Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

hyperbox-brain: A Python toolbox for hyperbox-based machine learning algorithms

Thanh Tung Khuat [*], Bogdan Gabrys

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

ARTICLE INFO

Article history:
Received 4 November 2022
Received in revised form 28 May 2023
Accepted 29 May 2023

Keywords:
Hyperbox-based machine learning
Hyperbox fuzzy sets
Fuzzy min–max neural networks
General fuzzy min–max neural network
Explainable machine learning
Classifier

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. The `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.

Code metadata

Current code version	v0.1.5
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-22-00360
Permanent link to Reproducible Capsule	
Legal Code License	GPL-3
Code versioning system used	git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	Python $\geq$ 3.6, scikit-learn $\geq$ 0.24, numpy $\geq$ 1.14.6, scipy $\geq$ 1.1, joblib $\geq$ 0.11, threadpoolctl $\geq$ 2.0.0 pandas $\geq$ 0.25, matplotlib $\geq$ 2.2.3, plotly $\geq$ 4.10
If available Link to developer documentation/manual	https://hyperbox-brain.readthedocs.io/
Support email for questions	thanhtung09t2@gmail.com

1. Motivation and significance

The `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

* Corresponding author.
E-mail address: thanhtung.khuat@uts.edu.au (Thanh Tung Khuat).

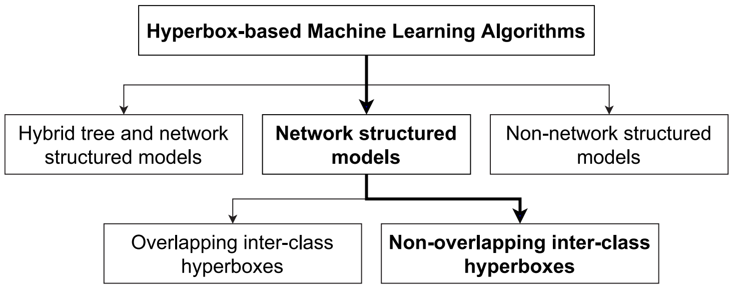
Hyperbox-based Machine Learning Algorithms
Hybrid tree and network structured models
Network structured models
Non-network structured models
Overlapping inter-class hyperboxes
Non-overlapping inter-class hyperboxes

Fig. 1. Taxonomy of hyperbox-based machine learning algorithms.

use of hyperboxes for learning systems can deal effectively with the pattern classification and clustering problems [1]. Moreover, this kind of learning exposes numerous essential properties for lifelong machine learning systems [2,3] 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 [4].

According to a recent survey [5], 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 `hypebox-brain` library primarily come from the network structured models with non-overlapping inter-class hyperboxes.

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) [6] proposed by Simpson and the variations and extension to the Adaptive Resonance Theory (ART) such as Fuzzy ART [7] and ARTMAP [8] 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) [9]. 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 [9,10], directly learning from data with missing values without the need for data imputation [11], combining all resulting hyperboxes in an ensemble model into a single model [12,13], combination of multiple decision trees into a single interpretable hyperbox-based model [14], and the capability of growing and including new classes of data without retraining the whole classifier [15]. 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 [5], 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. Software description

2.1. Software architecture

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 [16] to take advantage of the availability of cross-validation, feature transformation, hyperparameter optimisation, and model evaluation methods. As a result, each model in the `hyperbox-brain` library 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 the `scikit-learn`, the `hyperbox-brain` library employs `numpy`'s structured arrays [17] whose data type is a combination of simpler data types. To accelerate the running speed of learning algorithms, the `hyperbox-brain` library uses matrix formats to store sets of coordinates for lower and upper bounds of hyperboxes. This representation allows us to redefine time-consuming operations of learning algorithms, such as membership computation, expansion constraint checking, hyperbox overlap testing, and similarity computation of hyperboxes, using vector and matrix operations, as presented in [18]. These vector and matrix operations can be accelerated by leveraging the computational power of the `numpy` library. Additionally, the use of matrix operations enables easy parallel execution of learning algorithms on graphics processing units, especially for high-dimensional data [18].

2.2. 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 [6] and the general fuzzy min–max neural network [9]. 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.

