

hyperbox-brain: A Python Toolbox for Hyperbox-based Machine Learning Algorithms

Thanh Tung Khuat

*Complex Adaptive Systems Lab, The Data Science Institute
University of Technology Sydney
Sydney, NSW 2007, Australia*

THANHTUNG.KHUAT@UTS.EDU.AU

Bogdan Gabrys

*Complex Adaptive Systems Lab, The Data Science Institute
University of Technology Sydney
Sydney, NSW 2007, Australia*

BOGDAN.GABRYS@UTS.EDU.AU

Editor: ABC

Abstract

Hyperbox-based machine learning algorithms are an important and popular branch of machine learning in the construction of classifiers using fuzzy sets and logic theory and neural network architectures. This type of learning is characterised by many strong points of modern predictors such as a high scalability, explainability, online adaptation, effective learning from a small amount of data, native ability to deal with missing data and accommodating new classes. Nevertheless, there is no comprehensive existing package for hyperbox-based machine learning which can serve as a benchmark for research and allow non-expert users to apply these algorithms easily. **hyperbox-brain** is an open-source Python library implementing the leading hyperbox-based machine learning algorithms. This library exposes a unified API which closely follows and is compatible with the renowned **scikit-learn** and **numpy** toolboxes. The library may be installed from Python Package Index (PyPI) and the **conda** package manager and is distributed under the GPL-3 license. The source code, documentation, detailed tutorials, and the full descriptions of the API are available at <https://uts-caslab.github.io/hyperbox-brain>.

Keywords: Hyperbox-based machine learning, hyperbox fuzzy sets, fuzzy min-max neural networks, general fuzzy min-max neural network, explainable machine learning, classifier

1. Introduction

This **hyperbox-brain** toolbox has been developed by the researchers within the Complex Adaptive Systems laboratory at the University Technology Sydney. It is a result of many years of developing versatile machine learning algorithms with hyperboxes as the foundational representation element at their core.

Hyperbox-based machine learning algorithms use min-max hyperboxes as their fundamental building blocks to partition the sample space into various regions. A collection of hyperboxes representing the same class can form the regions of arbitrary shape and complexity. Each min-max hyperbox is usually characterised by the minimum and maximum vertices together with a fuzzy membership function acting as a distance or similarity measure. During the training procedure, these hyperboxes are formed, as needed, and adjusted

to accommodate the incoming input samples based on the degree-of-fit of each input pattern to given hyperboxes expressed by their membership values. The use of hyperboxes for learning systems can deal effectively with the pattern classification and clustering problems (Khuat and Gabrys, 2020a). Moreover, this kind of learning exposes numerous essential properties for lifelong machine learning systems (Hamker, 2001; Crowder et al., 2020) such as scalability, explainability, incremental adaptation in dynamically changing environments, continuously learning ability from a limited amount of data and new samples, absorption of new knowledge as well as accommodating new classes with a better ability to avoid catastrophic forgetting, due to their local data cluster representations, and capability to manage the stability-plasticity dilemma (McCloskey and Cohen, 1989).

According to a recent survey (Khuat et al., 2021b), hyperbox-based machine learning algorithms can be divided into three main groups as illustrated in Fig. 1. The first group includes network structured learning models. This group contains two types of learning algorithms. The first sub-group allows the occurrence of overlapped areas among hyperboxes representing different class labels, while the hyperboxes belonging to different classes generated by the algorithms in the other sub-group are not allowed to overlap with each other. The second main group consists of hybrid tree and network structured models. The last group includes non-network structured models. The learning algorithms supported by this initial release of the `hyperbox-brain` library primarily come from the network structured models with non-overlapping inter-class hyperboxes.

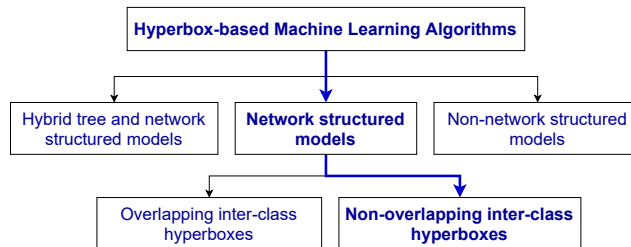


Figure 1: Taxonomy of hyperbox-based machine learning algorithms.

