ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

End-to-End Neural Network Training for Hyperbox-Based Classification

Denis Mayr Lima Martins, Christian Lülf, Fabian Gieseke

University of Münster, ERCIS - Department of Information Systems
Leonardo Campus 3, 48149 - Münster, Germany

Abstract. Hyperbox-based classification has been seen as a promising technique in which decisions on the data are represented as a series of orthogonal, multidimensional boxes (i.e., hyperboxes) that are often interpretable and human-readable. However, existing methods are no longer capable of efficiently handling the increasing volume of data many application domains face nowadays. We address this gap by proposing a novel, fully differentiable framework for hyperbox-based classification via neural networks. In contrast to previous work, our hyperbox models can be efficiently trained in an end-to-end fashion, which leads to significantly reduced training times and superior classification results.

1 Introduction

Hyperbox-based classification has been widely studied in the context of machine learning and data mining [1, 2, 3]. The goal of the corresponding approaches is to identify/produce a set of hyperboxes (i.e., multidimensional rectangles) that collectively cover the data of interest (e.g., data points belonging to a class of interest in the context of classification scenarios) [4], as shown in Figure 1. Using hyperboxes to represent regions of interest in the data has various advantages. One of them is that the resulting models can be interpreted more easily. For instance, identifying such hyperboxes allows selecting representative data points or to provide user-friendly predicates/decision rules to describe objects belonging to a specific class. While there is no binary tree associated with such decisions, like it is the case for decision trees, the "individual rules are often simpler" [1]. Another advantage of simple predicates is the fact that they can give rise to orthogonal range queries in low-dimensional sub-spaces, which can efficiently be supported via indexing structures in the context of modern database management systems [5]. These characteristics make hyperbox-based models promising alternatives to classic, opaque models (e.g., deep neural networks) for data-intense tasks in medicine, healthcare, pharmaceutical, and cybersecurity domains [3].

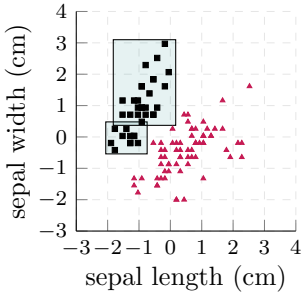


Fig. 1: Hyperbox-based classification for the Iris data set. Only a user-defined target class (black squares) is covered by two axes-aligned boxes.

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

Table 1: Comparison of hyperbox-based classification methods.

Approach	Training	Large d	Large N	End-to-end	Mult. hyperboxes
PRIM [4]	Hill climbing	✗	✗	✗	✓
FMM [6]	Fuzzy membership	✗	✗	✗	✓
HyperNN (Ours)	Gradient-based	✓	✓	✓	✓

Under existing approaches, *patient rule induction method* (PRIM) [1] and *fuzzy min-max neural networks* (FMMs) [6] have been the *de facto* for hyperbox-based classification. These approaches are, however, not yet capable to cope with the increasing amounts of data many domains are confronted with. Also, one generally has little to no control over the number, size, and dimensionality of the induced hyperboxes. In particular, current hyperbox-based neural networks [3] rely on non-differentiable modules, which prevents both end-to-end training via gradient-based optimization and the use of modern optimizers (see Table 1).

In this work, we introduce HyperNN, a novel neural network for hyperbox-based classification method that can be trained in an end-to-end training fashion. We demonstrate via our experimental analysis that HyperNN achieves a competitive if not superior classification performance compared to other state-of-the-art approaches, while reducing both training and inference times. Hence, to the best of our knowledge, this is the first work to propose a fully differentiable, end-to-end approach for hyperbox-based classification, which can be easily adapted via the use of appropriate loss functions and regularizers, and readily combined to modern deep neural networks (e.g., ResNets [7]) for enhanced classification.

