Confidence in Prediction: An Approach for Dynamic Weighted Ensemble

Confidence in prediction: an approach for dynamic weighted ensemble

Duc Thuan Do[1], Tien Thanh Nguyen[2][0000−0002−7107−5611], The Trung Nguyen[3], Anh Vu Luong[4], Alan Wee-Chung Liew[4], and John McCall[2]

[1] School of Applied Mathematics and Informatics, Hanoi University of Science and Technology, Vietnam
[2] School of Computing Science and Digital Media, Robert Gordon University, Aberdeen, UK
[3] School of Information and Communication Technology, Hanoi University of Science and Technology, Vietnam
[4] School of Information and Communication Technology, Griffith University, Australia
Email: t.nguyen11@rgu.ac.uk

Abstract. Combining classifiers in an ensemble is beneficial in achieving better prediction than using a single classifier. Furthermore, each classifier can be associated with a weight in the aggregation to boost the performance of the ensemble system. In this work, we propose a novel dynamic weighted ensemble method. Based on the observation that each classifier provides a different level of confidence in its prediction, we propose to encode the level of confidence of a classifier by associating with each classifier a credibility threshold, computed from the entire training set by minimizing the entropy loss function with the mini-batch gradient descent method. On each test sample, we measure the confidence of each classifier's output and then compare it to the credibility threshold to determine whether a classifier should be attended in the aggregation. If the condition is satisfied, the confidence level and credibility threshold are used to compute the weight of contribution of the classifier in the aggregation. By this way, we are not only considering the presence but also the contribution of each classifier based on the confidence in its prediction on each test sample. The experiments conducted on a number of datasets show that the proposed method is better than some benchmark algorithms including a non-weighted ensemble method, two dynamic ensemble selection methods, and two Boosting methods.

Keywords: Supervised Learning, Classification, Ensemble Method, Ensemble Learning, Multiple Classifier System, Weighted Ensemble

1 Introduction

In recent years, learning with an ensemble of classifiers (EoC) has enjoyed increased attention in the machine learning community due to its advantage in achieving better prediction than using a single classifier [11]. In ensemble method, the diverse classifiers are obtained by learning different algorithms on a training

set (heterogeneous ensemble) or learning one algorithm on many different training sets (homogeneous ensemble) [12]. Each learning algorithm learns a classifier with the aim of describing the relationship between the features and the class label of the training observations. The generated classifier returns the output in the form of crisp labels (0-1 class memberships) or posterior probabilities (fuzzy class memberships) [8]. A combiner is then used to aggregate the outputs of all classifiers to obtain the final decision.

In the combining method, simple averaging can be conducted on the classifiers' output by assigning equal weights for all individual classifiers. In ensemble systems where individual classifiers exhibit nonidentical strength, unequal weights in the aggregation may achieve a better performance than simple averaging [11]. In this work, we focus on weighted ensemble in which the prediction of each classifier is associated with a weight when combining for final decision. In fact, weighted ensemble is a special case of ensemble pruning (which is also known as selective ensemble or ensemble selection) where the weights on some classifiers are set to zero. In ensemble pruning, the EoC can be obtained via static or dynamic approach. In detail, the static approach learns one optimal EoC on the training data and uses it to assign a label for all test samples. This, therefore, limits the flexibility of the selection procedure. Meanwhile, the dynamic approach selects a classifier or an EoC with the most competencies in a defined region associated with each test sample. Although this approach provides more flexibility than the static approach, the performance of the dynamic approach is dependent on the performance of the techniques that define the region of competence (RoC) [3]. A new weighted ensemble method which benefits from the advantage of both static and dynamic selective method i.e. learning the optimal condition on the training data to select classifiers and then determining particular weights for each test sample would be beneficial.

Our idea for weighted ensemble is based on the observation that classifiers in an ensemble are generated from different methodologies, and therefore have different confidence level in their predictions. On a particular dataset, some classifiers can provide very high confidence in the classification while others can have difficulty in decision when assigning a label for a test sample based on their outputs. We come up with an idea to encode the level of confidence of a classifier by associating with each classifier a credibility threshold, computed from the entire training set by minimizing the entropy loss function with the gradient descent method. We also measure the confidence level of each classifier's prediction on each test sample i.e. how confident it makes the decision. The confidence in the prediction is then compared to the credibility threshold to determine whether the output of the classifier should be included in the aggregation. In a procedure to assign a label for a test sample, when a classifier is attended, its confidence level and the credibility threshold will be used to compute the weight to show its contribution to the aggregation. By this way, the proposed method integrates both static and dynamic approach of ensemble selection through finding the credibility threshold like in the static methods and assigning particular weights for classifiers on each test sample like in dynamic methods.

