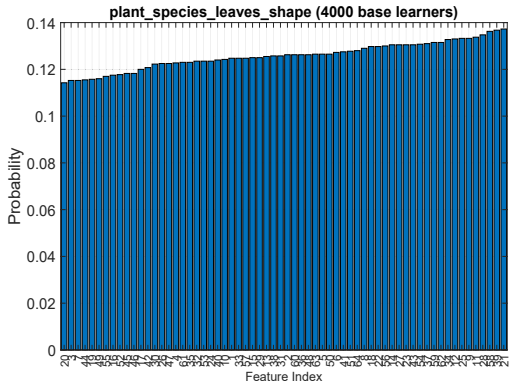
When building predictive models for problems with very high dimensional data, the performance of models is negatively influenced by the redundancy of features. This problem is known as the Curse of Dimensionality (Friedman 1997). This experiment is to assess the robustness of the random hyperboxes classifier for high dimensional data in comparison to the single IOL-GFMM model. Two very high dimensional datasets were used, i.e., *PEMS database* (Cuturi 2011) and *Complex Hydraulic System* (Helwig et al. 2015). 80% of samples in each dataset were used as training data and the remaining 20% of samples were testing data. The summaries of these datasets were shown in Table 6.1 in Chapter 6.

In this experiment, each base learner in the random hyperboxes model is trained on 50% of samples randomly selected from the training data. The maximum number of used features for each base learner is set to $2\sqrt{n}$, where n is the number of dimensions of the dataset. The number of base learners for each random hyperboxes model is $M = 100$. The weighted-F1 scores of the random hyperboxes and single IOL-GFMM model through different values of θ are given in Figure F.5 for the *PEMS database* dataset and in Figure F.6 for the *Complex Hydraulic System* dataset.

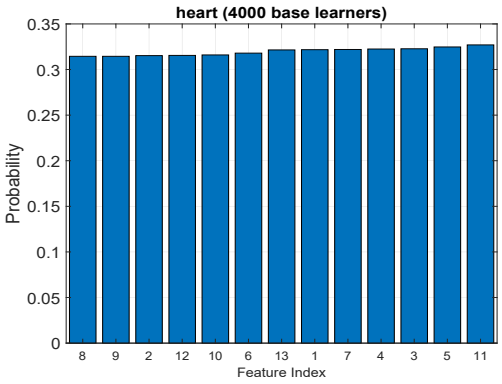
It can be observed that the IOL-GFMM has consistently lower performance than RH with the very high dimensional data. In contrast, the random hyperboxes can achieve high accuracy using only $2\sqrt{n}$ random features at most for each base learner.



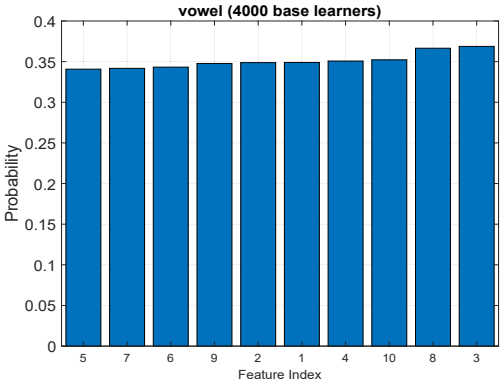
(a) Plant species leaves margin



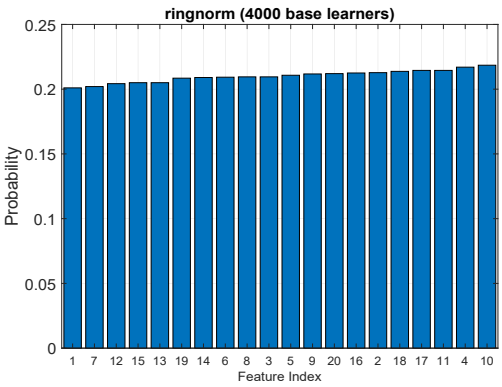
(b) Plant species leaves shape



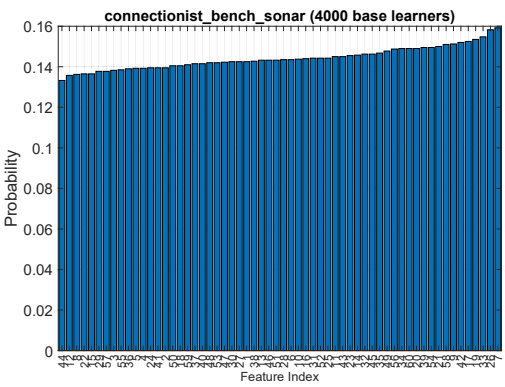
(c) Heart



(d) Vowel



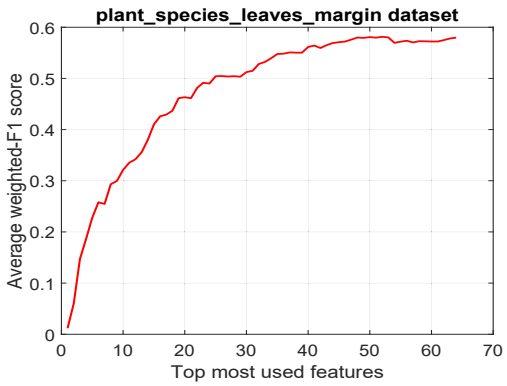
(e) Ringnorm



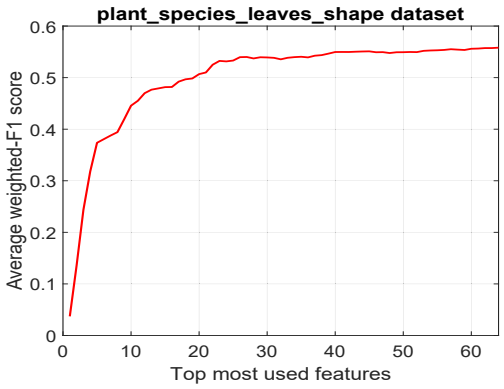
(f) Connectionist bench sonar

Figure F.3 : The probability of each feature used for all base learners over different datasets.

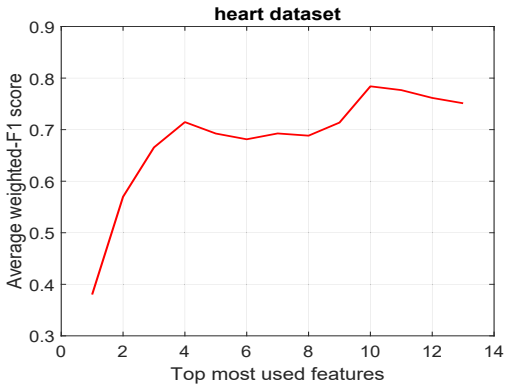
The diversity in the base learners and the use of a low number of features allow the random hyperboxes to obtain better performance across the maximum hyperbox size values. Because each base learner in the random hyperboxes model uses a much



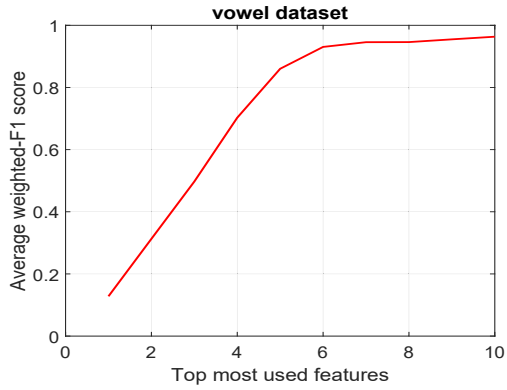
(a) Plant species leaves margin



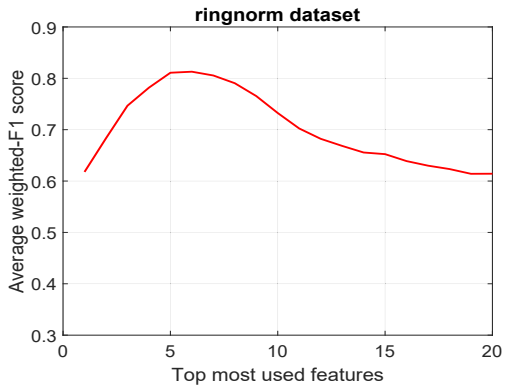
(b) Plant species leaves shape



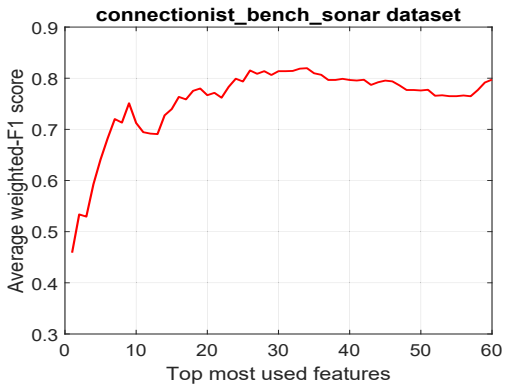
(c) Heart



(d) Vowel



(e) Ringnorm



(f) Connectionist bench sonar

Figure F.4 : Average weighted-F1 scores over 40 testing folds of a single model using training sets with top-k most used features over different datasets.