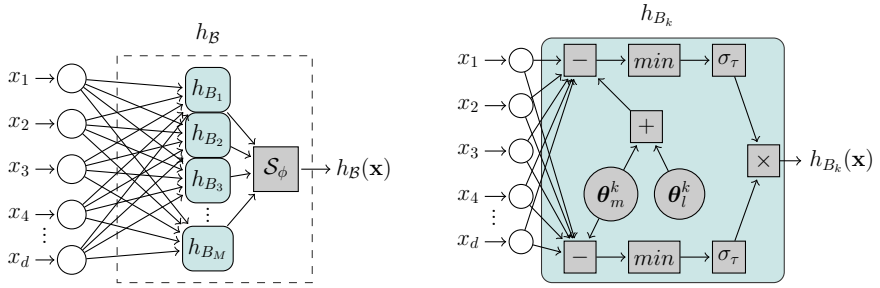
2 Problem Formulation

Given a d-dimensional space, a hyperbox $B = B_{\boldsymbol{\theta}_m, \boldsymbol{\theta}_l} = \{\mathbf{x} \in \mathbb{R}^d \mid \boldsymbol{\theta}_m \leq \mathbf{x} \leq \boldsymbol{\theta}_m + \boldsymbol{\theta}_l\} \subset \mathbb{R}^d$ can be characterized via its minimal point $\boldsymbol{\theta}_m \in \mathbb{R}^d$ along with a vector $\mathbf{0} \leq \boldsymbol{\theta}_l \in \mathbb{R}^d$ containing the length spans. For a point $\mathbf{x} \in \mathbb{R}^d$, let $\mathbb{1}_B(\mathbf{x}) = 1$ if $\mathbf{x} \in B$ and $\mathbb{1}_B(\mathbf{x}) = 0$, otherwise. Accordingly, for the union $\mathcal{B} = \bigcup_{k=1}^{M} B_k$ of M hyperboxes $B_1, \ldots, B_M$, we have $\mathbb{1}_{\mathcal{B}}(\mathbf{x}) = \max(\mathbb{1}_{B_1}(\mathbf{x}), \ldots, \mathbb{1}_{B_M}(\mathbf{x}))$.

We consider binary classification tasks with training sets of the form $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^d \times \{0, 1\}$, where each instance i is represented by a feature vector $\mathbf{x}_i$ and an associated class label y_i. The goal of the learning process is to find a set $B_1, \ldots, B_M$ of M hyperboxes such that the binary classification model $\mathbb{1}_{\mathcal{B}} : \mathbb{R}^d \to \{0, 1\}$ induced by the union $\mathcal{B}$ of those boxes minimizes $G(\mathcal{B}) = {}^1\!/_N \sum_{i=1}^{N} \mathcal{L}(\mathbb{1}_{\mathcal{B}}(\mathbf{x}_i), y_i)$, where $\mathcal{L} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a suitable loss function. Here, we use the binary cross entropy (BCE), which leads to $G_{BCE}(\mathcal{B}) = -{}^1\!/_N \sum_{i=1}^{N} y_i \log(\mathbb{1}_{\mathcal{B}}(\mathbf{x}_i)) + (1 - y_i) \log(1 - \mathbb{1}_{\mathcal{B}}(\mathbf{x}_i))$ as objective.

For the sake of simplicity, this work focuses on binary classification tasks and numerical features. However, our approach can be readily adapted to target other data types such as image and text (with an additional feature extraction step), or alternative tasks such as multi-class classification (by modifying $\mathcal{L}$).

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.



(a) HyperNN Network Structure (b) Hyperbox Neuron

Fig. 2: Architecture of HyperNN.

3 Differentiable Hyperbox-Based Classification

The HyperNN architecture in Figure 2a is similar to the one introduced by Simpson [6], where each neuron in the hidden layer represents a hyperbox characterized by two trainable weight vectors (i.e., model parameters) $\boldsymbol{\theta}_m \in \mathbb{R}^d$ and $\boldsymbol{\theta}_l \in \mathbb{R}^d$. Such hidden neurons are named *hyperbox neurons* thereafter. The number of neurons in the hidden layer corresponds to the maximum number of hyperboxes to be induced, which is controlled by a hyperparameter M.