The contribution of this work are: (i) We propose a measure to qualify the confidence in the output of each classifier. (ii) We propose a dynamic weighted ensemble method to select classifiers based on the confidence of its prediction. (iii) We formulate the optimization problem on the convex entropy loss function to search for the credibility threshold (iv) Experiments on a number of datasets demonstrate that the proposed method is better than several well-known benchmark algorithms.

2 Background and Related work

In ensemble system, the outputs of classifiers are combined to obtain the final discriminative decision. Traditionally, simple combining methods like Sum and Vote are frequently applied to the outputs of base classifiers to predict class labels [13]. In fact, the simple combining method is the special case of the weighted combining method where the classifiers are treated equally in the aggregation, i.e. all classifiers make the equal contributions in the final collaborated decision. In weighted combining methods, each classifier can put different weight on the prediction result and the combining algorithm works by taking M weighted linear combinations of posterior probabilities for the M classes. Several approaches have been proposed to find the weights. In [15], Ting et al. proposed MLR method which depends on solving M Linear Regression models corresponding to the M classes based on meta-data and the training data labels in crisp form to find these combining weights. Yijing et al. [18] proposed the new weighted combining rules in which the weight of each classifier is computed based on its performance on the training data measured by Area under the ROC Curve (AUC). Wu [17] proposed a new ensemble learning paradigm that takes into account information about the performance ordering of the base classifiers reported in previous literature. By measuring the similarity between two learning tasks, the performance ranking of the trained classifiers of a given learning task can be inferred so as to obtain the optimal combining weights of the trained classifiers. Nguyen et al. [10] weighed the base classifiers generated on projected data of training observations by the linear regression model.

Boosting is also a family of weighed ensemble methods. The idea of this approach is to learn weak classifiers with respect to a distribution to form a strong classifier. When weak classifiers are combined, they will have weights which usually are related to the weak classifiers' accuracy. Some well-known examples of the boosting approach are AdaBoost [5] where the weak classifier is tweaked to handle previously misclassified samples, LPBoost [4] where the margin between training samples of different classes is maximized via linear programming, and RUSBoost [14] where imbalanced datasets are handled by learning from skewed training data.

Ensemble pruning is a special case of weighted ensemble in which the weights of some classifiers are set to zero. The purpose of ensemble pruning is to search for a suitable subset of classifiers that is better than using the whole ensemble. In this technique, a single classifier or an EoC can be obtained via static or dynamic approach. The static approach selects only one subset of classifiers during the training phase and uses it to predict for all unseen samples. In the past

years, many statistic ensemble selection methods have been proposed to search for the optimal or sub-optimal subset of ensembles, and they can be grouped into three categories: ordering-based methods [9], clustering-based methods [1], and optimization-based methods via mathematical programming [19], probabilistic pruning [2], or heuristic search [6]. Zhang et al. [19] formulated the ensemble pruning problem as a quadratic integer programming problem and used semi-definite programming to acquire an approximate solution. Although this method outperforms the other heuristics in the author's evaluation, fixing the number of selected base classifiers is a hindrance to efficient performance. Chen et al. [2] propose a probabilistic pruning method which includes a sparsity-inducing prior distribution introduced over the combination weights. The maximum a posteriori estimation of the weights is then acquired by the Expectation Propagation algorithm.

On the other hand, in the dynamic approach, a classifier or an EoC is selected to classify each test sample based on the competence level of the classifiers computed according to some criteria on a local region of the feature space [3]. Here the region to compute competence can be defined by kNN methods [3, 7] and potential functions [16]. Comparison experiments indicated that a simple dynamic selection method like KNORA Union can be competitive or sometimes outperforms more complex methods.

3 Proposed Method

3.1 Problem Formulation

Given the training set $\mathcal{D}$ with N data points and K learning algorithms $\boldsymbol{\mathcal{K}} = [\mathcal{K}_k]$. The base classifier h_k is generated by training $\mathcal{K}_k$ on $\mathcal{D}$. Denote $\mathbf{P} = [p_{k,j}]$, $p_{k,j} = P(y_j = 1|\mathbf{x})$ as the prediction of h_k for a sample $\mathbf{x}$ to class label $y_j = 1$. For example, prediction vector (0.3,0.6,0.1) of a classifier for a sample in a 3-class classification problem means that probabilities this sample belongs to class y_1, y_2, and y_3 are 0.3, 0.6, and 0.1, respectively. To measure the confidence on the prediction of h_k on $\mathbf{x}$, we define $\mathbf{e} = [e_k]$ as the difference between the maximum value among the predictions and the average of the other values. In fact, the class label is assigned based on the maximum value of the posterior probability. By defining e_k, we aim to measure the convincing decision in this decision strategy. The higher e_k results in the bigger gap between the maximum value of the posterior probability and the average of the others, making it the more convincing decision.