The use of hyperbox fuzzy sets as fundamental representation blocks for machine learning based classifiers dates back to the early 1990s with the most prominent early works including a fuzzy min-max neural network (FMNN) (Simpson, 1992) proposed by Simpson and the variations and extension to the Adaptive Resonance Theory (ART) such as Fuzzy ART (Carpenter et al., 1991b) and ARTMAP (Carpenter et al., 1991a) proposed by Carpenter et al.

This `hyperbox-brain` library focuses mainly on the FMNN and its improved, later variants such as the general fuzzy min-max neural network (GFMMNN) (Gabrys and Bargiela, 2000). Apart from scalability, explainability, incremental adaptation, and continuous learning from limited input samples, learning algorithms of GFMMNN also exhibit unique learning abilities such as learning from interval input data, native learning ability from a mixed labelled and unlabelled training sets (Gabrys and Bargiela, 2000; Gabrys, 2002b), directly learning from data with missing values without the need for data imputation (Gabrys, 2002a), combining all resulting hyperboxes in an ensemble model into a single model (Gabrys, 2004, 2002c), combination of multiple decision trees into a single interpretable hyperbox-based model (Eastwood and Gabrys, 2011), and the capability of

growing and including new classes of data without retraining the whole classifier (Gabrys and Bargiela, 1999). All of these unique learning characteristics are supported by the library, and their details will be presented in the next sections.

Although many hyperbox-based machine learning algorithms have been developed over the years with many very recent examples (Khuat et al., 2021b), there is no comprehensive software library gathering them in one convenient package allowing their easy usage, benchmarking and further development. Therefore, this paper presents a `scikit-learn` compatible hyperbox-based machine learning library in Python to fill this gap and serve as a facilitator for further research and applications in this field.

2. Overview and Design

To achieve a high performance when applying machine learning algorithms for real-world problems, it is necessary to combine learning algorithms with a hyperparameter search, cross-validation, and feature engineering techniques at a large scale. The `hyperbox-brain` library, therefore, is designed to be compatible with the `scikit-learn` toolbox (Pedregosa et al., 2011) to take advantage of the availability of cross-validation, feature transformation, hyperparameter optimisation, and model evaluation methods. As a result, each model in `hyperbox-brain` inherits from `BaseEstimator` or `BaseEnsemble` and `ClassifierMixin` in the module `sklearn.base`. By this inheritance mechanism, the library can set hyperparameters using `set_params()`, train a model using `fit(X, y)`, make a prediction via `predict(X)`, and evaluate the trained model on a hold-out dataset using `score()`. To integrate with `scikit-learn`, `hyperbox-brain` employs `numpy`'s structured arrays (Harris et al., 2020) whose data type is a combination of simpler data types.

This library aims to provide users with a wide range of learning algorithm categories suitable for addressing different ML problems and is therefore organised into the following modules:

base: Providing base classes and functions for all hyperbox-based models in the library.

mixed data learner: Containing the specialised estimators which can work on mixed-attribute data. However, categorical features given in a text form can also be encoded by various encoding methods so that they can be processed by the following learning algorithms for the numerical data only (Khuat and Gabrys, 2021a).

incremental learner: Including estimators for numerical data which use the instance based incremental learning approaches.

batch learner: Comprising hyperbox-based learning algorithms for numerical data using batch learning approaches.

multigranular learner: Containing classifiers for numerical data using the multigranularity learning methods.

ensemble learner: Including the combination of hyperbox-based learners integrated with various ensemble learning methods.

utils: Containing utility functions associated with unit tests which can be executed on all supported Python versions by the continuous integration workflow.

3. The hyperbox-brain Library at a Glance

The `hyperbox-brain` library is intended for researchers and practitioners as an easily accessible toolbox of hyperbox-based machine learning algorithms. This library is implemented in Python, providing numerous learning algorithms using hyperboxes as fundamental building blocks for solving classification and clustering problems. Our purpose is to create an easy-to-use package which may be extended by the community, while also providing essential functionality and characteristics to enable researchers and practitioners to benchmark, reproduce, further develop and apply this type of algorithms for their problems.