In a nutshell, the hidden layer is responsible to check for individual hyperbox containment, i.e., each hyperbox neuron checks whether a data instance is covered by its associated hyperbox. The output layer, in turn, consists of a single neuron that checks whether a data instance is contained in *at least one* of the hyperboxes. The sequence of operations performed by each hyperbox neuron is depicted in Figure 2b. We detail these operations next.

Let $h_{\mathcal{B}}$ be a HyperNN network including M hyperbox neurons $h_{B_1}, \ldots, h_{B_M}$, see again Figure 2a. In a first step, for each hyperbox neuron $h_{B_k}, 1 \leq k \leq M$, upper hyperbox bounds are computed as $\boldsymbol{\theta}_u^k = \boldsymbol{\theta}_m^k + \boldsymbol{\theta}_l^k$, where $\boldsymbol{\theta}_m^k$ and $\boldsymbol{\theta}_l^k$ are the two trainable weight vectors of neuron h_{B_k}. Generally, a hyperbox containment check $h_{B_k}(\mathbf{x})$ for a data instance $\mathbf{x} = [x_1, \ldots, x_d]^\top$ could be performed using $h_{B_k}(\mathbf{x}) = \mathbb{1}_{B_k}(\mathbf{x})$. However, such an indicator function formulation would lead to a gradient of zero during backpropagation, which, in turn, would render gradient-based optimization not applicable. Instead, we implement the containment check by computing $\delta_u^k(\mathbf{x}) = \boldsymbol{\theta}_u^k - \mathbf{x}$ and $\delta_m^k(\mathbf{x}) = \mathbf{x} - \boldsymbol{\theta}_m^k$.

Note that, for $\mathbf{x}$ to be covered by the hyperbox represented by neuron h_{B_k}, both $\delta_m^k(\mathbf{x})$ and $\delta_u^k(\mathbf{x})$ must be non-negative for all the d dimensions. As before, in order to obtain meaningful gradient information in the backpropagation phase, we cannot resort to element-wise step functions to check for this property (i.e., $S_j(z) = 1$ if $z \geq 0$, and $S_j(z) = 0$ otherwise, for $j = 1, \ldots, d$). Instead, we resort to a differentiable surrogate applied to the minimum value (across all d dimensions) of both $\delta_m^k(\mathbf{x})$ and $\delta_u^k(\mathbf{x})$, respectively. More precisely, for $\delta_m^k(\mathbf{x})$, we implement this check via a generalized sigmoid function:

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

$$\sigma_\tau(min(\delta_m^k(\mathbf{x}))) = \frac{1}{1+\exp(^{-min(\delta_m^k(\mathbf{x}))}/_\tau)},$$

where τ is a temperature hyperparameter that controls the smoothness of the containment check. Small values of τ lead to an approximation to the original indicator function $\mathbb{1}_{B_k}(\mathbf{x})$, while still providing valuable gradient information. Accordingly, we implement the upper bound check via $\sigma_\tau(min(\delta_u^k(\mathbf{x})))$.

Hence, each hyperbox neuron outputs a value between $[0, 1]$ that expresses the degree of containment of $\mathbf{x}$ within its associated hyperbox.

Likewise, the neural network output $h_{\mathcal{B}}(\mathbf{x})$ must indicate whether at least one of the hyperboxes represented by the hidden neurons contains the input data point $\mathbf{x}$. This could be achieved by simply taking the maximum over all the outputs $h_{B_1}(\mathbf{x}), \ldots, h_{B_K}(\mathbf{x})$.