2.3. 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 section

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

Model	Feature type	Model type	Learning type
EIOL-GFMM [19]	Mixed	Single	Instance-incremental
Freq-Cat-Onln-GFMM [1]	Mixed	Single	Batch-incremental
OneHot-Onln-GFMM [19]	Mixed	Single	Batch-incremental
Onln-GFMM [9]	Numerical	Single	Instance-incremental
IOL-GFMM [20]	Numerical	Single	Instance-incremental
FMNN [6]	Numerical	Single	Instance-incremental
EFMNN [21]	Numerical	Single	Instance-incremental
KNEFMNN [22]	Numerical	Single	Instance-incremental
RFMNN [23]	Numerical	Single	Instance-incremental
AGGLO-SM [10]	Numerical	Single	Batch
AGGLO-2 [10]	Numerical	Single	Batch
MRHGRC [24]	Numerical	Single	Multi-Granularity
Decision-level Bagging [13]	Numerical	Combination	Ensemble
Decision-level Bagging + hyperparameter optimisation for base learners	Numerical	Combination	Ensemble
Model-level Bagging [13]	Numerical	Combination	Ensemble
Model-level Bagging + hyperparameter optimisation for base learners	Numerical	Combination	Ensemble
Random hyperboxes [25]	Numerical	Combination	Ensemble
Random hyperboxes + hyperparameter optimisation for base learners	Numerical	Combination	Ensemble

and evaluation techniques as well as the pipeline composition approaches (see Section 3.2). Moreover, the library can be compatible with other hyperparameter optimisation libraries which may be integrated with `scikit-learn` such as `hyperopt` [26].
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 Section 3.4 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 [11].
Combination of multiple models at the model level: Learning algorithms for the GFMMNN in the library can combine multiple decision trees [14] or resulting hyperboxes generated by multiple hyperbox-based models [13] 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 the `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 [27] or pruned [12] 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 [9,10]. 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 [15]. 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* platform.[1] It provides a detailed API reference, essential background, and a wide range of tutorials and examples[2] 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 guideline[3] 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.

2.4. Software functionalities

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 [28].

[1] https://hyperbox-brain.readthedocs.io/en/latest/
[2] https://hyperbox-brain.readthedocs.io/en/latest/tutorials/tutorial_index.html
[3] https://hyperbox-brain.readthedocs.io/en/latest/developers/contributing.html

- **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. Illustrative examples

This section presents several examples showcasing the typical functionalities of the `hyperbox-brain` library. Additionally, we have created a detailed tutorial[4] that covers the use of various learning algorithms and fundamental functionalities available within the library.

3.1. 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 GitHub.[5] 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. Listing 1 shows a simple example of fitting and assessing a model in `hyperbox-brain`. More elaborate examples and tutorials can be accessed via the online documentation.