smaller number of features compared to the IOL-GFMM model trained using all features, training time and testing time of the random hyperboxes is faster than that of the IOL-GFMM model. The training and testing time of each classifier

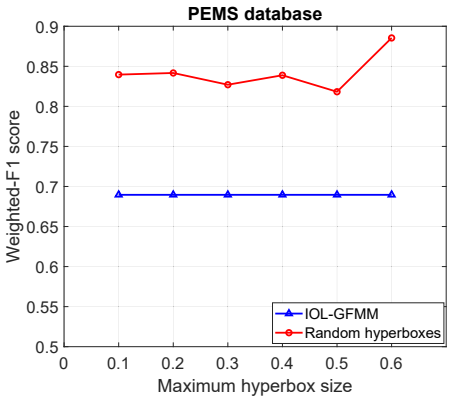


Figure F.5 : Weighted-F1 score of the random hyperboxes and IOL-GFMM for the *PEMS database* dataset

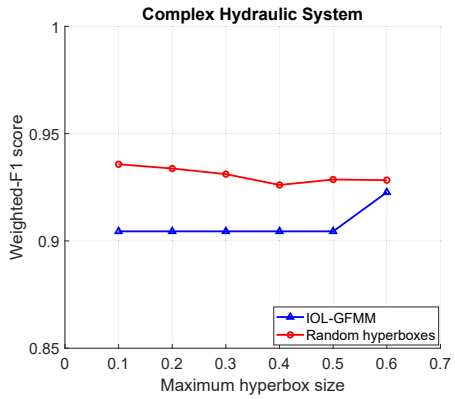


Figure F.6 : Weighted-F1 score of the random hyperboxes and IOL-GFMM for the *Complex Hydraulic System* dataset

is given in Tables F.1 and F.2. Fast training and testing time along with better accuracy confirm the efficiency of the ensemble model in comparison to the single model using the same learning algorithm.

F.4.3 Supplementary Part for Analyzing the Roles of the Number of Base Learners, Maximum Number of Used Features, and Maximum hyperbox size in the Random Hyperboxes models

This experiment was performed on eight different datasets with diversity in the numbers of samples, features, and classes, i.e., *plant species leaves margin*, *plant*

Table F.1 : Training time (s) of the IOL-GFMM and random hyperboxes model on the high dimensional datasets

Dataset	**Algorithm**	$\theta = 0.1$	$\theta = 0.2$	$\theta = 0.3$	$\theta = 0.4$	$\theta = 0.5$	$\theta = 0.6$
PEMS database	IOL-GFMM	51.3784	56.5849	52.6432	52.12905	56.7359	57.1392
	RH	26.2364	26.4292	27.0474	27.3853	29.3139	28.7593
Complex Hydraulic	IOL-GFMM	2093.5169	2235.3104	2045.8519	1914.7439	1987.5575	1785.5609
System	RH	154.9104	125.8966	100.0234	84.0987	75.5298	66.7039

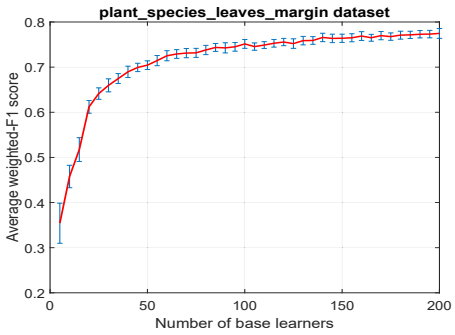
Table F.2 : Testing time (s) of the IOL-GFMM and random hyperboxes model on the high dimensional datasets

Dataset	**Algorithm**	$\theta = 0.1$	$\theta = 0.2$	$\theta = 0.3$	$\theta = 0.4$	$\theta = 0.5$	$\theta = 0.6$
PEMS database	IOL-GFMM	121.2674	126.1965	121.4517	122.0169	126.4106	126.3136
	RH	11.3272	11.3308	12.8158	11.6774	12.3205	10.8228
Complex Hydraulic	IOL-GFMM	1440.4623	1506.2449	1467.3662	1357.6034	1277.8380	1083.6029
System	RH	118.4271	69.8559	44.8562	29.9445	23.3218	17.1749

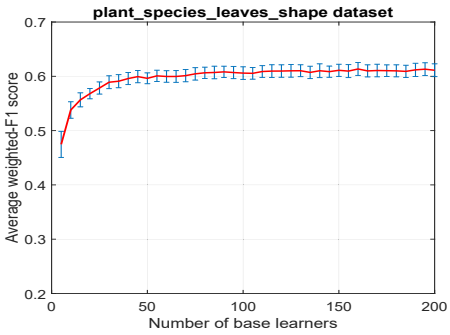
species leaves shape, *movement libras*, *connectionist bench sonar*, *vehicle sihouettes*, *breast cancer wisconsin*, *heart*, and *vowel*. The purpose of this experiment is to study the impacts of the number of base learners, the maximum number of used features, and the maximum hyperbox size threshold on the classification performance of the random hyperboxes model.

Figure F.7 shows the change in the average weighted-F1 score when increasing the number of base estimators. It can be observed a general trend over all experimental datasets which is that the increase in the number of base learners does not lead to the decrease in the classification accuracy. These empirical results are consistent with the statements in the theoretical part (subsection 8.2.3 in Chapter 8).

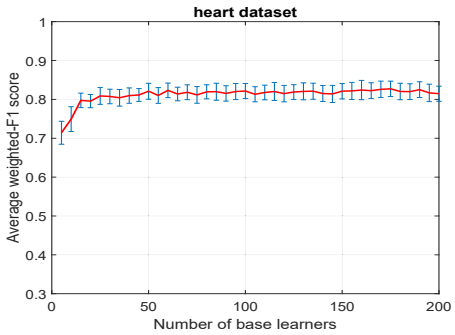
Figure F.8 presents the change in the classification performance when the maximum number of used features increases. A general trend can be observed in which the classification accuracy only increases up to a certain value of the maximum number of used features, and then decreases if the maximum number of features available for the base classifiers is increased. The reason for this trend is explained by the correlation between base learners as shown in subsection 8.3.1 in Chapter 8.



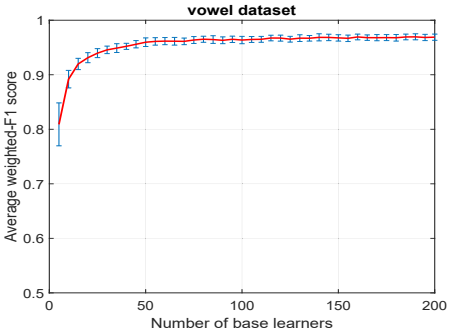
(a) Plant species leaves margin



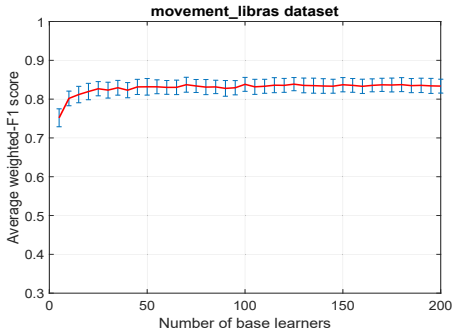
(b) Plant species leaves shape



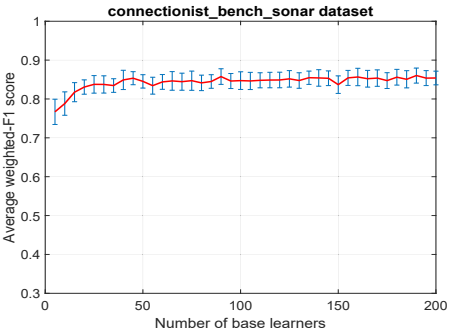
(c) Heart



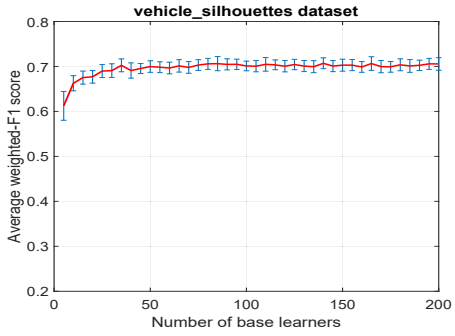
(d) Vowel



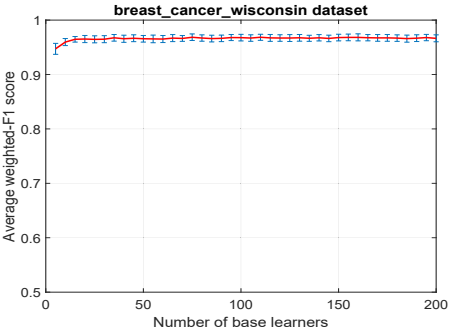
(e) Movement libras



(f) Connectionist bench sonar



(g) Vehicle silhouettes



(h) Breast cancer wisconsin

Figure F.7 : The change in the average weighted-F1 scores when increasing the number of base learners for different datasets.