$$\begin{cases} e_k = p_{k,s} - \frac{1}{M-1} \sum_{j=1, j\neq s}^{M} p_{k,j} \\ s = \text{argmax}_{j=1,\ldots,M} p_{k,j} \end{cases} \tag{1}$$

Proposition 1: e_k is bounded in $[0, 1]$

As mentioned above, we measure the confidence level in the prediction of each classifier and then compare with the credibility threshold associated with this classifier. If the confidence level is higher than the threshold, the classifier

will be attended to the aggregation. By this way, we define a *'relu-like'* function a_k for activation calculation concerning the confidence threshold $\boldsymbol{\beta} = [\beta_k]$:

$$a_k = \max(0, e_k - \beta_k) \;\; k = 1, \ldots, K \tag{2}$$

The combining vector $\boldsymbol{pc} = [pc_j]$ from all K classification probability vectors for the class y_j $(j = 1, \ldots, M)$ is:

$$pc_j = \sum_{k=1}^{K} a_k p_{k,j} \tag{3}$$

Clearly, when $e_k > \beta_k$, the function $a_k > 0$ that means classifier h_k is activated in the combination in (3) and its contribution to the combination is the value of the activated function a_k. The proposed method is a dynamic weighted ensemble since the weight of each classifier in combining vector is different on each test sample via the different confidence in its prediction. We use softmax function to transform the combination vector to the ensemble classification probability $\boldsymbol{pe} = [pe_j]$ as:

$$pe_j = \frac{e^{pc_j}}{\sum_{j=1}^{M} e^{pc_j}} \tag{4}$$

In this work, the credibility threshold is found by minimizing the convex entropy loss function. This loss function on a data point $(\mathbf{x}, \mathbf{y})$ in which $\mathbf{y}$ is one-hot vector of class label of $\mathbf{x}$ is given by:

$$\mathcal{L}(\boldsymbol{\beta}) = -\sum_{j=1}^{M} y_j \log pe_j \tag{5}$$

3.2 Optimization

We use the gradient descent approach to solve the optimization problem for the function in (5). First, the entropy loss function is transformed to:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\beta}) &= -\sum_{j=1}^{M} y_j \log pe_j = -\sum_{j=1}^{M} y_j \log \frac{e^{pc_j}}{\sum_{j=1}^{M} e^{pc_j}} \\
&= -\sum_{j=1}^{M} \left(y_j pc_j - y_j \log \sum_{j'=1}^{M} e^{pc_{j'}} \right) \\
&= -\sum_{j=1}^{M} y_j pc_j + \log \sum_{j'=1}^{M} e^{pc_{j'}} (\text{Due to } \sum_{j=1}^{M} y_j = 1)
\end{aligned}$$

We compute the gradient of the lost function at each data point $(\mathbf{x}, \mathbf{y})$. The gradient of $\mathcal{L}$ along $\boldsymbol{\beta}$ is $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = \left[\frac{\partial \mathcal{L}}{\partial \beta_k}\right]$ in which each subgradient is computed by:

$$\frac{\partial \mathcal{L}}{\partial \beta_k} = -\sum_{j=1}^{M} y_j \frac{\partial pc_j}{\partial \beta_k} + \frac{\sum_{j=1}^{M} \frac{\partial e^{pc_j}}{\partial \beta_k}}{\sum_{j=1}^{M} e^{pc_j}} \tag{6}$$

In detail:

$$\begin{aligned}
\frac{\partial pc_j}{\partial \beta_k} &= \frac{\partial \left(\sum_{k=1}^{K} a_k p_{k,j}\right)}{\partial \beta_k} \\
&= \frac{\partial \left(a_k p_{k,j}\right)}{\partial \beta_k} \left(\text{Due to } \frac{\partial \left(a_{k'} p_{k',j}\right)}{\partial \beta_k} = 0 \ \forall k' \neq k\right) \\
&= \frac{\partial \left(\max\left(0, e_k - \beta_k\right) p_{k,j}\right)}{\partial \beta_k} \\
&= \begin{cases} 0, & \text{if } e_k \leq \beta_k \\ -p_{k,j}, & \text{if } otherwise \end{cases} \coloneqq g_{k,j} \qquad (7) \\
\frac{\partial e^{pc_j}}{\partial \beta_k} &= e^{pc_j} \frac{\partial pc_j}{\partial \beta_k} = e^{pc_j} g_{k,j} \qquad (8)
\end{aligned}$$