3.1 Supported Hyperbox-based Machine Learning Algorithms

The `hyperbox-brain` library currently includes 18 hyperbox-based algorithms, which are used to train network-structured learning models without enabling the overlapped areas among inter-class hyperboxes. All of them are original and improved learning algorithms for the fuzzy min-max neural network (Simpson, 1992) and the general fuzzy min-max neural network (Gabrys and Bargiela, 2000). Of these 18 algorithms, there are three mixed-data learners and 15 learners for numerical data. Out of 15 numerical data learners, there are six instance-incremental learners, two batch learners, six ensemble learners, and one multi-granularity learner. These algorithms are summarised in Table 1.

Model	Feature type	Model type	Learning type
EIOL-GFMM (Khuat and Gabrys, 2020b)	Mixed	Single	Instance-incremental
Freq-Cat-Onln-GFMM (Khuat and Gabrys, 2020a)	Mixed	Single	Batch-incremental
OneHot-Onln-GFMM (Khuat and Gabrys, 2020b)	Mixed	Single	Batch-incremental
Onln-GFMM (Gabrys and Bargiela, 2000)	Numerical	Single	Instance-incremental
IOL-GFMM (Khuat et al., 2020)	Numerical	Single	Instance-incremental
FMNN (Simpson, 1992)	Numerical	Single	Instance-incremental
EFMNN (Mohammed and Lim, 2014)	Numerical	Single	Instance-incremental
KNEFMNN (Mohammed and Lim, 2017)	Numerical	Single	Instance-incremental
RFMNN (Al Sayaydeh et al., 2020)	Numerical	Single	Instance-incremental
AGGLO-SM (Gabrys, 2002b)	Numerical	Single	Batch
AGGLO-2 (Gabrys, 2002b)	Numerical	Single	Batch
MRHGRC (Khuat et al., 2021a)	Numerical	Single	Multi-Granularity
Decision-level Bagging (Gabrys, 2002c)	Numerical	Combination	Ensemble
Decision-level Bagging + hyperparameter optimisation for base learners	Numerical	Combination	Ensemble
Model-level Bagging (Gabrys, 2002c)	Numerical	Combination	Ensemble
Model-level Bagging + hyperparameter optimisation for base learners	Numerical	Combination	Ensemble
Random hyperboxes (Khuat and Gabrys, 2021b)	Numerical	Combination	Ensemble
Random hyperboxes + hyperparameter optimisation for base learners	Numerical	Combination	Ensemble

Table 1: Hyperbox-based machine learning algorithms are supported by `hyperbox-brain`.

3.2 Typical Features of the Library

scikit-learn compatibility: The library is designed to be compatible with and benefits from the `scikit-learn` toolbox’s many features including hyperparameter search, model selection and evaluation techniques as well as the pipeline composition approaches (see Section 4). Moreover, the library can be compatible with other hyperparameter optimisation libraries which may be integrated with `scikit-learn` such as `hyperopt` (Bergstra et al., 2013).

Explainability: One of the interesting characteristics of the use of hyperbox fuzzy sets for building pattern classifiers is the explainability of the predicted results (see the Appendix A for more details). The library supports this functionality by possible parallel coordinates plots based visualisation of representative hyperboxes from different classes together with an input pattern to be classified.

Capability of directly handling missing data: General fuzzy min-max neural networks supported by the library have the ability to handle the classification of inputs with missing data directly without the need for replacing or imputing missing values as in other classifiers (Gabrys, 2002a).

Combination of multiple models at the model level: Learning algorithms for the GFMMNN in the library can combine multiple decision trees (Eastwood and Gabrys, 2011) or resulting hyperboxes generated by multiple hyperbox-based models (Gabrys, 2002c) into a single model. This feature contributes to the increase of explainability of ensemble models.

Data editing and pruning approaches: By integrating the repeated cross-validation methods provided by `scikit-learn` and hyperbox-based learning algorithms, evidence from training multiple models can be used for identifying which points from the original data set or the hyperboxes from the generated multiple models should be retained and those that should be edited out (Gabrys, 2001) or pruned (Gabrys, 2004) before further processing.