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.preprocessing import
    MinMaxScaler
>>> from sklearn.model_selection import
    train_test_split
>>> from sklearn.metrics import accuracy_score
>>> from hbbrain.numerical_data.
    incremental_learner.onln_gfmm import
    OnlineGFMM
>>> # Load dataset
>>> X, y = load_iris(return_X_y=True)
>>> # Normalise features into [0, 1] as required
    by hyperbox-based models
>>> scaler = MinMaxScaler()
>>> scaler.fit(X)
MinMaxScaler()
>>> XX = scaler.transform(X)
>>> # Split data into training and testing sets
>>> X_train, X_test, y_train, y_test =
    train_test_split(XX, y, test_size=0.3,
    random_state=42)
>>> # Training a model
>>> clf = OnlineGFMM(theta=0.1).fit(X_train,
    y_train)
>>> # Make prediction
>>> y_pred = clf.predict(X_test)
>>> acc = accuracy_score(y_test, y_pred)
>>> print(f'Accuracy = {acc * 100: .2f}\%')
Accuracy =  97.78%
```

Listing 1: Illustrative code example of using the library to train and test an GFMM classifier

3.2. 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 the `scikit-learn` can be used with hyperbox-based models of `hyperbox-brain` as shown in the example in Section 3.1.
- `cross_val_score` method of the `scikit-learn` can be transparently applied to hyperbox-based models for cross-validation evaluation as Listing 2:

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

Listing 2: An example shows how to use classifiers in the library within the cross-validation evaluation module of sklearn

- Hyperparameter search functions such as `grid_search` and `random_search` can be used directly for models in the `hyperbox-brain` as described in the online examples.[6]
- Hyperbox-based estimators can be integrated with `Pipeline` of the `scitkit-learn` to form a combination of feature engineering methods and classifiers as various examples shown in the online tutorials.[7]

3.3. Learning from labelled and unlabelled data

This example demonstrates how to use various learning algorithms of the GFMMNN for learning from datasets including both labelled and unlabelled samples. We refer readers to the tutorial[8] for a complete demonstration. To accomplish this, we created a synthetic training dataset consisting of 250 samples, which includes 29 unlabelled samples, 112 samples labelled as class 1, and 109 samples labelled as class 2. The illustration of this training dataset is presented in Fig. 2. The synthetic testing dataset includes 500 samples labelled as class 1 and 500 samples labelled as class 2. We will compare the performance of a GFMMNN trained on fully labelled training data (125 samples with class 1 and 125 samples with class 2) with that of GFMM models trained on mixed labelled and unlabelled data, based on the resulting hyperboxes and their classification accuracy. In this example, all four learning algorithms use the same maximum hyperbox size parameter ($\theta = 0.2$). Table 2 shows the performance of the four learning algorithms of the GFMMNN trained on fully labelled and mixed labelled and unlabelled data. It can be seen that the accuracy of the learning algorithms trained on the mixed labelled and unlabelled data is very close to that of the algorithms trained on the fully labelled data.