Replace (6) by results in (7) and (8), we have:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \beta_k} &= -\sum_{j=1}^{M} y_j g_{k,j} + \frac{\sum_{j=1}^{M} e^{pc_j} g_{k,j}}{\sum_{j=1}^{M} e^{pc_j}} \\
&= -\sum_{j=1}^{M} y_j g_{k,j} + \sum_{j=1}^{M} pe_j g_{k,j} \\
&= -\langle \mathbf{y}, \mathbf{g}_k \rangle + \langle \boldsymbol{pe}, \mathbf{g}_k \rangle = \langle \boldsymbol{pe} - \mathbf{y}, \mathbf{g}_k \rangle \qquad (9)
\end{aligned}$$

where $\mathbf{g}_k = [g_{k,j}]$

To calculate the gradient of $\mathcal{L}$ along $\boldsymbol{\beta}$ for a mini-batch of n data points, we take the average of gradients of these points as:

$$\frac{\partial \mathcal{L}}{\partial \beta_k} \coloneqq \frac{1}{n} \sum_{i=1}^{n} \langle \boldsymbol{pe}^{(i)} - \mathbf{y}^{(i)}, \mathbf{g}_k^{(i)} \rangle \qquad (10)$$

Now, we update β_k according to gradient descent method with learning rate η_k at k^{th} iteration:

$$\beta_{k+1} = \beta_k - \eta_k \frac{\partial \mathcal{L}}{\partial \beta_k} \qquad (11)$$

In this work, we applied the proposed weighted ensemble method to the heterogeneous ensemble systems where several different learning algorithms learn on one training set to obtain the base classifiers. As these classifiers perform differently on each dataset because of the differences in learning strategies, it is expected to obtain better results than simple aggregation [11]. First, we learn base classifiers $\boldsymbol{\mathcal{H}} = [h_k]$ on $\mathcal{D}$ using the given learning algorithms. The meta-data $\mathbf{P}$ of $\mathcal{D}$ is generated via the T-fold cross validation procedure. Specifically, $\mathcal{D}$ is divided into T disjoint parts. The meta-data of observations in one part is then created by the classifiers generated by training the K learning algorithms on the complement. All meta-data sets from each part are concatenated to generate the meta-data $\mathbf{P}$.

Having $\mathbf{P}$ in hand, we compute the e_k from $\mathbf{P}$ for each $\mathbf{x} \in \mathcal{D}$ by using Eqn. (1). The threshold β_k is initialized to 0. The algorithm loops via a number of single passes through the full training set (*epochs*). In each epoch, we create the random permutation for N observations and then shuffle on $\mathbf{P}$ and associated class labels $\mathbf{y}$ using this permutation. Based on the batch size n, one permutation is divided in to $\left(nb = \frac{N}{n}\right)$ mini-batch. On each mini-batch, using the value of shuffled $\mathbf{P}$ and $\mathbf{y}$, we compute the weights $\boldsymbol{a} = [a_k]$ of the base classifiers. The posterior probabilities $pb_{k,j}$ of observations in the mini-batch is combined via the weights a_k to obtain the pc_j (Eqn. (3)). The gradient of the lost function is updated based on condition in Eqn. (7). Each β_k finally is updated by using Eqn. (11).

The classification process works in a straightforward way. Each testing sample first is predicted by the base classifiers in $\boldsymbol{\mathcal{H}}$ to obtain the posterior probabilities. We then compute the value of $[e_k]$ on the prediction results and compute the weight vector $[a_k]$ by Eqn. (2). The base classifier with $e_k \leq \beta_k$ will not be attended in the final ensemble for the testing sample, and therefore contribute nothing to the final aggregation. The other base classifiers will join to the final ensemble with their contribution $(e_k - \beta_k)$. The class label is assigned by returning the class label associated with the maximum of the combination vector (3).

4 Experimental studies

4.1 Experimental setup

Eighteen real world and synthetic datasets are used in the experiment. For the six real-world datasets (Chess-krvk, DownJones-1985-2003, Electricity, Letter, Penbased Skin_NonSkin) we collected a number of datasets from the UCI[5] and OPENML[6] data sources. For the synthetic datasets, we used MOA library[7] to generate the data. The detailed information of the datasets is summarized in Table 1.

We applied the proposed method to a heterogeneous ensemble system, generated by using 3 learning algorithms named Linear Discriminant Analysis (denoted by LDA), Naïve Bayes, and kNearest Neighbors (where the value of k was set to 5, denoted as kNN5). The proposed method in this case was denoted by Proposed Method$_3$. For the mini-batch approach in the mini-batch gradient descent method, we initialized learning rate $\eta = 0.001$, the credibility threshold $\boldsymbol{\beta} = [\beta_0] = \mathbf{0}$.