Figure F.9 describes the change in the classification performance when the maximum hyperbox size threshold increases. It can be seen that the performance of six out of eight datasets slightly decreases when increasing the values of θ. The remaining two datasets slightly increases the classification accuracy when the value of θ goes up. However, in general, the change of the average weighted-F1 score between the different values of θ is small (lower than 5%). These outcomes indicated that the random hyperboxes models are less impacted by the change in the values of the maximum hyperbox size parameter.

F.4.4 Datasets and Parameter Settings

In Chapter 8, 20 datasets with diversity in the numbers of samples, features, and classes taken from the UCI repository (Dua and Graff 2019) were used. Table F.3 summarizes the information of these datasets. Each dataset is normalized to the range of [0, 1] according to the requirement of the fuzzy min-max neural networks. The experiments were executed on the computer using Red Hat Enterprise Linux 7.5 with Intel Xeon Gold 6150 2.7GHz CPU and 64GB RAM.

For experiments, the maximum hyperbox size of based learners in the random hyperboxes model, as well as different types of FMNNs, is set to $\theta = 0.1$ and the sensitivity parameter of the membership function is fixed at $\gamma = 1$. In terms of default settings for the RH model without hyperparameter tuning, this study deployed the threshold $2\sqrt{n}$ for the maximum number of used features and 50% of training samples were randomly sampled to train base learners ($r_s = 0.5$), the number of base learners $M = 100$ was also set.

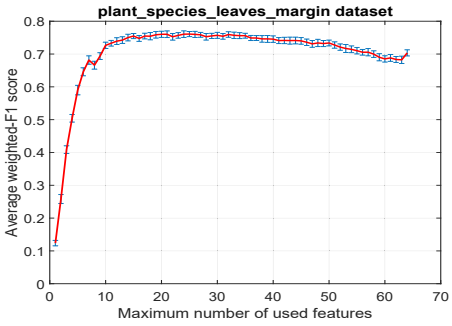
To compare the performance of the RH model to other ensemble models, the grid-search method along with 3-fold cross-validation was employed for each training fold to discover the best setting combination. After that, each ensemble model was trained on the full training set using the best hyperparameters. Three hyperparameters were tuned for ensemble models, i.e., number of base learners ($M \in \{30, 50, 70, 100, 150, 200\}$), the maximum number of used features ($m_f \in \{0.2, 0.3, 0.4, 0.5, 0.6\} \cdot n$), and sampling rate for samples ($r_s \in \{0.3, 0.5, 0.7\}$). The other para-

Table F.3 : The descriptions of the used datasets in Chapter 8

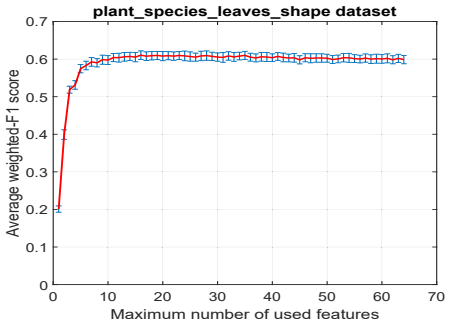
ID	Dataset	# samples	# features	# classes
1	Balance scale	625	4	3
2	banknote authentication	1372	4	2
3	blood transfusion	748	4	2
4	breast cancer wisconsin	699	9	2
5	BreastCancerCoimbra	116	9	2
6	connectionist bench sonar	208	60	2
7	haberman	306	3	2
8	heart	270	13	2
9	movement libras	360	90	15
10	pima diabetes	768	8	2
11	plant species leaves margin	1600	64	100
12	plant species leaves shape	1600	64	100
13	ringnorm	7400	20	2
14	landsat satellite	6435	36	6
15	twonorm	7400	20	2
16	vehicle silhouettes	846	18	4
17	vertebral column	310	6	3
18	vowel	990	10	11
19	waveform	5000	21	3
20	wireless indoor localization	2000	7	4

maters got default settings of libraries such as scikit-learn (Pedregosa et al. 2011b), XGBoost (Chen and Guestrin 2016), LightGBM (Ke et al. 2017) apart from the maximum tree depth of decision trees and tree-based ensemble methods is set to the value of 8 to prevent overfitting (Bertsimas and Dunn 2017; Chen and Guestrin 2016).

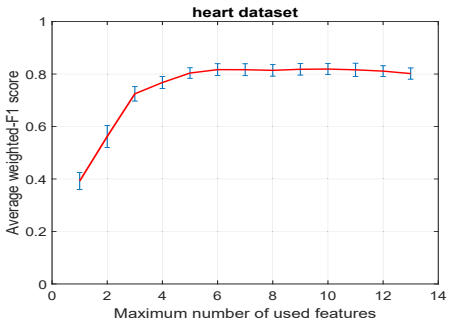
For support vector machines, the Radial Basis function was used as a kernel and two hyperparameters, i.e., the penalty parameter ($C \in \{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$) and the gamma parameter ($\gamma \in \{2^{-15}, 2^{-13}, \ldots, 2^{3}\}$) were adjusted as shown in (Hsu et al. 2003). For K-nearest neighbours model, the searching range of K was set in the range of $\{1, 3, \ldots, 15\}$. In terms of decision trees, the minimum number of samples in each leaf was searched in the range of $\{1, \ldots, 50\}$.



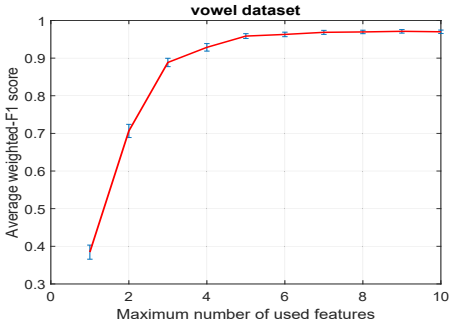
(a) Plant species leaves margin



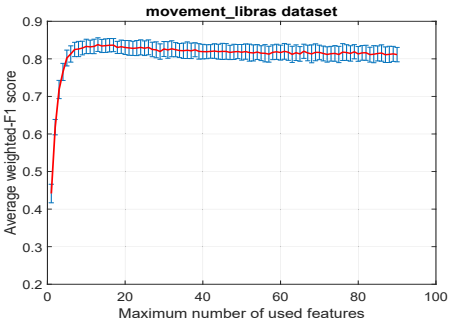
(b) Plant species leaves shape



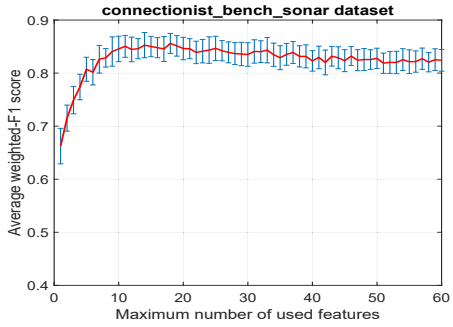
(c) Heart



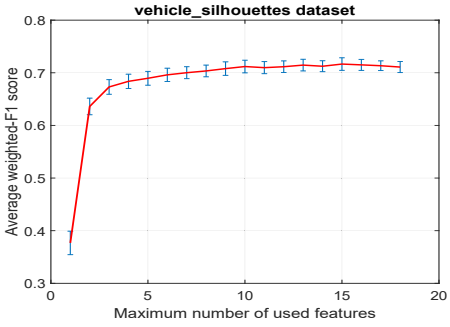
(d) Vowel



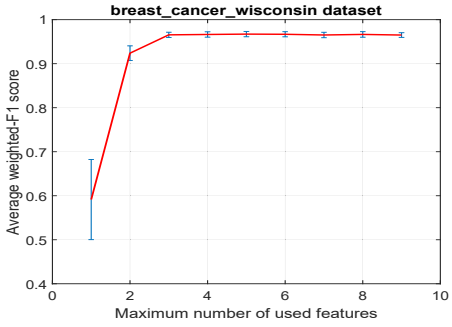
(e) Movement libras



(f) Connectionist bench sonar



(g) Vehicle silhouettes



(h) Breast cancer wisconsin

Figure F.8 : The change in the average weighted-F1 scores when increasing the maximum number of used dimensions for different datasets.