Native ability to learn from both labelled and unlabelled data: One of the outstanding features of learning algorithms for the GFMMNN is the ability to form classification boundaries between known classes and ability to cluster data and represent them as hyperboxes when labels are not available in the data (Gabrys and Bargiela, 2000; Gabrys, 2002b). Unlabelled hyperboxes may be then labelled based on the evidence of incoming input patterns.

Ability to learn from new classes in an incremental manner: Incremental learning algorithms of hyperbox-based models provided in the library can grow and include new classes of data without the need for retraining the whole classifier (Gabrys and Bargiela, 1999). Incremental learning algorithms themselves can develop new hyperboxes to represent clusters of new data with potentially new labels both in the middle of normal training process and in the operating time where the initial training has been completed. This characteristic is a key feature for life-long learning systems.

Documentation and tutorials: A comprehensive documentation is developed using `sphinx` and `numpydoc` and is provided to users via the *Read the Doc* platform¹. It provides a detailed API reference, essential background, and a wide range of tutorials and

1. <https://hyperbox-brain.readthedocs.io/en/latest/>

examples² under the interactive *Jupyter notebook* to allow new users to explore how the classifiers in the library are used for solving classification problems.

Build robustness: The library uses GitHub Actions for continuous integration. Automated scripts are used for automated testing and building the library under different versions of Python and operating systems. Tests are executed for each commit made to master branch or when a pull request is opened.

Quality assurance: Code in the project follows the PEP8 style standard for Python. In addition, essential utility functions and code blocks with high complexity are accompanied with a set of unit tests. Furthermore, continuous integration is conducted to guarantee backward compatibility and integrate new code in an easy fashion.

Community-based development: We welcome the contributions from the community to the library via collaborative tools such as Git and GitHub. We provide a documented contribution guideline³ to describe various ways that contributors can join and contribute to the library. In addition, GitHub’s issue tracker and discussion are used to discuss ideas and report bugs regarding the library. The `hyperbox-brain` library is distributed under the GPL-3.0 license.

3.3 Installation and Usage

The `hyperbox-brain` toolbox can be downloaded and installed via PyPI using the command `pip install hyperbox-brain` or from `conda-forge` using the command `conda install -c conda-forge hyperbox-brain`. It is also possible to clone the source code directly from GitHub⁴. In this case, the library can be installed by executing the existing setup script in the root directory through the command `python setup.py install`. Once installed, all available hyperbox-based algorithms and functions in the library can be accessed via importing the desirable class within the `hbbrain` module. A simple example of fitting and assessing a model in `hyperbox-brain` is given below. More elaborate examples and tutorials can be accessed via the online documentation.

```

1 >>> from sklearn.datasets import load_iris
2 >>> from sklearn.preprocessing import MinMaxScaler
3 >>> from sklearn.model_selection import train_test_split
4 >>> from sklearn.metrics import accuracy_score
5 >>> from hbbrain.numerical_data.incremental_learner.onln_gfmm import
   OnlineGFMM
6 >>> # Load dataset
7 >>> X, y = load_iris(return_X_y=True)
8 >>> # Normalise features into [0, 1] as required by hyperbox-based models
9 >>> scaler = MinMaxScaler()
10 >>> scaler.fit(X)
11 MinMaxScaler()
12 >>> XX = scaler.transform(X)
13 >>> # Split data into training and testing sets
14 >>> X_train, X_test, y_train, y_test = train_test_split(XX, y, test_size
   =0.3, random_state=42)
15 >>> # Training a model

```

2. https://hyperbox-brain.readthedocs.io/en/latest/tutorials/tutorial_index.html

3. <https://hyperbox-brain.readthedocs.io/en/latest/developers/contributing.html>

4. <https://github.com/UTS-CASLab/hyperbox-brain>

```

16 >>> clf = OnlineGFMM(theta=0.1).fit(X_train, y_train)
17 >>> # Make prediction
18 >>> y_pred = clf.predict(X_test)
19 >>> acc = accuracy_score(y_test, y_pred)
20 >>> print(f'Accuracy = {acc * 100: .2f}%')
21 Accuracy = 97.78%

```

4. scikit-learn Compatibility

A critical property of the compatibility of `hyperbox-brain` with `scikit-learn` is the inheritance and usage of hyperparameter optimisation, model selection and evaluation, and pipeline functionalities. For example,