We performed extensive comparative studies using a number of existing algorithms as benchmarks: two homogeneous ensemble method named AdaBoost [5] and RUSBoost [14] with 100 classifiers. We also compared with Sum Rule [13] in a heterogeneous ensemble where the set of learning algorithms is similar to our method. For the ensemble pruning methods, we selected two high-performance

[5] http://archive.ics.uci.edu/ml/datasets.html
[6] https://www.openml.org
[7] https://moa.cms.waikato.ac.nz

Table 1. Information of experimental datasets

#	Datasets	# of samples	# of dimensions	# of class labels
1	Agrawal	1000000	9	2
2	AssetNegotiation-F2	1000000	5	2
3	AssetNegotiation-F3	1000000	5	2
4	AssetNegotiation-F4	1000000	5	2
5	BNG-bridge-v1	1000000	12	6
6	BNG_zoo	1000000	17	7
7	Chess-krvk	28056	6	18
8	DowJones-1985-2003	138166	8	30
9	Electricity-normalized	45312	8	2
10	Hyperplane	1000000	10	2
11	Letter	20000	16	26
12	Penbased	10992	16	10
13	RandomTree	1000000	10	2
14	RBF	1000000	50	4
15	Sine	1000000	4	2
16	Skin_NonSkin	245057	3	2
17	Stagger	1000000	3	2
18	Waveform	1000000	21	3

dynamic ensemble selection methods, namely KNORA Union and KNORA Eliminate (denoted by KNORA-U and KNORA-E) [7] as the benchmark algorithms. The number of nearest neighbors in these dynamic methods was set to 7 [3]. For all methods, we performed the same experimental procedure i.e. run 10-fold cross validation 3 times to obtain 30 test results for each dataset (presented in Table 2). The non-parametric two-tailed Wilcoxon signed-rank test [10] was used to compare the experimental results of the proposed method and a benchmark algorithm on a particular dataset in which p-value < 0.05 deems as difference in experimental results is significant.

4.2 The influence of parameters

In this study, we used the mini-batch approach to compute the gradient in updating the credibility threshold through iterations. We examined the influence of the batch size and number of epochs on the performance of the proposed method. Figure 1 presents the relationship between classification error rate and values of batch size $n \in \{1, 16, 32, 64, 128, 256, 521, 1024, 2048, 4096, 8192, All\}$ where *All* means all the number of training data were used as a single batch. Clearly, the line graphs show a common upward trend with different slopes on all datasets as increasing value of batch size can downgrade performance of proposed method. On some datasets like Hyperplane, RBF, Letter and Penbased, the classification error rate increases sharply with the increase of n. Meanwhile, on other datasets such as Agrawal, Electricity-normalized, and BNG-bridge-v1, classification error rates only increase slightly with the increase of the number of batch size. It is also noted that the cost of training is more expensive with smaller value of n. In practice, depending on the training resource and expected

Table 2. Classification error rates of benchmark algorithms and proposed method (using 3 learning algorithms)