[4] https://hyperbox-brain.readthedocs.io/en/latest/tutorials/tutorial_index.html
[5] https://github.com/UTS-CASLab/hyperbox-brain
[6] https://hyperbox-brain.readthedocs.io/en/latest/tutorials/hyperparameter_opt.html
[7] https://hyperbox-brain.readthedocs.io/en/latest/tutorials/pipline_integration.html
[8] https://hyperbox-brain.readthedocs.io/en/latest/tutorials/learning_from_labelled_unlabelled_data.html

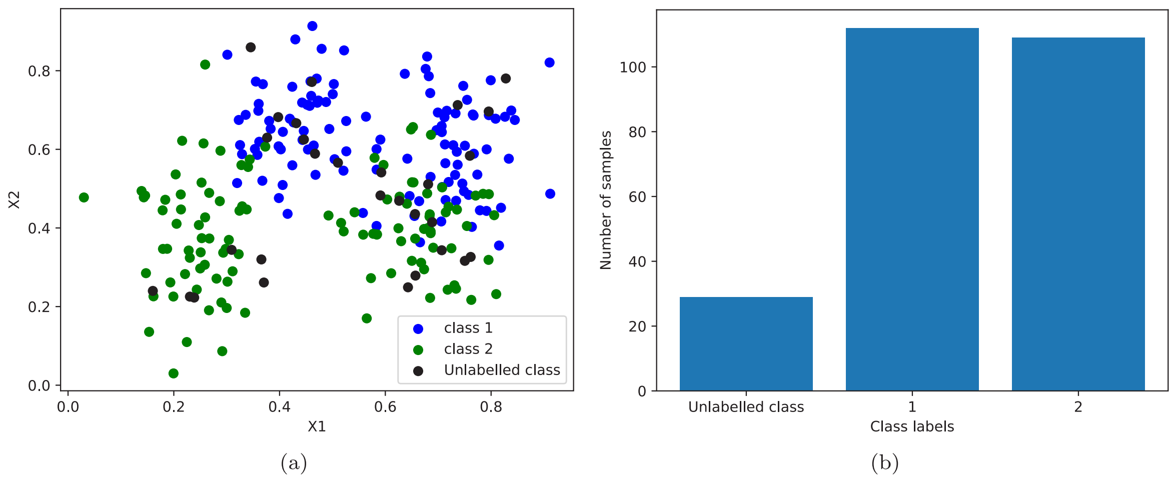

Fig. 2. (a) Illustration of the labelled and unlabelled training dataset. (b) Summary of the number of samples for each class in the training set.

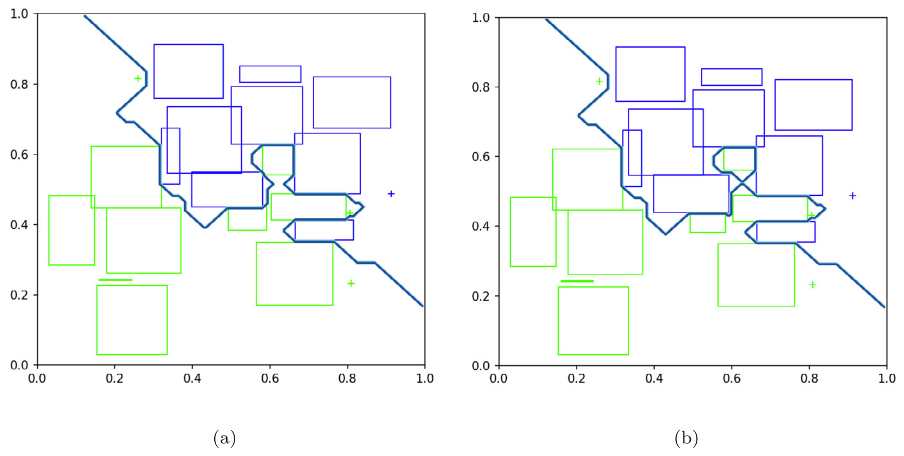

Fig. 3. The hyperboxes generated by the GFMMNN model were trained on (a) fully labelled data and (b) a combination of labelled and unlabelled data. The decision boundaries between two classes are represented by the dark-blue line. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
The results of learning algorithms of the GFMM model trained on fully labelled data, labelled and unlabelled data.

Model	Mode	No. hyperboxes	Accuracy (%)
Onln-GFMM	Fully labelled data	22	86.1
	Labelled and unlabelled data	22	86
IOL-GFMM	Fully labelled data	50	87.9
	Labelled and unlabelled data	51	87.5
AGGLO-2	Fully labelled data	43	87.3
	Labelled and unlabelled data	42	87
AGGLO-SM	Fully labelled data	45	86.2
	Labelled and unlabelled data	45	86.4

Fig. 3 shows the resulting hyperboxes generated from the Onln-GFMM algorithm trained on fully labelled data and mixed labelled and unlabelled data, along with their decision boundaries. It can be observed that in this case, all unlabelled samples were absorbed and assigned appropriate class labels to form hyperboxes. Unlabelled samples located in areas dominated by samples belonging to only one class were correctly classified to the same class. Only samples in the overlapping areas between two classes resulted in a slight difference in the resulting hyperboxes between the model trained on a fully labelled dataset and the model trained on a mixed labelled and unlabelled dataset.

3.4. Explainability of the predicted outcomes

This part is dedicated to clarify the explanability 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. 4(a). 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. 4(b). The predicted class for an unseen pattern is the same with the hyperbox which has the highest membership value to that input pattern. Let us take the GFMMNN [9] 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. 4(b) 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.

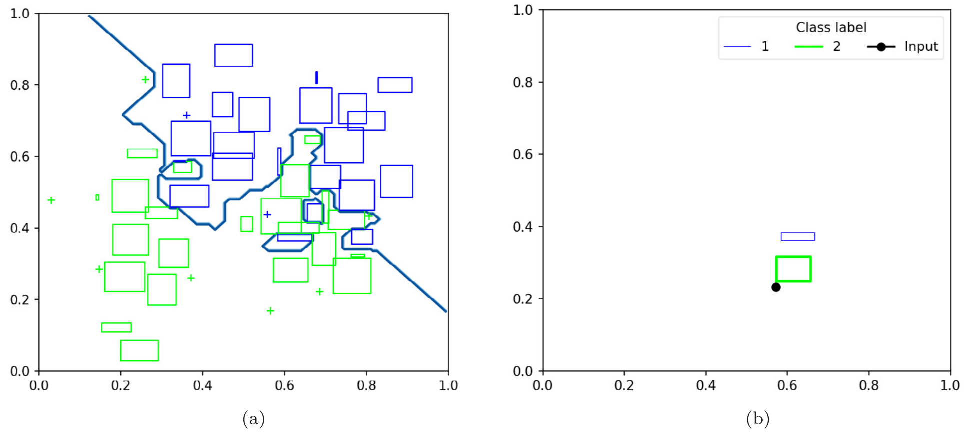

Fig. 4. (a) Visualisation of two dimensional hyperboxes of a trained GFMMNN. The decision boundaries between two classes are represented by the dark-blue line. (b) Illustration of two dimensional representative (i.e. winning) hyperboxes from two distinct classes used for the classification of an input pattern as a part of possible suggested decision explanation approach. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