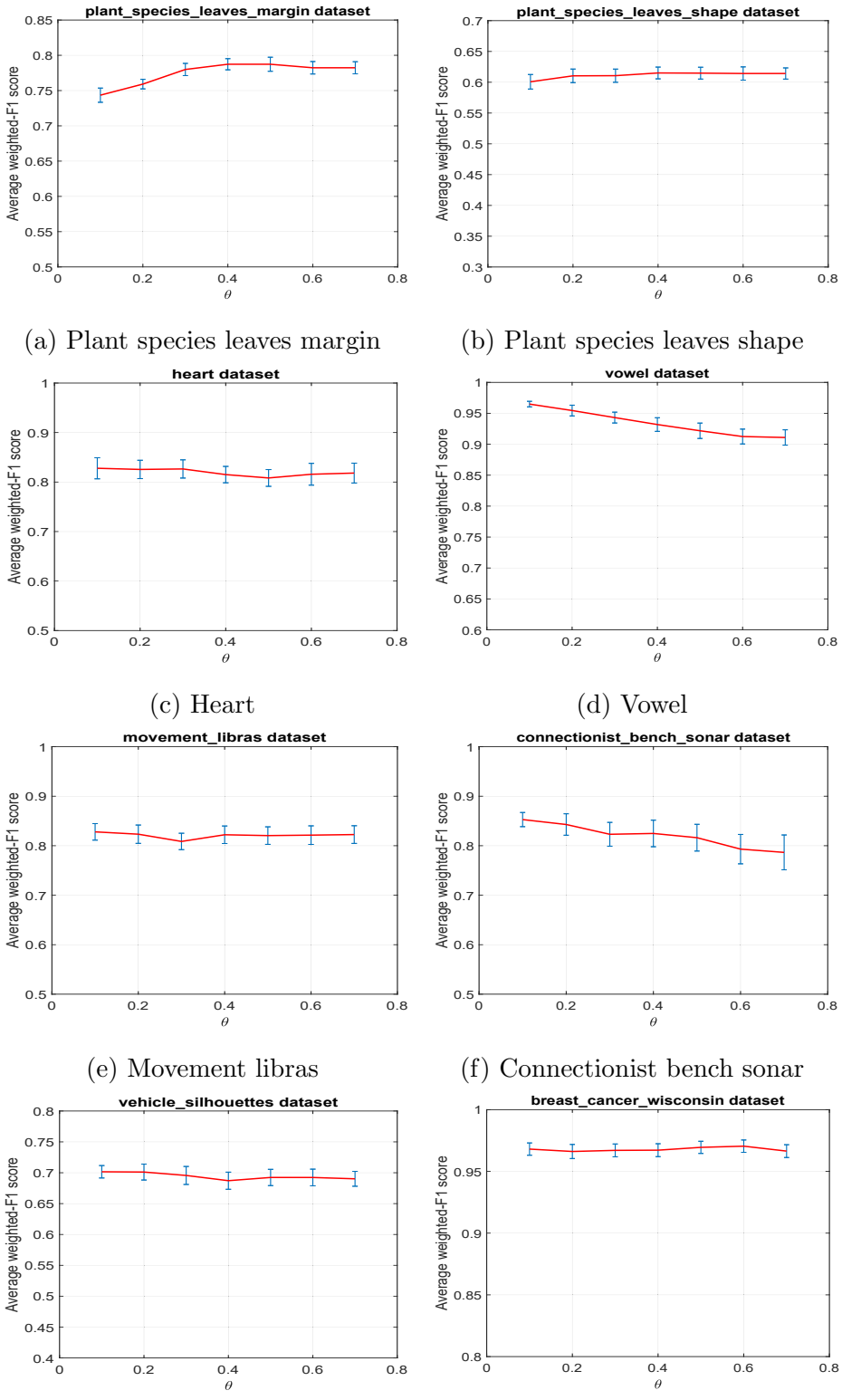


Figure F.9 : The change in the average weighted-F1 scores when increasing the maximum hyperbox size threshold for different datasets.

Bibliography

Abe, S., 2001, 'Dynamic fuzzy rule generation', *Pattern Classification: Neuro-fuzzy Methods and Their Comparison*, Springer London, pp. 177–196.

Abe, S. & Ming-Shong, L., 1995, 'A method for fuzzy rules extraction directly from numerical data and its application to pattern classification', *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 1, pp. 18–28.

Acharya, U. R., Oh, S. L., Hagiwara, Y., Tan, J. H., Adam, M., Gertych, A. & Tan, R. S., 2017, 'A deep convolutional neural network model to classify heartbeats', *Computers in Biology and Medicine*, vol. 89, pp. 389 – 396.

Ahmedt-Aristizabal, D., Fernando, T., Denman, S., Petersson, L., Aburn, M. J. & Fookes, C., 2020, 'Neural memory networks for seizure type classification', *Proceedings of the 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 569–575.

Al-Hmouz, R., Pedrycz, W., Balamash, A. S. & Morfeq, A., 2018, 'Granular description of data in a non-stationary environment', *Soft Computing*, vol. 22, no. 2, pp. 523–540.

Al Sayaydeh, O. N., Mohammed, M. F., Alhroob, E., Tao, H. & Lim, C. P., 2020, 'A refined fuzzy min-max neural network with new learning procedures for pattern classification', *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 10, pp. 2480 – 2494.

Al Sayaydeh, O. N., Mohammed, M. F. & Lim, C. P., 2019, 'Survey of fuzzy min–max neural network for pattern classification variants and applications', *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 4, pp. 635–645.

Alibart, F., Zamanidoost, E. & Strukov, D. B., 2013, 'Pattern classification by memristive crossbar circuits using ex situ and in situ training', *Nature Communications*, vol. 4, p. 2072.

Altman, N. S., 1992, 'An introduction to kernel and nearest-neighbor nonparametric regression', *The American Statistician*, vol. 46, no. 3, pp. 175–185.

Amit, Y. & Geman, D., 1997, 'Shape quantization and recognition with randomized trees', *Neural Computation*, vol. 9, no. 7, pp. 1545–1588.

Andras, P., 2018, 'Random projection neural network approximation', *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, pp. 2380–2387.

Azad, C. & Jha, V. K., 2016, 'A novel fuzzy min-max neural network and genetic algorithm-based intrusion detection system', *Proceedings of the Second International Conference on Computer and Communication Technologies*, pp. 429–439.

Azad, C. & Jha, V. K., 2017, 'Fuzzy min–max neural network and particle swarm optimization based intrusion detection system', *Microsystem Technologies*, vol. 23, no. 4, pp. 907–918.

Bai, T., Zhang, S., Egleston, B. L. & Vucetic, S., 2018, 'Interpretable representation learning for healthcare via capturing disease progression through time', *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 43–51.

Baldi, P., Sadowski, P. & Whiteson, D., 2014, 'Searching for exotic particles in high-energy physics with deep learning', *Nature Communications*, vol. 5, p. 4308.

Bargiela, A. & Pedrycz, W., 2003, *Granular computing: an introduction*, The Springer international series in engineering and computer science, Springer.

Bargiela, A., Pedrycz, W. & Tanaka, M., 2004, 'An inclusion/exclusion fuzzy hyperbox classifier', *International Journal of Knowledge-based and Intelligent Engineering Systems*, vol. 8, no. 2, pp. 91–98.

Berthold, M. R. & Huber, K.-P., 1998, 'Missing values and learning of fuzzy rules', *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 6, no. 2, pp. 171–178.

Bertsimas, D. & Dunn, J., 2017, 'Optimal classification trees', *Machine Learning*, vol. 106, no. 7, pp. 1039–1082.

Biau, G., Devroye, L. & Lugosi, G., 2008, 'Consistency of random forests and other averaging classifiers', *Journal of Machine Learning Research*, vol. 9, pp. 2015–2033.

Bortolan, G. & Pedrycz, W., 2007, 'Hyperbox classifiers for arrhythmia classification', *Kybernetes*, vol. 36, no. 3/4, pp. 531–547.

Boucheron, S., Bousquet, O. & Lugosi, G., 2005, 'Theory of classification: a survey of some recent advances', *ESAIM: Probability and Statistics*, vol. 9, p. 323–375.

Breiman, L., 1996, 'Bagging predictors', *Machine Learning*, vol. 24, no. 2, pp. 123–140.

Breiman, L., 2001, 'Random forests', *Machine Learning*, vol. 45, no. 1, pp. 5–32.

Breiman, L., Friedman, J., Stone, C. J. & Olshen, R., 1984, *Classification and Regression Trees*, Chapman and Hall/CRC.

Brouwer, R. K., 2002, 'A feed-forward network for input that is both categorical and quantitative', *Neural Networks*, vol. 15, no. 7, pp. 881–890.

Budka, M. & Gabrys, B., 2013, 'Density-preserving sampling: robust and efficient alternative to cross-validation for error estimation', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 1, pp. 22–34.