#	KNORA-U		KNORA-E		AdaBoost		Sum Rule		RUSBoost		Proposed Method$_3$	
	Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance
1	0.3293▲(4.5)	3.65E-08	0.3440▲(6)	2.53E-07	0.0494▼(1)	3.36E-07	0.3293▲(4.5)	3.04E-08	0.0545▼(2)	3.75E-07	0.3280 (3)	2.50E-11
2	0.0717▲(5)	9.01E-07	0.0558▲(4)	5.55E-07	0.0511▼(1)	4.08E-07	0.1042▲(6)	7.53E-07	0.0553▲(3)	4.02E-07	0.0519 (2)	3.43E-07
3	0.0656▲(5)	7.19E-07	0.0560▲(3)	6.26E-07	0.0531▲(2)	1.01E-06	0.0905▲(6)	7.78E-07	0.0589▲(4)	5.16E-07	0.0525 (1)	3.48E-07
4	0.0740▲(4)	4.34E-07	0.0608▲(3)	2.99E-06	0.0529▼(1)	2.94E-07	0.1139▲(6)	7.08E-07	0.0838▲(5)	5.17E-07	0.0542 (2)	2.52E-06
5	0.3117▲(5)	1.17E-06	0.3017▲(3)	1.51E-06	0.2762▼(1)	1.42E-06	0.3100▲(4)	9.96E-07	0.3376▲(6)	1.42E-06	0.2869 (2)	1.47E-06
6	0.0695▲(4.5)	4.83E-07	0.0585▲(2)	5.55E-07	0.0603▲(3)	6.44E-07	0.0695▲(4.5)	4.34E-07	0.1155▲(6)	1.70E-06	0.0522 (1)	2.50E-07
7	0.3242▲(4)	6.93E-05	0.2955▲(2)	6.11E-05	0.6705▲(5)	3.41E-05	0.3031▲(3)	4.59E-05	0.7433▲(6)	8.09E-05	0.2650 (1)	5.77E-05
8	0.0055▲(5)	2.60E-07	7.0447E-04▲(3)	4.12E-08	2.0024E-04▲(2)	6.52E-09	0.0043▲(4)	3.15E-07	0.6376▲(6)	5.46E-09	0.0000 (1)	0.00E+00
9	0.2192▲(5)	4.93E-05	0.2024▲(3)	2.42E-05	0.1561▼(1)	3.96E-05	0.2126▲(4)	6.07E-05	0.2297▲(6)	9.51E-05	0.1893 (2)	3.22E-05
10	0.0281▲(5)	2.36E-07	0.0128▲(2)	9.55E-08	0.0279▲(4)	3.18E-07	0.0276▲(3)	2.40E-07	0.2542▲(6)	9.62E-06	0.0049 (1)	4.35E-08
11	0.1089▲(3)	5.00E-05	0.0617▲(2)	3.36E-05	0.3460▲(5)	1.15E-04	0.1395▲(4)	6.18E-05	0.7147▲(6)	2.27E-04	0.0507 (1)	3.33E-05
12	0.0435▲(4)	2.58E-05	0.0097▼(1)	1.10E-05	0.0424▲(3)	3.90E-05	0.0895▲(5)	5.46E-05	0.2966▲(6)	9.54E-05	0.0122 (2)	1.41E-05
13	0.1248▲(5)	6.18E-07	0.1179▲(3)	5.43E-07	0.0793▼(1)	1.16E-05	0.1236▲(4)	9.39E-07	0.2529▲(6)	8.36E-06	0.1053 (2)	5.90E-07
14	0.0037▲(3)	4.52E-08	8.7833E-04▲(2)	8.41E-09	0.0475▲(5)	3.01E-02	0.0257▲(4)	1.85E-07	0.1682▲(6)	2.08E-06	2.6667E-06 (1)	1.96E-11
15	0.0119▲(5)	1.40E-07	0.0099▲(3)	6.70E-08	0.0026▼(1)	3.72E-08	0.0115▲(4)	1.05E-07	0.0423▲(6)	7.39E-07	0.0088 (2)	9.35E-08
16	9.0047E-04▲(4)	4.54E-08	5.1145E-04▲(2)	1.64E-08	6.2979E-04▲(3)	2.73E-08	0.0412▲(6)	1.22E-06	0.0265▲(5)	2.05E-06	4.7608E-04 (1)	1.74E-08
17	0.0000 (2.5)	0.00E+00	0.0000 (2.5)	0.00E+00	0.1116▲(5.5)	9.00E-12	0.0000 (2.5)	0.00E+00	0.1116▲(5.5)	9.00E-12	0.0000 (2.5)	0.00E+00
18	0.1591▲(3)	1.06E-06	0.1518▲(2)	1.67E-06	0.1610▲(5)	1.08E-06	0.1594▲(4)	1.18E-06	0.2622▲(6)	1.41E-05	0.1364 (1)	1.38E-06
	Win: 17; Equal:1; Lost:0		Win: 16; Equal: 1; Lost: 1		Win: 11; Equal:0; Lost: 7		Win: 17; Equal:1; Lost: 0		Win:17; Equal:0; Lost:1			
Rk	4.25		2.69		2.75		4.36		5.36		1.58	

** ▲ and ▼ indicate that proposed method is better or worse than benchmark algorithm; (.) indicates the rank of method on the dataset*

Rk *indicates average ranking of each method*

performance score, we can choose a suitable value for the batch size parameter. In the next section, we used the batch size $n = 16$ in comparison to the baselines.

Figure 2 presents the classification error rates of the proposed method on experimental datasets where *epochs* parameter was set to 5 and 50. In general, although increasing the number of epochs can improve the ensemble performance, differences in two performance scores on these datasets are not significant. One datasets like Argawal and AssetNegotiation-F2, F3 and F4, the classification error rates only change slightly or remain unchanged with the change of the number of epochs. Only on three datasets Letter, Penbased, and Hyperplane, the differences in classification error rate between two cases are remarkable. In practice, in case of limited resource available, we can choose a small number of epochs for the training process.

4.3 Comparing to the baselines

Table 2 presents the classification error rates of benchmark algorithms and proposed method in case of using 3 learning algorithms. The following observations can be made:

– Proposed Method$_3$ achieves lowest average rank among all methods (rank value 1.58). On 18 experimental datasets, Proposed Method$_3$ ranks first in 9 cases (50%) and ranks second in 7 cases (38.89%). Our method only performs poorly on Agrawal dataset in which it ranks third.