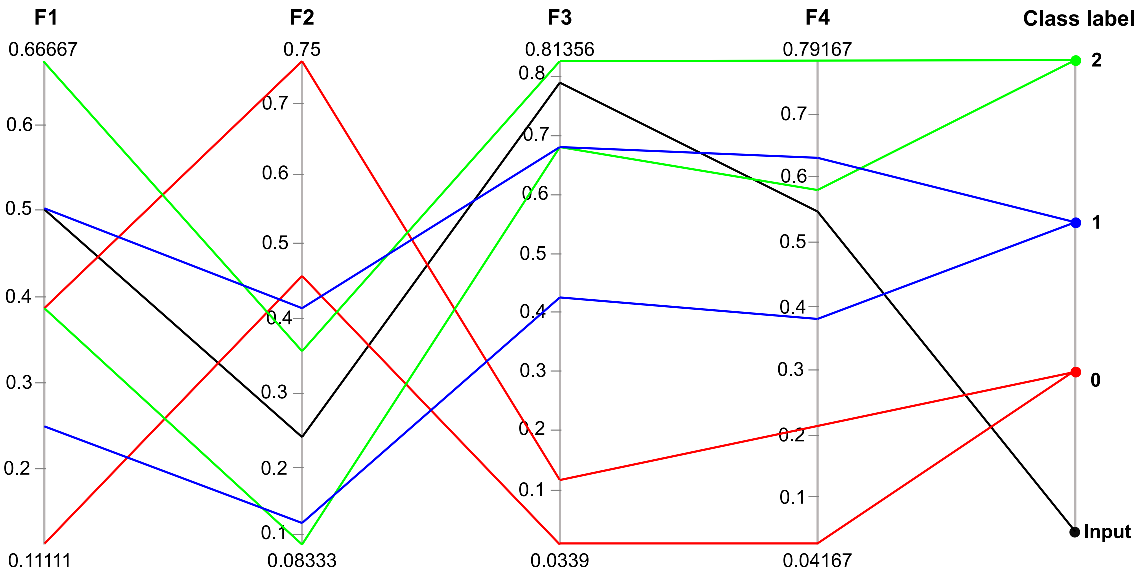

Fig. 5. A parallel coordinates graph shows the representative (i.e. winning) hyperboxes for all classes in the context of an input pattern (black colour) to be classified. In this case, the predicted class for the input pattern is *green* (class 2) based on the highest membership value. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

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. 5. 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.

4. Impact

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. As presented in Section 1, machine learning algorithms based on hyperbox representations exhibits many fundamental characteristics of a modern smart adaptive learning system such as scalability, explainability, incremental learning and adaptation in dynamically changing environments, continuously learning ability from a limited amount of data and new samples. As far as we know, however, there is currently no comprehensive library for this type of machine learning.

The `hyperbox-brain` library is already being used by researchers and developers to build classifiers for their problems. The library has been downloaded more than 6500 times from the

PyPi Python package manager[9] and more than 1900 times from the Anaconda site[10] (until May, 2023).

5. Conclusion

The `hypberbox-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.

CRediT authorship contribution statement

Thanh Tung Khuat: Conceptualization, Methodology, Validation, Software, Writing – original draft. **Bogdan Gabrys:** Conceptualization, Methodology, Writing – review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to the source code and documents in the manuscript

References

[1] Khuat TT, Gabrys B. A comparative study of general fuzzy min-max neural networks for pattern classification problems. Neurocomputing 2020;386:110–25.

[2] Hamker FH. Life-long learning cell structures—Continuously learning without catastrophic interference. Neural Netw 2001;14(4–5):551–73.

[3] Crowder JA, Carbone J, Friess S. Methodologies for continuous, life-long machine learning for AI systems. In: Artificial psychology: psychological modeling and testing of AI systems. Springer International Publishing; 2020, p. 129–38.

[4] McCloskey M, Cohen NJ. Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of learning and motivation, vol. 24. Elsevier; 1989, p. 109–65.

[5] Khuat TT, Ruta D, Gabrys B. Hyperbox-based machine learning algorithms: A comprehensive survey. Soft Comput 2021;25(2):1325–63.

[6] Simpson PK. Fuzzy min—Max neural NetWorks—Part 1: Classification. IEEE Trans Neural Netw 1992;3(5):776–86.

[7] Carpenter GA, Grossberg S, Rosen DB. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural Netw 1991;4(6):759–71.

[8] Carpenter GA, Grossberg S, Reynolds JH. ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. Neural Netw 1991;4(5):565–88.

[9] Gabrys B, Bargiela A. General fuzzy min-max neural network for clustering and classification. IEEE Trans Neural Netw 2000;11(3):769–83.