- `train_test_split` of `scikit-learn` can be used with hyperbox-based models of `hyperbox-brain` as shown in the example in subsection 3.3.
- `cross_val_score` method of `scikit-learn` can be transparently applied to hyperbox-based models for cross-validation evaluation as below:

```

1 >>> # Instantiating a hyperbox-based model
2 >>> clf = OnlineGFMM(theta=0.1)
3 >>> from sklearn.model_selection import cross_val_score
4 >>> # usage cross_val_score on the hyperbox-based model
5 >>> cross_val_score(clf, XX, y, cv=5)
6 array([0.96666667, 0.96666667, 0.86666667, 0.9, 1.])

```

- Hyperparameter search functions such as `grid_search` and `random_search` can be used directly for models in `hyperbox-brain` as described in the online examples⁵.
- Hyperbox-based estimators can be integrated with Pipeline of `scikit-learn` to form a combination of feature engineering methods and classifiers as various examples shown in the online tutorials⁶.

5. Conclusion

`hyperbox-brain` is a free licensed Python toolbox which implements popular hyperbox-based machine learning models. With the high compatibility with `scikit-learn` API, this library provides users with an easy-to-use package of algorithms which can be integrated with existing cross-validation, pipeline, model selection and evaluation techniques to formulate powerful data analytics pipelines. As shown in Fig. 1, there are many existing hyperbox-based learning algorithms in the literature, therefore, we are continuously adding new models, enhancing usability, code quality, unit tests, documents, and tutorials. Finally, we strongly encourage the contributions of the community to expand this free library for the benefit of both researchers and practitioners interested in and able to benefit from this type of ML algorithms.

5. https://hyperbox-brain.readthedocs.io/en/latest/tutorials/hyperparameter_opt.html

6. https://hyperbox-brain.readthedocs.io/en/latest/tutorials/pipline_integration.html

Appendix A. Explainability of Predicted Outcomes

This appendix is dedicated to clarify the explainability of hyperbox-based learning algorithms for the predicted results of a given input pattern through different types of visualisation supported by the `hyperbox-brain` library.

For two dimensional training samples, the library provides a functionality to show the generated hyperboxes and decision boundaries among classes of a trained model, e.g., GFMMNN, by a hyperbox-based machine learning algorithm as in Fig. 2a. For a given two dimensional input pattern and a trained hyperbox-based model, the library shows representative hyperboxes of all class labels joining the prediction to make the predicted class in a two dimensional plane as shown in Fig. 2b. The predicted class for an unseen pattern is the same with the hyperbox which has the highest membership value to that input pattern. Let’s take the GFMMNN (Gabrys and Bargiela, 2000) as an example, the membership degree between a hyperbox and an input sample is computed based on the longest distance between that hyperbox and the input sample over all features. The smaller this distance is, the higher membership value is. Therefore, it is easily observed that the green hyperbox in Fig. 2b is closer to the input pattern than the blue hyperbox. As a result, the predicted class for the input pattern in this case is green.