However, using the maximum only yields gradient information for a single box. Instead, we resort to a smooth maximum function $\mathcal{S}_\phi$ to conduct this step, where values close to one denote containment of $\mathbf{x}$, and ϕ controls smoothness of $\mathcal{S}_\phi$, as follows:

$$\mathcal{S}_\phi(h_{B_1}(\mathbf{x}), \ldots, h_{B_M}(\mathbf{x})) = \frac{\sum_{k=1}^M h_{B_k}(\mathbf{x}) \exp(^{h_{B_k}(\mathbf{x})}/_\phi)}{\sum_{k=1}^M \exp(^{h_{B_k}(\mathbf{x})}/_\phi)}.$$

Overall, we obtain meaningful gradient information via the simple, yet crucial modifications described above, which allows training the networks in an end-to-end fashion.

Training $h_{\mathcal{B}}$ involves finding, for each neuron h_{B_k}, suitable assignments for the associated weight vectors $\boldsymbol{\theta}_m^k$ and $\boldsymbol{\theta}_l^k$, in order to minimize the loss function introduced in Section 2.

4 Experiments and Results

We report an experimental design and analysis on several benchmark datasets, with focus on (1) effectiveness of our approach in comparison to widely-used baselines; (2) efficiency in terms of training and inference times; (3) sensitivity to the number of hyperboxes (M).

Table 2: Data Sets.

Data set	N	d	c
`iris`	150	4	3
`wine`	178	13	3
`cancer`	569	30	2
`blood`	748	5	2
`cars`	1,728	6	4
`satimage`	6,430	36	6
`letter`	20,000	16	26
`sensit`	98,528	100	3
`covtype`	581,012	54	7

4.1 Experimental Design

We consider nine data sets included in the UCI Repository (see Table 2, where c denotes the number of distinct classes). We employ a "one-versus-all" strategy to transform the original task into a binary classification. We use the ratio 70/30 to split the data into training and test sets, and evaluate all methods in terms of F_1-score, training time ($\mathcal{T}_{train}$), and inference time ($\mathcal{T}_{pred}$).

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

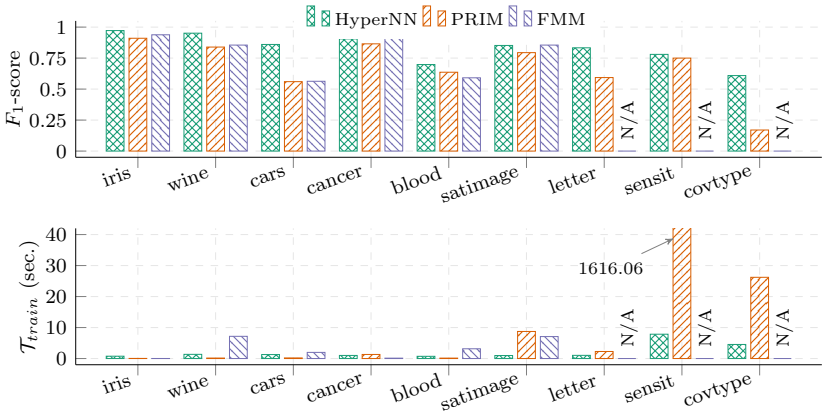


Fig. 3: Mean F_1-score (above) and $\mathcal{T}_{train}$ (below) obtained in our experiments.

For comparison, we use the PRIM implementation provided by David Hadka[1], and the recent FMM implementation by Thanh Tung Khuat[2], while HyperNN is implemented in Python/PyTorch[3]. In all experiments, we conduct hyperparameter tuning using grid search. Best performing models are selected via averaged F_1-score over 5-fold cross-validation. We set the training epochs to $10,000$, with early stopping of 200 epochs when no further improvement is achieved on a holdout validation data set. For HyperNN, we use the Adam optimizer. All experiments are conducted on an Ubuntu 18.04 server with 24 AMD EPYC 7402P cores, 192 GB RAM, and NVIDIA GeForce RTX 3090 GPU. In contrast to HyperNN, both PRIM and FMM *do not* make use of a GPU for fast computations.