[10] Gabrys B. Agglomerative learning algorithms for general fuzzy min-max neural network. J VLSI Signal Process Syst Signal Image Video Technol 2002;32(1):67–82.

[11] Gabrys B. Neuro-fuzzy approach to processing inputs with missing values in pattern recognition problems. Internat J Approx Reason 2002;30(3):149–79.

[12] Gabrys B. Learning hybrid neuro-fuzzy classifier models from data: To combine or not to combine? Fuzzy Sets and Systems 2004;147(1): 39–56.

[13] Gabrys B. Combining neuro-fuzzy classifiers for improved generalisation and reliability. In: Proceedings of the 2002 international joint conference on neural networks, vol. 3. IEEE; 2002, p. 2410–5.

[14] Eastwood M, Gabrys B. Model level combination of tree ensemble hyperboxes via GFMM. In: Proceedings of the eighth international conference on fuzzy systems and knowledge discovery, vol. 1. 2011, p. 443–7.

[15] Gabrys B, Bargiela A. Neural networks based decision support in presence of uncertainties. J Water Resour Plan Manag 1999;125:272–80.

[16] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011;12:2825–30.

[17] Harris CR, Millman KJ, Van Der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with numpy. Nature 2020;585(7825):357–62.

[18] Khuat TT, Gabrys B. 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. Springer; 2019, p. 583–95.

[19] Khuat TT, Gabrys B. An online learning algorithm for a neuro-fuzzy classifier with mixed-attribute data. Appl Soft Comput 2023;137:110152.

[20] Khuat TT, Chen F, Gabrys B. An improved online learning algorithm for general fuzzy min-max neural network. In: Proceedings of the international joint conference on neural networks. 2020, p. 1–9.

[21] Mohammed MF, Lim CP. An enhanced fuzzy min–max neural network for pattern classification. IEEE Trans Neural Netw Learn Syst 2014;26(3):417–29.

[22] Mohammed MF, Lim CP. Improving the fuzzy min-max neural network with a K-nearest hyperbox expansion rule for pattern classification. Appl Soft Comput 2017;52:135–45.

[23] Al Sayaydeh ON, Mohammed MF, Alhroob E, Tao H, Lim CP. A refined fuzzy min–max neural network with new learning procedures for pattern classification. IEEE Trans Fuzzy Syst 2020;28(10):2480–94.

[24] Khuat TT, Chen F, Gabrys B. An effective multiresolution hierarchical granular representation based classifier using general fuzzy min-max neural network. IEEE Trans Fuzzy Syst 2021;29(2):427–41.

[25] Khuat TT, Gabrys B. Random hyperboxes. IEEE Trans Neural Netw Learn Syst 2023;34(2):1008–22.

[26] Bergstra J, Yamins D, Cox D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In: Proceedings of international conference on machine learning. 2013, p. 115–23.

[27] Gabrys B. Data editing for neural fuzzy classifier. In: Proceedings of the SOCO/ISFI'2001 conference. 2001, p. 77.

[28] Khuat TT, Gabrys B. An in-depth comparison of methods handling mixed-attribute data for general fuzzy min–max neural network. Neurocomputing 2021;464:175–202.

[9] https://pepy.tech/project/hyperbox-brain

[10] https://anaconda.org/conda-forge/hyperbox-brain