Burger, C., Redlich, R., Grotegerd, D., Meinert, S., Dohm, K., Schneider, I., Zaremba, D., Förster, K., Alferink, J., Bölte, J., Heindel, W., Kugel, H., Arolt, V. & Dannlowski, U., 2017, 'Differential abnormal pattern of anterior cingulate gyrus activation in unipolar and bipolar depression: an fmri and pattern classification approach', *Neuropsychopharmacology, Nature*, vol. 42, p. 1399.

Cannings, T. I. & Samworth, R. J., 2017, 'Random-projection ensemble classification', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 79, no. 4, pp. 959–1035.

Canzanese, R., Mancoridis, S. & Kam, M., 2015, 'Run-time classification of malicious processes using system call analysis', *Proceedings of the 10th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 21–28.

Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H. & Rosen, D. B., 1992, 'Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps', *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 698–713.

Carpenter, G. A., Grossberg, S. & Rosen, D. B., 1991, 'Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system', *Neural Networks*, vol. 4, no. 6, pp. 759 – 771.

Castillo, P. R. D. & Cardenosa, J., 2012, 'Fuzzy min-max neural networks for categorical data: application to missing data imputation', *Neural Computing and Applications*, vol. 21, no. 6, pp. 1349–1362.

Chang, C.-C. & Lin, C.-J., 2011, 'Libsvm: A library for support vector machines', *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27.

Chen, C. P. & Zhang, C.-Y., 2014, 'Data-intensive applications, challenges, techniques and technologies: A survey on big data', *Information Sciences*, vol. 275, pp. 314 – 347.

Chen, T. & Guestrin, C., 2016, 'Xgboost: A scalable tree boosting system', *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794.

Cheng, M.-Y., Prayogo, D. & Wu, Y.-W., 2018, 'Prediction of permanent deformation in asphalt pavements using a novel symbiotic organisms search–least squares support vector regression', *Neural Computing and Applications*, pp. 1–13.

Cheng, Y. & Miao, D., 2011, 'Rule extraction based on granulation order in interval-valued fuzzy information system', *Expert System with Applications*, vol. 38, no. 10, pp. 12249–12261.

Coates, A., Huval, B., Wang, T., Wu, D. J., Ng, A. Y. & Catanzaro, B., 2013, 'Deep learning with cots hpc systems', *Proceedings of the 30th International Conference on International Conference on Machine Learning*, pp. III–1337–III–1345.

Cuturi, M., 2011, 'Fast global alignment kernels', *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pp. 929–936.

Davtalab, R., Dezfoulian, M. H. & Mansoorizadeh, M., 2014, 'Multi-level fuzzy min-max neural network classifier', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 3, pp. 470–482.

Davtalab, R., Parchami, M., Dezfoulian, M. H., Mansourizade, M. & Akhtar, B., 2012, 'M-fmcn: modified fuzzy min-max classifier using compensatory neurons', *Proceedings of the 11th WSEAS international conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, World Scientific and Engineering Academy and Society (WSEAS), 2183081, pp. 77–82.

de Jesús Rubio, J., 2009, 'Sofmls: online self-organizing fuzzy modified least-squares network', *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 6, pp. 1296–1309.

de Jesús Rubio, J., 2017, 'A method with neural networks for the classification of fruits and vegetables', *Soft Computing*, vol. 21, no. 23, pp. 7207–7220.

Demsar, J., 2006, 'Statistical comparisons of classifiers over multiple data sets', *Journal of Machine Learning Research*, vol. 7, pp. 1–30.

Domingos, P., 2012, 'A few useful things to know about machine learning', *Communications of the ACM*, vol. 55, no. 10, p. 78–87.

Dorigo, M. & Stutzle, T., 2004, *Ant Colony Optimization*, The MIT Press.

Dua, D. & Graff, C., 2019, 'UCI machine learning repository', <http://archive.ics.uci.edu/ml>.

Durrant, R. J. & Kabán, A., 2015, 'Random projections as regularizers: learning a linear discriminant from fewer observations than dimensions', *Machine Learning*, vol. 99, no. 2, pp. 257–286.

Eastwood, M. & Gabrys, B., 2011, 'Model level combination of tree ensemble hyperboxes via gfmm', *Proceedings of The Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, , vol. 1pp. 443–447.

Eastwood, M. & Jayne, C., 2014, 'Evaluation of hyperbox neural network learning for classification', *Neurocomputing*, vol. 133, pp. 249–257.

Eisinga, R., Heskes, T., Pelzer, B. & Te Grotenhuis, M., 2017, 'Exact p-values for pairwise comparison of friedman rank sums, with application to comparing classifiers', *BMC Bioinformatics*, vol. 18, no. 1, p. 68.

Fauber, S. & Schwenker, F., 2015, 'Selective neural network ensembles in reinforcement learning: Taking the advantage of many agents', *Neurocomputing*, vol. 169, pp. 350–357.

Fontama, V., Barga, R. & Tok, W. H., 2015, *Predictive Analytics with Microsoft Azure Machine Learning*, 2nd edn., Apress.

Forghani, Y. & Yazdi, H. S., 2015, 'Fuzzy min-max neural network for learning a classifier with symmetric margin', *Neural Processing Letters*, vol. 42, no. 2, pp. 317–353.

Freund, Y. & Schapire, R. E., 1997, 'A decision-theoretic generalization of on-line learning and an application to boosting', *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119 – 139.

Friedman, J. H., 1997, 'On bias, variance, 0/1—loss, and the curse-of-dimensionality', *Data mining and knowledge discovery*, vol. 1, no. 1, pp. 55–77.

Friedman, J. H., 2001, 'Greedy function approximation: A gradient boosting machine', *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232.

Friedman, M., 1940, 'A comparison of alternative tests of significance for the problem of m rankings', *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92.

Fukunaga, K., 1990, *Introduction to Statistical Pattern Recognition*, 2nd edn., Academic Press Professional, Inc., San Diego, CA, USA.

Gabrys, B., 2001, 'Data editing for neural fuzzy classifier', *Proceedings of the SOCO/ISFI'2001 Conference*, p. 77.

Gabrys, B., 2002a, 'Agglomerative learning algorithms for general fuzzy min-max neural network', *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 32, no. 1, pp. 67–82.

Gabrys, B., 2002b, 'Combining neuro-fuzzy classifiers for improved generalisation and reliability', *Proceedings of the 2002 International Joint Conference on Neural Networks*, pp. 2410–2415.

Gabrys, B., 2002c, 'Neuro-fuzzy approach to processing inputs with missing values in pattern recognition problems', *International Journal of Approximate Reasoning*, vol. 30, no. 3, pp. 149–179.

Gabrys, B., 2004, 'Learning hybrid neuro-fuzzy classifier models from data: to combine or not to combine?', *Fuzzy Sets and Systems*, vol. 147, no. 1, pp. 39–56.

Gabrys, B. & Bargiela, A., 1999, 'Neural networks based decision support in presence of uncertainties', *Journal of Water Resources Planning and Management*, vol. 125, pp. 272–280.

Gabrys, B. & Bargiela, A., 2000, 'General fuzzy min-max neural network for clustering and classification', *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 769–783.

Gabrys, B., Leiviska, K. & Strackeljan, J., 2005, *Do Smart Adaptive Systems Exist? - Best Practice for Selection and Combination of Intelligent Methods*, Springer series on Studies in Fuzziness and Soft Computing, Springer.

Ghasemi, E. & Chow, P., 2017, 'Accelerating apache spark with fpgas', *Concurrency and Computation: Practice and Experience*, vol. e4222, pp. 1–17.

Goodfellow, I. J., Shlens, J. & Szegedy, C., 2015, 'Explaining and harnessing adversarial examples', *Proceedings of the 3rd International Conference on Learning Representations*, .

Grossberg, S., 1980, 'How does a brain build a cognitive code?', *Psychological Review*, vol. 87, no. 1, pp. 1–51.

Grossberg, S., 2013, 'Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world', *Neural networks*, vol. 37, pp. 1–47.

Grzegorzewski, P., 2013, 'Granular regression', *Proceedings of Joint IFSA World Congress and NAFIPS Annual Meeting*, pp. 974–979.

Hastie, T., Robert, T. & Friedman, J., 2009, *The Elements of Statistical Learning: Data mining, Inference and Prediction*, 2nd edn., Springer, New York.

Helwig, N., Pignanelli, E. & Schütze, A., 2015, 'Condition monitoring of a complex hydraulic system using multivariate statistics', *The Proc. of IEEE International Instrumentation and Measurement Technology Conference*, pp. 210–215.

Holm, S., 1979, 'A simple sequentially rejective multiple test procedure', *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70.

Hsu, C., Chang, C. & Lin, C., 2003, 'A practical guide to support vector classification', Tech. rep., Department of Computer Science, National Taiwan University.