- Proposed Method_3 is better than two DES methods. Comparing to KNORA-U, our method wins in 17 cases and does not lose on any case. Proposed Method_3 underperforms KNORA-E on only Penbased datasets (0.0097 vs. 0.0122) while wins this method on 16 datasets.
- Proposed Method_3 is significantly better than Sum Rule on 17 datasets. One Stagger dataset where two methods performs equally, both obtain 100% of classification accuracy.
- The performance of Proposed Method_3 is better than RUSBoost in 17 cases. Although RUSBoost is special designed for imbalanced data, it significantly underperforms Proposed Method_3 on some imbalanced datasets in our experiments such as Chess-krvk and Skin_NonSkin.
- AdaBoost is a high performance ensemble in our experiment in which our method only wins in 11 cases and loses in 7 cases. However, Proposed Method_3 is only significantly worse than AdaBoost on Agrawal datasets while our method significantly outperforms on at least 6 datasets Chess-krvk, Hyperplance, Letter, RBF, Stagger, and DowJones-1958-2003.
- The variances of classification error rate of experimental methods on some datasets, especially on synthetic ones such as RBF and Sine, are very small. That means the differences in the classification error rates among 30 results in the test procedure on these datasets are not significant.

To summarize, Proposed Method_3 achieves better performance than two Boosting methods, two DES methods and one simple non-weighted combining method. Proposed Method_3 significantly outperforms Sum Rule in all cases which demonstrates the advantage of ensemble weighting technique compared to the simple combining methods.

5 Conclusions and future work

We have presented a novel weighted ensemble method for ensemble systems which considers the confidence in the prediction of each classifier. Based on the observation that each classifier provides a different level of confidence in its prediction for each sample, we propose to associate a credibility threshold with each classifier. The confidence in the prediction of each classifier on a sample is compared to the credibility threshold to determine whether the classifier's output should be included in the aggregation. To show the contribution of a classifier in the selected ensemble, we use the difference between the confidence in the prediction and the credibility threshold. This allows us to integrate both the static and dynamic approaches in the proposed method i.e. learning the credibility threshold on the training data by minimizing the entropy loss function and assigning a particular weight associated with each classifier for each test sample. The experiments on diverse data sources show the advantage of the proposed method compared to the benchmark algorithms.

In the future we plan to 1) analyse the convergence of the proposed method, 2) expand the proposed method to handle data stream with concept drift.

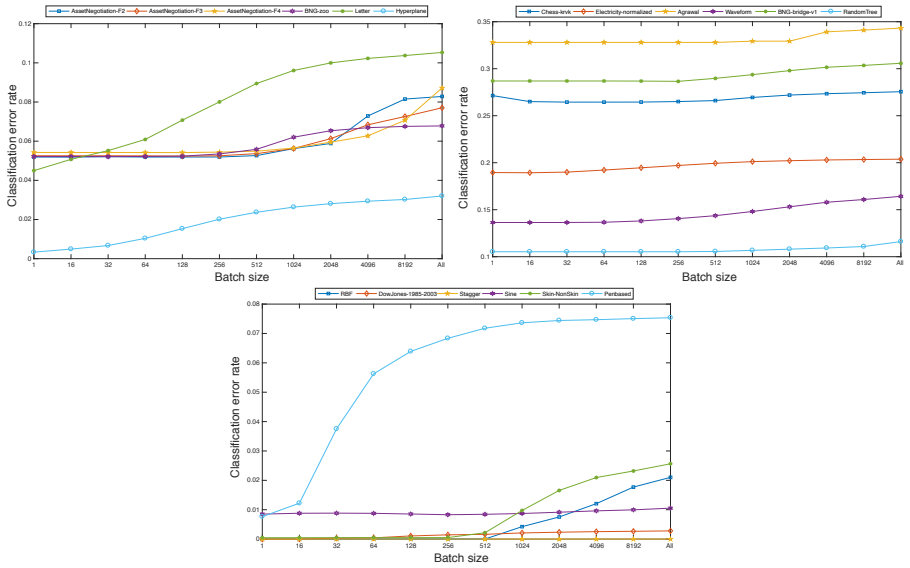

Fig. 1. The influence of number of batch size to classification error rate of proposed method