4.2 Results

Figure 3 reports results averaged over three runs using different random seeds. Note that we do not report FMM results on the larger data sets, since training time has not been concluded after a pre-defined time limit of ten hours. Both PRIM and FMM achieves high classification performance in terms of F_1-score for all data sets. For large data sets such as `satimage` and `sensit`, however, these results are produced at a cost of high training times. In contrast, HyperNN shows similar classification performance while keeping lower training times for almost all data sets. For `satimage` and `sensit`, HyperNN achieves an F_1-score close to PRIM in a fraction of the training time of the latter.

We also explore how sensitive HyperNN is to changes in its main hyperparameters. Figure 4 shows the effect of M in terms of F_1-score, $\mathcal{T}_{train}$, and $\mathcal{T}_{pred}$, where HyperNN shows a stable scalability and generalization performance for an increasing M. For small datasets, such as `iris`, `wine`, and `cancer`, increasing M

[1] https://github.com/Project-Platypus/PRIM
[2] https://github.com/UTS-AAi/comparative-gfmm
[3] https://github.com/mlde-ms/hypernn

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

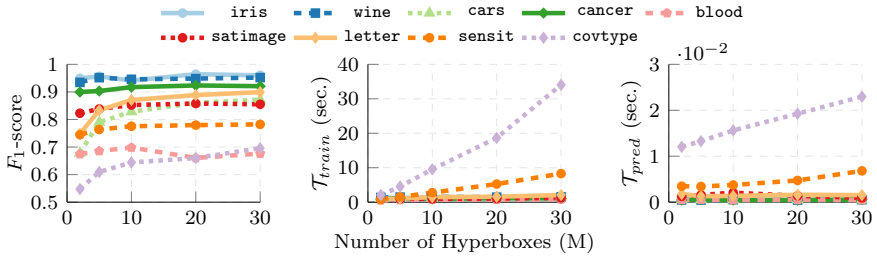


Fig. 4: Effect of M on F_1-score (left), $\mathcal{T}_{train}$ (center), and $\mathcal{T}_{pred}$ (right).

brings almost no benefit in terms of F_1-score. In contrast, for `letter`, `sensit`, and `covtype`, a high M rapidly improves classification performance, at a cost of higher training and prediction times. However, for `blood`, increasing M from 10 to 20 decreases F_1-score due to overfitting. Such a degradation in classification performance could be alleviated by, e.g., an adaptive training procedure where M is adapted (i.e., increased or decreased) if the validation loss deteriorates.

5 Conclusion

We propose HyperNN, a fully differential approach for hyperbox-based classification. We provide an efficient, GPU-ready implementation that produced highly competitive models in terms of both classification and runtime performance, when compared to state-of-the-art techniques such as PRIM and FMM. As future work, we plan to apply HyperNN to image data, in combination with other modern deep learning models (e.g., CNNs, ResNets), where both suitable features and hyperboxes must be learned jointly in an end-to-end fashion.

References

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2 edition, 2009.

[2] Vadim Arzamasov and Klemens Böhm. Reds: Rule extraction for discovering scenarios. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, pages 115–128, New York, NY, USA, 2021. Association for Computing Machinery.

[3] Thanh Tung Khuat, Dymitr Ruta, and Bogdan Gabrys. Hyperbox-based machine learning algorithms: a comprehensive survey. *Soft Computing*, 25(2):1325–1363, 2021.

[4] Jerome H. Friedman and Nicholas I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9(2):123–143, 1999.

[5] Roy Friedman and Rana Shahout. Box queries over multi-dimensional streams. In *Proceedings of the 15th International Conference on Distributed and Event-Based Systems*, DEBS '21, pages 90–101, New York, NY, USA, 2021. Association for Computing Machinery.

[6] P.K. Simpson. Fuzzy min-max neural networks. i. classification. *IEEE Transactions on Neural Networks*, 3(5):776–786, 1992.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.