Huang, T., He, Y., Dai, D., Wang, W. & Huang, J. Z., 2019, 'Neural network-based deep encoding for mixed-attribute data classification', *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pp. 153–163.

Hulten, G., Spencer, L. & Domingos, P., 2001, 'Mining time-changing data streams', *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 97–106.

Ilager, S. & Prasad, P. S., 2017, 'Scalable mapreduce-based fuzzy min-max neural network for pattern classification', *Proceedings of the 18th International Conference on Distributed Computing and Networking*, ACM, pp. 1–7.

Iman, R. L. & Davenport, J. M., 1980, 'Approximations of the critical region of the fbietkan statistic', *Communications in Statistics - Theory and Methods*, vol. 9, no. 6, pp. 571–595.

Jain, B. & Kolhe, V., 2015, 'Survey on fuzzy min-max neural network classification', *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 12, pp. 30–34.

Jambhulkar, R. K., 2014, 'A review on pattern classification using multilevel and other fuzzy min max neural network classifier', *International Journal of Science and Research*, vol. 3, no. 12, pp. 898–900.

Jen, E., 2003, 'Stable or robust? what's the difference?', *Complexity*, vol. 8, no. 3, pp. 12–18.

Jokar, P., Arianpoo, N. & Leung, V. C. M., 2016, 'Electricity theft detection in ami using customers' consumption patterns', *IEEE Transactions on Smart Grid*, vol. 7, no. 1, pp. 216–226.

Jordan, M. I., 2019, 'Dr. ai or: How i learned to stop worrying and love economics', *Harvard Data Science Review*, vol. 1, no. 1.

Joshi, A., Ramakrishman, N., Houstis, E. N. & Rice, J. R., 1997, 'On neurobiological, neuro-fuzzy, machine learning, and statistical pattern recognition techniques', *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 18–31.

Kachuee, M., Fazeli, S. & Sarrafzadeh, M., 2018, 'Ecg heartbeat classification: A deep transferable representation', *IEEE International Conference on Healthcare Informatics (ICHI)*, pp. 443–444.

Kaggle, 2019, 'Kaggle datasets', <https://www.kaggle.com/datasets>.

Kasabov, N., 2001, 'Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 31, no. 6, pp. 902–918.

Kasabov, N. K., 2007, *Evolving connectionist systems: the knowledge engineering approach*, Springer Science & Business Media.

Kasabov, N. K., 2019, 'Artificial neural networks. evolving connectionist systems', *Time-Space, Spiking Neural Networks and Brain-Inspired Artificial Intelligence*, chap. 2, Springer Berlin Heidelberg, pp. 39–83.

Kasabov, N. K. & Song, Q., 2002, 'Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction', *IEEE transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 144–154.

Kaski, S. & Peltonen, J., 2003, 'Informative discriminant analysis', *Proceedings of the Twentieth International Conference on Machine Learning*, pp. 329–336.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. & Liu, T.-Y., 2017, 'Lightgbm: A highly efficient gradient boosting decision tree', *Advances in Neural Information Processing Systems 30*, pp. 3146–3154.

Khuat, T. & Gabrys, B., 2020, 'A comparative study of general fuzzy min-max neural networks for pattern classification problems', *Neurocomputing*, vol. 386, pp. 110 – 125.

Khuat, T. T., Chen, F. & Gabrys, B., 2020, 'An improved online learning algorithm for general fuzzy min-max neural network', *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9.

Khuat, T. T., Chen, F. & Gabrys, B., 2021a, 'An effective multi-resolution hierarchical granular representation based classifier using general fuzzy min-max neural network', *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 2, pp. 427–441.

Khuat, T. T. & Gabrys, B., 2019, 'Accelerated training algorithms of general fuzzy

min-max neural network using gpu for very high dimensional data', *International Conference on Neural Information Processing*, Springer, pp. 583–595.

Khuat, T. T. & Gabrys, B., 2020, 'An Online Learning Algorithm for a Neuro-Fuzzy Classifier with Mixed-Attribute Data', *arXiv e-prints*, p. arXiv:2009.14670.

Khuat, T. T. & Gabrys, B., 2021a, 'Accelerated learning algorithms of general fuzzy min-max neural network using a novel hyperbox selection rule', *Information Sciences*, vol. 547, pp. 887–909.

Khuat, T. T. & Gabrys, B., 2021b, 'An in-depth comparison of methods handling mixed-attribute data for general fuzzy min–max neural network', *Neurocomputing*, vol. 464, pp. 175–202.

Khuat, T. T. & Gabrys, B., 2021c, 'Random hyperboxes', *IEEE Transactions on Neural Networks and Learning Systems*, vol. (Early Access), pp. 1–15.

Khuat, T. T., Ruta, D. & Gabrys, B., 2021b, 'Hyperbox-based machine learning algorithms: a comprehensive survey', *Soft Computing*, vol. 25, no. 2, pp. 1325–1363.

Kim, H. & Lee, S., 2013, 'Rule extraction from a modified fuzzy min–max neural network for sign language recognition', *Proceedings of the 3rd International Conference on Emerging Trends in Computer and Image Processing*, pp. 182–186.

Kim, H.-J., Lee, J. & Yang, H.-S., 2006, 'A weighted fmm neural network and its application to face detection', King, I., Wang, J., Chan, L.-W. & Wang, D. (eds.) *Proceedings of Neural Information Processing*, Springer Berlin Heidelberg, pp. 177–186.

Kim, H. J., Ryu, T. W., Nguyen, T. T., Lim, J. S. & Gupta, S., 2004, 'A weighted fuzzy min-max neural network for pattern classification and feature extraction', *International Conference on Computational Science and Its Application*, Computational Science and Its Applications – ICCSA 2004, Springer Berlin Heidelberg, pp. 791–798.

Kim, H.-J. & Yang, H.-S., 2005, 'A weighted fuzzy min-max neural network and its application to feature analysis', *Proceedings of International Conference on Natural Computation*, pp. 1178–1181.

Kim, K., Patron, E. R. & Braatz, R. D., 2011, 'Universal approximation with error bounds for dynamic artificial neural network models: A tutorial and some new results', *Proceedings of 2011 IEEE International Symposium on Computer-Aided Control System Design*, pp. 834–839.

Kononenko, I., Simec, E. & Robnik-Sikonja, M., 1997, 'Overcoming the myopia of inductive learning algorithms with relieff', *Appl. Intell*, vol. 7, p. 39.

Kulkarni, S. & Honwadkar, K., 2016, 'Review on classification and clustering using fuzzy neural networks', *International Journal of Computer Applications*, vol. 136, no. 3, pp. 18–23.

Lakshminarayanan, B., Roy, D. M. & Teh, Y. W., 2014, 'Mondrian forests: Efficient online random forests', *Proceedings of the 27th International Conference on Neural Information Processing Systems*, , vol. 2p. 3140–3148.

Li, G., Law, R., Vu, H. Q., Rong, J. & Zhao, X., 2015, 'Identifying emerging hotel preferences using emerging pattern mining technique', *Tourism Management*, vol. 46, pp. 311–321.

Li, T., Ma, S. & Ogihara, M., 2004, 'Entropy-based criterion in categorical clustering', *Proceedings of the Twenty-First International Conference on Machine Learning (ICML)*, pp. 68–75.

Li, T. & Zhou, M., 2016, 'Ecg classification using wavelet packet entropy and random forests', *Entropy*, vol. 18, no. 8.

Likas, A., 2001, 'Reinforcement learning using the stochastic fuzzy min–max neural network', *Neural Processing Letters*, vol. 13, no. 3, pp. 213–220.

Likas, A. & Blekas, K., 1996, 'A reinforcement learning approach based on the fuzzy min-max neural network', *Neural Processing Letters*, vol. 4, no. 3, pp. 167–172.

Liu, J., Ma, Y., Qu, F. & Zang, D., 2020, 'Semi-supervised fuzzy min–max neural network for data classification', *Neural Processing Letters*, vol. 51, pp. 1445—1464.

Liu, J., Ma, Y., Zhang, H., Su, H. & Xiao, G., 2017, 'A modified fuzzy min–max neural network for data clustering and its application on pipeline internal inspection data', *Neurocomputing*, vol. 238, pp. 56–66.

Lughofer, E. & Pratama, M., 2018, 'Online active learning in data stream regression using uncertainty sampling based on evolving generalized fuzzy models', *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 1, pp. 292–309.

Ma, D., Liu, J. & Wang, Z., 2012, 'The pattern classification based on fuzzy min-max neural network with new algorithm', Wang, J., Yen, G. G. & Polycarpou, M. M. (eds.) *Proceedings of International Symposium on Neural Networks*, pp. 1–9.