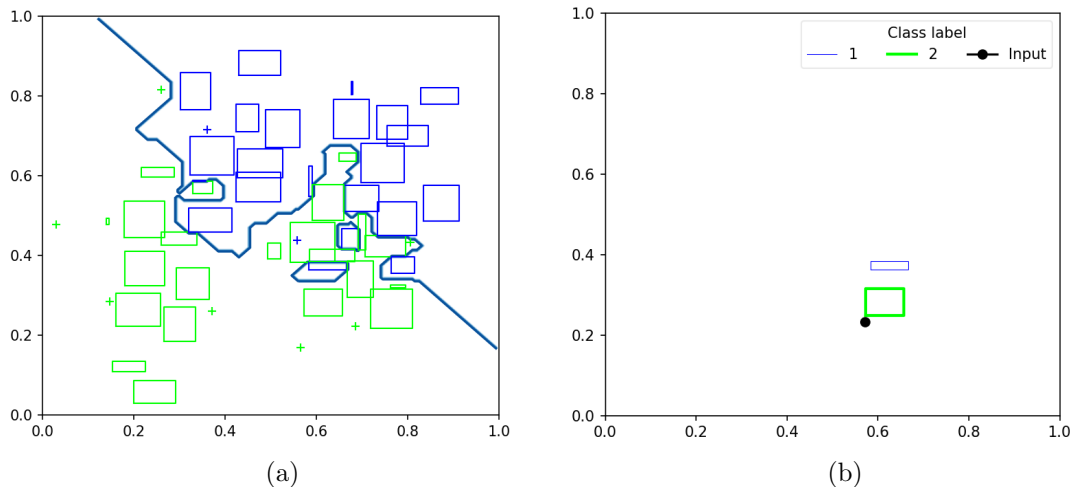


Figure 2: (a) Visualisation of two dimensional hyperboxes and the decision boundaries of a trained GFMMNN. (b) Illustration of two dimensional representative (i.e. winning) hyperboxes from two distinct classes used for the classification of an input pattern as a part of possible suggested decision explanation approach.

The library also provides a general way of explanation for predicted outcomes using the parallel coordinates graph to visualise the coordinates of representative hyperboxes of all classes joining the prediction process as shown in Fig. 3. In this case, the membership value between the green hyperbox and the input pattern is computed based on the fourth feature (F4), which shows the smallest distance compared to the distance values of other hyperboxes to the input pattern. Hence, the predicted class in this case is green.

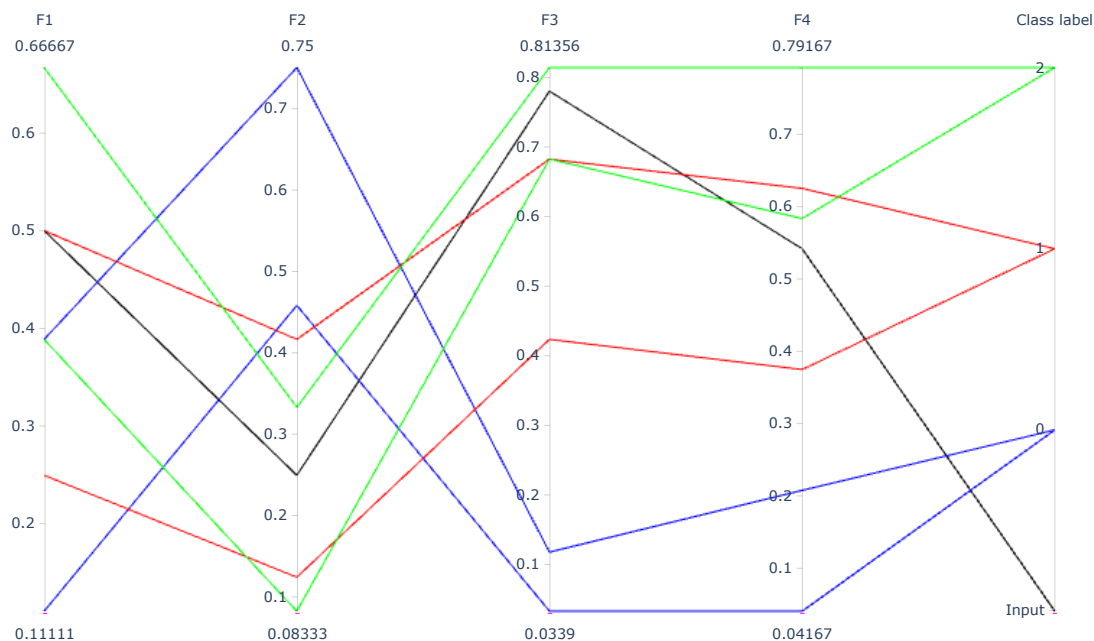


Figure 3: A parallel coordinates graph shows the representative (i.e. winning) hyperboxes for all classes in the context of an input pattern (black color) to be classified. In this case, the predicted class for the input pattern is *green* based on the highest membership value.