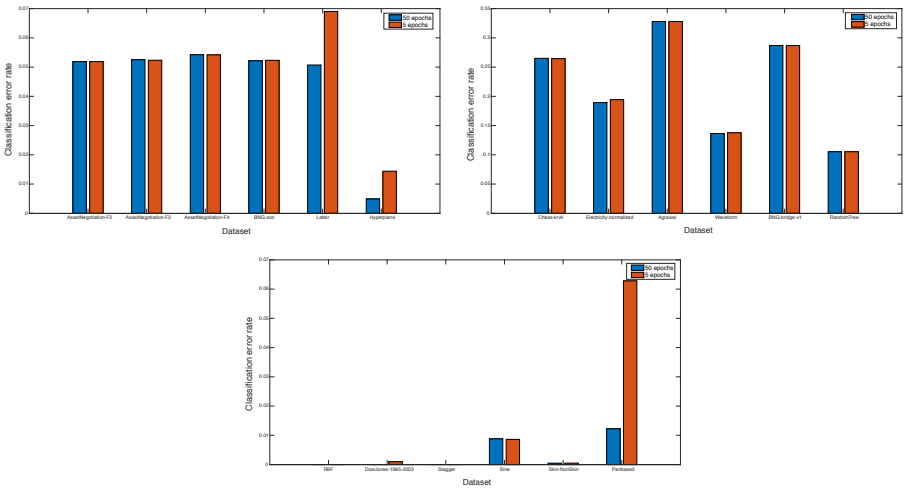

Fig. 2. The performance comparison between proposed method using 5 and 50 epochs

References

1. Bakker, B., Heskes, T.: Clustering ensembles of neural network models. Neural networks **16**(2), 261–269 (2003)
2. Chen, H., Tiňo, P., Yao, X.: Predictive ensemble pruning by expectation propagation. IEEE Transactions on Knowledge & Data Engineering **21**(7), 999–1013 (2009)
3. Dang, M.T., Luong, A.V., Vu, T.T., Nguyen, Q.V.H., Nguyen, T.T., Stantic, B.: An ensemble system with random projection and dynamic ensemble selection. In: Asian Conference on Intelligent Information and Database Systems. pp. 576–586. Springer (2018)
4. Demiriz, A., Bennett, K.P., Shawe-Taylor, J.: Linear programming boosting via column generation. Machine Learning **46**(1-3), 225–254 (2002)
5. Freund, Y., Schapire, R.E., et al.: Experiments with a new boosting algorithm. In: icml. vol. 96, pp. 148–156. Citeseer (1996)
6. Kim, K.J., Cho, S.B.: An evolutionary algorithm approach to optimal ensemble classifiers for dna microarray data analysis. IEEE Transactions on Evolutionary Computation **12**(3), 377–388 (2008)
7. Ko, A.H., Sabourin, R., Britto Jr, A.S.: From dynamic classifier selection to dynamic ensemble selection. Pattern recognition **41**(5), 1718–1731 (2008)
8. Kuncheva, L.I., Bezdek, J.C., Duin, R.P.: Decision templates for multiple classifier fusion: an experimental comparison. Pattern recognition **34**(2), 299–314 (2001)
9. Margineantu, D.D., Dietterich, T.G.: Pruning adaptive boosting. In: ICML. vol. 97, pp. 211–218. Citeseer (1997)
10. Nguyen, T.T., Dang, M.T., Liew, A.W., Bezdek, J.C.: A weighted multiple classifier framework based on random projection. Information Sciences **490**, 36–58 (2019)
11. Nguyen, T.T., Nguyen, M.P., Pham, X.C., Liew, A.W.C., Pedrycz, W.: Combining heterogeneous classifiers via granular prototypes. Applied Soft Computing **73**, 795–815 (2018)
12. Nguyen, T.T., Nguyen, T.T.T., Pham, X.C., Liew, A.W.C.: A novel combining classifier method based on variational inference. Pattern Recognition **49**, 198–212 (2016)
13. Nguyen, T.T., Pham, X.C., Liew, A.W.C., Pedrycz, W.: Aggregation of classifiers: A justifiable information granularity approach. IEEE transactions on cybernetics **49**(6), 2168–2177 (2018)
14. Seiffert, C., Khoshgoftaar, T.M., Van Hulse, J., Napolitano, A.: Rusboost: Improving classification performance when training data is skewed. In: 2008 19th International Conference on Pattern Recognition. pp. 1–4. IEEE (2008)
15. Ting, K.M., Witten, I.H.: Issues in stacked generalization. Journal of artificial intelligence research **10**, 271–289 (1999)
16. Woloszynski, T., Kurzynski, M., Podsiadlo, P., Stachowiak, G.W.: A measure of competence based on random classification for dynamic ensemble selection. Information Fusion **13**(3), 207–213 (2012)
17. Wu, O.: Classifier ensemble by exploring supplementary ordering information. IEEE Transactions on Knowledge and Data Engineering **30**(11), 2065–2077 (2018)
18. Yijing, L., Haixiang, G., Xiao, L., Yanan, L., Jinling, L.: Adapted ensemble classification algorithm based on multiple classifier system and feature selection for classifying multi-class imbalanced data. Knowledge-Based Systems **94**, 88–104 (2016)
19. Zhang, Y., Burer, S., Street, W.N.: Ensemble pruning via semi-definite programming. Journal of Machine Learning Research **7**(Jul), 1315–1338 (2006)