Ma, Y., Liu, J. & Zhao, Y., 2021, 'Evolved fuzzy min-max neural network for unknown labeled data and its application on defect recognition in depth', *Neural Processing Letters*, vol. 53, p. 85–105.

Macia, N., Bernado-Mansilla, E., Orriols-Puig, A. & Ho, T. K., 2013, 'Learner excellence biased by data set selection: A case for data characterisation and artificial data sets', *Pattern Recognition*, vol. 46, no. 3, pp. 1054 – 1066.

Mallah, C., Cope, J. & Orwell, J., 2013, 'Plant leaf classification using probabilistic integration of shape, texture and margin features', *Signal Processing, Pattern Recognition and Applications*, vol. 5, no. 1, pp. 45–54.

Mamdani, E. H. & Assilian, S., 1975, 'An experiment in linguistic synthesis with a fuzzy logic controller', *International journal of man-machine studies*, vol. 7, no. 1, pp. 1–13.

Martis, R. J., Acharya, U. R., Lim, C. M., Mandana, K. M., Ray, A. K. & Chakraborty, C., 2013, 'Application of higher order cumulant features for cardiac health diagnosis using ecg signals', *International Journal of Neural Systems*, vol. 23, no. 4, p. 1350014.

Maskooki, A., 2013, 'Improving the efficiency of a mixed integer linear programming based approach for multi-class classification problem', *Computers and Industrial Engineering*, vol. 66, no. 2, pp. 383–388.

McCloskey, M. & Cohen, N. J., 1989, 'Catastrophic interference in connectionist networks: The sequential learning problem', *Psychology of Learning and Motivation*, vol. 24, pp. 109–165.

Meneganti, M., Saviello, F. S. & Tagliaferri, R., 1998, 'Fuzzy neural networks for classification and detection of anomalies', *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 848–861.

Mirzaei, A. & Rahmati, M., 2010, 'A novel hierarchical-clustering-combination scheme based on fuzzy-similarity relations', *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 1, pp. 27–39.

Mirzamomen, Z. & Kangavari, M., 2016, 'Fuzzy min-max neural network based decision trees', *Intelligent Data Analysis*, vol. 20, no. 4, pp. 767–782.

Mirzamomen, Z. & Kangavari, M. R., 2017, 'Evolving fuzzy min–max neural network based decision trees for data stream classification', *Neural Processing Letters*, vol. 45, no. 1, pp. 341–363.

Mitchell, T., 1997, *Machine Learning*, McGraw Hill.

Mohammed, M. F. & Lim, C. P., 2015, 'An enhanced fuzzy min-max neural network for pattern classification', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 417–429.

Mohammed, M. F. & Lim, C. P., 2017a, 'Improving the fuzzy min-max neural network with a k-nearest hyperbox expansion rule for pattern classification', *Applied Soft Computing*, vol. 52, pp. 135–145.

Mohammed, M. F. & Lim, C. P., 2017b, 'A new hyperbox selection rule and a pruning strategy for the enhanced fuzzy min–max neural network', *Neural Networks*, vol. 86, pp. 69–79.

Morente-Molinera, J. A., Mezei, J., Carlsson, C. & Herrera-Viedma, E., 2017, 'Improving supervised learning classification methods using multigranular linguistic modeling and fuzzy entropy', *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 5, pp. 1078–1089.

Mosley, L., 2013, *A balanced approach to the multi-class imbalance problem*, Ph.D. thesis, Iowa State University.

Mukhopadhyay, S., Changhong, T., Huang, J., Mulong, Y. & Palakal, M., 2002, 'A comparative study of genetic sequence classification algorithms', *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*, pp. 57–66.

Nandedkar, A. & Biswas, P., 2007a, 'A fuzzy min-max neural network classifier with compensatory neuron architecture', *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 42–54.

Nandedkar, A. & Biswas, P., 2007b, 'A general reflex fuzzy min-max neural network', *Engineering Letters*, vol. 14, no. 1, pp. 1–11.

Nandedkar, A. V. & Biswas, P. K., 2004, 'A fuzzy min-max neural network classifier with compensatory neuron architecture', *Proceedings of the 17th International Conference on Pattern Recognition*, pp. 553–556.

Nandedkar, A. V. & Biswas, P. K., 2006a, 'Object recognition using reflex fuzzy min-max neural network with floating neurons', Kalra, P. K. & Peleg, S. (eds.) *Proceedings of Computer Vision, Graphics and Image Processing*, pp. 597–609.

Nandedkar, A. V. & Biswas, P. K., 2006b, 'A reflex fuzzy min max neural network for granular data classification', *Proceedings of The 18th International Conference on Pattern Recognition*, , vol. 2pp. 650–653.

Nandedkar, A. V. & Biswas, P. K., 2008, 'Reflex fuzzy min max neural network for semi-supervised learning', *Journal of Intelligent Systems*, vol. 17, no. 1-3, pp. 5–18.

Nandedkar, A. V. & Biswas, P. K., 2009, 'A granular reflex fuzzy min-max neural network for classification', *IEEE Transactions on Neural Networks*, vol. 20, no. 7, pp. 1117–1134.

Nauck, D. D., 2003, 'Measuring interpretability in rule-based classification systems', *The 12th IEEE International Conference on Fuzzy Systems*, , vol. 1pp. 196–201.

Nauck, D. D., 2005, *Neuro-Fuzzy Systems for Explaining Data Sets*, Springer Berlin Heidelberg, pp. 305–319.

Nugent, C. & Cunningham, P., 2005, 'A case-based explanation system for black-box systems', *Artificial Intelligence Review*, vol. 24, no. 2, pp. 163–178.

Olivas, E. S., Guerrero, J. D. M., Martinez-Sober, M., Magdalena-Benedito, J. R. & Lopez, A. J. S., 2009, *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 1st edn., IGI Global.

Olson, C. F., 1998, 'A probabilistic formulation for hausdorff matching', *Proceedings of 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 150–156.

Palmer-Brown, D. & Jayne, C., 2011, 'Hypercube neural network algorithm for classification', *International Conference on Engineering Applications of Neural Networks, IFIP Advances in Information and Communication Technology*, Engineering Applications of Neural Networks, pp. 41–51.

Park, B. J., Jang, E. H., Kim, S. H. & Chung, M. A., 2014, 'A study on hyperbox classifier with domino extension in pattern recognition: Hyperbox driven classifier in pattern recognition', *Proceedings of International Conference on Information Science, Electronics and Electrical Engineering*, pp. 1585–1589.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E., 2011a, 'Scikit-learn: Machine learning in python', *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O. et al., 2011b, 'Scikit-learn: Machine learning in python', *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830.

Pedrycz, W., 1998, 'Conditional fuzzy clustering in the design of radial basis function neural networks', *IEEE Transactions on Neural Networks*, vol. 9, no. 4, pp. 601–612.

Pedrycz, W., 2014, 'Allocation of information granularity in optimization and decision-making models: Towards building the foundations of granular computing', *European Journal of Operational Research*, vol. 232, no. 1, pp. 137 – 145.

Pedrycz, W., Succi, G., Sillitti, A. & Iljazi, J., 2015, 'Data description: A general framework of information granules', *Knowledge-Based Systems*, vol. 80, pp. 98 – 108.

Peters, G., 2011, 'Granular box regression', *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 6, pp. 1141–1152.

Polikar, R., Upda, L., Upda, S. S. & Honavar, V., 2001, 'Learn++: an incremental learning algorithm for supervised neural networks', *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 31, no. 4, pp. 497–508.

Politis, D. N., Romano, J. P. & Wolf, M., 1999, *Subsampling*, Springer Science & Business Media.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V. & Gulin, A., 2018, 'Catboost: Unbiased boosting with categorical features', *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, p. 6639–6649.

Putnam, A., Caulfield, A. M., Chung, E. S., Chiou, D., Constantinides, K. & al., e., 2014, 'A reconfigurable fabric for accelerating large-scale datacenter services', *Proceeding of the 41st annual international symposium on Computer architecuture*, IEEE Press, 2665678, pp. 13–24.

Pytorch, 2021, <https://github.com/pytorch/pytorch>, [Online; accessed 15-March-2021].

Quteishat, A. & Lim, C. P., 2008, 'A modified fuzzy min–max neural network with rule extraction and its application to fault detection and classification', *Applied Soft Computing*, vol. 8, no. 2, pp. 985–995.

Quteishat, A., Lim, C. P. & Tan, K. S., 2010, 'A modified fuzzy min-max neural network with a genetic-algorithm-based rule extractor for pattern classification', *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 3, pp. 641–650.

Ramos, G. N., Dong, F. & Hirota, K., 2008, 'Hyperbox classifier with ant colony optimization', *SCIS & ISIS*, vol. 2008, pp. 1714–1718.