References

- Osama Nayel Al Sayaydeh, Mohammed Falah Mohammed, Essam Alhroob, Hai Tao, and Chee Peng Lim. A refined fuzzy min–max neural network with new learning procedures for pattern classification. *IEEE Transactions on Fuzzy Systems*, 28(10):2480–2494, 2020.
- James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of International Conference on Machine Learning*, pages 115–123, 2013.
- Gail A Carpenter, Stephen Grossberg, and John H Reynolds. Artmap: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural networks*, 4(5):565–588, 1991a.
- Gail A Carpenter, Stephen Grossberg, and David B Rosen. Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural networks*, 4(6):759–771, 1991b.
- James A. Crowder, John Carbone, and Shelli Friess. *Methodologies for Continuous, Life-Long Machine Learning for AI Systems*, pages 129–138. Springer International Publishing, 2020.
- Mark Eastwood and Bogdan Gabrys. Model level combination of tree ensemble hyperboxes via gfm. In *Proceedings of The Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, volume 1, pages 443–447, 2011.

- Bogdan Gabrys. Data editing for neural fuzzy classifier. In *Proceedings of the SOCO/-ISFI'2001 Conference*, page 77, 2001. ISBN 3-906454-27-4.
- Bogdan Gabrys. Neuro-fuzzy approach to processing inputs with missing values in pattern recognition problems. *International Journal of Approximate Reasoning*, 30(3):149–179, 2002a. ISSN 0888-613X.
- Bogdan Gabrys. Agglomerative learning algorithms for general fuzzy min-max neural network. *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, 32(1):67–82, 2002b.
- Bogdan Gabrys. Combining neuro-fuzzy classifiers for improved generalisation and reliability. In *Proceedings of the 2002 International Joint Conference on Neural Networks*, volume 3, pages 2410–2415. IEEE, 2002c.
- Bogdan Gabrys. Learning hybrid neuro-fuzzy classifier models from data: to combine or not to combine? *Fuzzy Sets and Systems*, 147(1):39–56, 2004. ISSN 0165-0114.
- Bogdan Gabrys and Andrzej Bargiela. Neural networks based decision support in presence of uncertainties. *Journal of Water Resources Planning and Management*, 125:272–280, 1999.
- Bogdan Gabrys and Andrzej Bargiela. General fuzzy min-max neural network for clustering and classification. *IEEE Transactions on Neural Networks*, 11(3):769–783, 2000.
- Fred H Hamker. Life-long learning cell structures—continuously learning without catastrophic interference. *Neural Networks*, 14(4-5):551–573, 2001.
- Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.
- Thanh Tung Khuat and Bogdan Gabrys. A comparative study of general fuzzy min-max neural networks for pattern classification problems. *Neurocomputing*, 386:110–125, 2020a.
- Thanh Tung Khuat and Bogdan Gabrys. An online learning algorithm for a neuro-fuzzy classifier with mixed-attribute data. *arXiv preprint arXiv:2009.14670*, 2020b.
- Thanh Tung Khuat and Bogdan Gabrys. An in-depth comparison of methods handling mixed-attribute data for general fuzzy min-max neural network. *Neurocomputing*, 464:175–202, 2021a.
- Thanh Tung Khuat and Bogdan Gabrys. Random hyperboxes. *IEEE Transactions on Neural Networks and Learning Systems*, 2021b.
- 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)*, pages 1–9, 2020.

- Thanh Tung Khuat, Fang Chen, and Bogdan Gabrys. An effective multiresolution hierarchical granular representation based classifier using general fuzzy min-max neural network. *IEEE Transactions on Fuzzy Systems*, 29(2):427–441, 2021a.
- Thanh Tung Khuat, Dymitr Ruta, and Bogdan Gabrys. Hyperbox-based machine learning algorithms: a comprehensive survey. *Soft Computing*, 25(2):1325–1363, 2021b.
- Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Elsevier, 1989.
- Mohammed Falah Mohammed and Chee Peng Lim. An enhanced fuzzy min–max neural network for pattern classification. *IEEE Transactions on Neural Networks and Learning Systems*, 26(3):417–429, 2014.
- Mohammed Falah Mohammed and Chee Peng Lim. Improving the fuzzy min-max neural network with a k-nearest hyperbox expansion rule for pattern classification. *Applied Soft Computing*, 52:135–145, 2017.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of Machine Learning Research*, 12:2825–2830, 2011.
- Patrick K Simpson. Fuzzy min—max neural networks—part 1: Classification. *IEEE Transactions on Neural Networks*, 3(5):776–786, 1992.