Ramos, G. N., Hatakeyama, Y., Dong, F. & Hirota, K., 2009, 'Hyperbox clustering with ant colony optimization (haco) method and its application to medical risk profile recognition', *Applied Soft Computing*, vol. 9, no. 2, pp. 632–640.

Reyes-Galaviz, O. F. & Pedrycz, W., 2015, 'Granular fuzzy modeling with evolving hyperboxes in multi-dimensional space of numerical data', *Neurocomputing*, vol. 168, pp. 240–253.

Rizzi, A., Mascioli, F. M. F. & Martinelli, G., 1998, 'Adaptive resolution min-max classifier', *Proceedings of IEEE International Conference on Fuzzy Systems, IEEE World Congress on Computational Intelligence*, pp. 1435–1440.

Rizzi, A., Panella, M. & Mascioli, F. M. F., 2002, 'Adaptive resolution min-max classifiers', *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 402–414.

Rizzi, A., Panella, M., Mascioli, F. M. F. & Martinelli, G., 2000, 'A recursive algorithm for fuzzy min-max networks', *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, Neural Computing: New Challenges and Perspectives for the New Millennium*, pp. 541–546.

Rodriguez, J. J., Kuncheva, L. I. & Alonso, C. J., 2006, 'Rotation forest: A new classifier ensemble method', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1619–1630.

Rubio, J. d. J., Ricardo Cruz, D., Elias, I., Ochoa, G., Balcazarand, R. & Aguilar, A., 2019, 'Anfis system for classification of brain signals', *Journal of Intelligent & Fuzzy Systems*, , no. Preprint, pp. 1–9.

Rudin, C., 2019, 'Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead', *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215.

Ruta, D. & Gabrys, B., 2005, 'Classifier selection for majority voting', *Information fusion*, vol. 6, no. 1, pp. 63–81.

Sakai, T., Niu, G. & Sugiyama, M., 2018, 'Semi-supervised auc optimization based on positive-unlabeled learning', *Machine Learning*, vol. 107, no. 4, pp. 767–794.

Salvador, M., Budka, M. & Gabrys, B., 2016, 'Effects of change propagation resulting from adaptive preprocessing in multicomponent predictive systems', *Procedia Computer Science*, vol. 96, pp. 713–722.

Salvador, S. & Chan, P., 2004, 'Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms', *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, pp. 576–584.

Sampson, C. J., Arnold, R., Bryan, S. & et al., 2019, 'Transparency in decision modelling: What, why, who and how?', *PharmacoEconomics*.

Sayaydeh, O. N., Mohammed, M. F. & Lim, C. P., 2019, 'A survey of fuzzy min-max neural networks for pattern classification: variants and applications', *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 4, pp. 635–645.

Seera, M. & Lim, C. P., 2014, 'Online motor fault detection and diagnosis using a hybrid fmm-cart model', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 4, pp. 806–812.

Seera, M., Lim, C. P., Ishak, D. & Singh, H., 2012, 'Fault detection and diagnosis of induction motors using motor current signature analysis and a hybrid fmm-cart model', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 1, pp. 97–108.

Seera, M., Lim, C. P., Loo, C. K. & Singh, H., 2015, 'A modified fuzzy min–max neural network for data clustering and its application to power quality monitoring', *Applied Soft Computing*, vol. 28, pp. 19–29.

Seera, M., Lim, C. P., Loo, C. K. & Singh, H., 2016, 'Power quality analysis using a hybrid model of the fuzzy min-max neural network and clustering tree', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 12, pp. 2760–2767.

Seera, M., Randhawa, K. & Lim, C. P., 2018, 'Improving the fuzzy min–max neural network performance with an ensemble of clustering trees', *Neurocomputing*, vol. 275, pp. 1744–1751.

Shafaei, M., Samghabadi, N. S., Kar, S. & Solorio, T., 2020, 'Age suitability rating: Predicting the mpaa rating based on movie dialogues', *Proceedings of the 12th Language Resources and Evaluation Conference*, pp. 1327–1335.

Shinde, S. & Kulkarni, U., 2016, 'Extracting classification rules from modified fuzzy min–max neural network for data with mixed attributes', *Applied Soft Computing*, vol. 40, pp. 364–378.

Simpson, P. K., 1992, 'Fuzzy min-max neural networks. i. classification', *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 776–786.

Simpson, P. K., 1993, 'Fuzzy min-max neural networks - part 2: Clustering', *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 1, p. 32.

Sonule, P. M. & Shetty, B. S., 2017, 'An enhanced fuzzy min–max neural network with ant colony optimization based-rule-extractor for decision making', *Neurocomputing*, vol. 239, pp. 204–213.

Sugeno, M., 1985, *Industrial applications of fuzzy control*, Elsevier Science Inc.

Suykens, J. A. & Vandewalle, J., 1999, 'Least squares support vector machine classifiers', *Neural processing letters*, vol. 9, no. 3, pp. 293–300.

Tagliaferri, R., Eleuteri, A., Meneganti, M. & Barone, F., 2001, 'Fuzzy min–max neural networks: from classification to regression', *Soft Computing*, vol. 5, no. 1, pp. 69–76.

Tariq, H., Eldridge, E. & Welch, I., 2018, 'An efficient approach for feature construction of high-dimensional microarray data by random projections', *PLOS ONE*, vol. 13, no. 4, pp. 1–8.

Thawonmas, R. & Abe, S., 1997, 'A novel approach to feature selection based on analysis of class regions', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 27, no. 2, pp. 196–207.

Tin Kam Ho, 1998, 'The random subspace method for constructing decision forests', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844.

Tung, A. K. H., Xu, X. & Ooi, B. C., 2005, 'Curler: Finding and visualizing nonlinear correlated clusters', *Proceedings of the ACM SIGMOD International Conference*, pp. 467–478.

Upasani, N. & Om, H., 2019, 'A modified neuro-fuzzy classifier and its parallel implementation on modern gpus for real time intrusion detection', *Applied Soft Computing*, vol. 82, p. 105595.

Vapnik, V., 2000, *The Nature of Statistical Learning Theory*, Springer, New York.

Vergara, A., Fonollosa, J., Mahiques, J., Trincavelli, M., Rulkov, N. & Huerta, R., 2013, 'On the performance of gas sensor arrays in open sampling systems using inhibitory support vector machines', *Sensors and Actuators B: Chemical*, vol. 185, pp. 462 – 477.

Wang, G., Yang, J. & Xu, J., 2017, 'Granular computing: from granularity optimization to multi-granularity joint problem solving', *Granular Computing*, vol. 2, no. 3, pp. 105–120.

Xu, G. & Papageorgiou, L. G., 2009, 'A mixed integer optimisation model for data classification', *Computers & Industrial Engineering*, vol. 56, no. 4, pp. 1205–1215.

Xu, J., Wang, G., Li, T. & Pedrycz, W., 2018, 'Local-density-based optimal granulation and manifold information granule description', *IEEE Transactions on Cybernetics*, vol. 48, no. 10, pp. 2795–2808.

Xu, W. & Yu, J., 2017, 'A novel approach to information fusion in multi-source datasets: A granular computing viewpoint', *Information Sciences*, vol. 378, pp. 410 – 423.

Yang, B.-S., Han, T. & Kim, Y.-S., 2004, 'Integration of art-kohonen neural network and case-based reasoning for intelligent fault diagnosis', *Expert Systems with Applications*, vol. 26, no. 3, pp. 387–395.

Yang, L., Liu, S., Tsoka, S. & Papageorgiou, L. G., 2015, 'Sample re-weighting hyper box classifier for multi-class data classification', *Computers and Industrial Engineering*, vol. 85, pp. 44–56.

Ye, J., 2007, 'Least squares linear discriminant analysis', *Proceedings of the 24th international conference on Machine learning*, pp. 1087–1093.

Zadeh, L. A., 1997, 'Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic', *Fuzzy Sets and Systems*, vol. 90, no. 2, pp. 111 – 127.

Zakaryazad, A. & Duman, E., 2016, 'A profit-driven artificial neural network (ann) with applications to fraud detection and direct marketing', *Neurocomputing*, vol. 175, pp. 121–131.

Zelnik-Manor, L. & Perona, P., 2004, 'Self-tuning spectral clustering', *Proceedings*

of the 17th International Conference on Neural Information Processing Systems, pp. 1601–1608.

Zhang, C. & Ma, Y., 2012, *Ensemble machine learning: methods and applications*, Springer.

Zhang, H., 2004, 'The optimality of naive bayes', *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*, p. 562–567.

Zhang, H., Liu, J., Ma, D. & Wang, Z., 2011, 'Data-core-based fuzzy min-max neural network for pattern classification', *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2339–2352.

Zhang, X.-M. & Han, Q.-L., 2017, 'State estimation for static neural networks with time-varying delays based on an improved reciprocally convex inequality', *IEEE transactions on neural networks and learning systems*, vol. 29, no. 4, pp. 1376–1381.