Applied Soft Computing 47 (2016) 262–270

Contents lists available at ScienceDirect

Applied Soft Computing

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

Self-correcting ensemble using a latent consensus model

Namhyoung Kim[a], Youngdoo Son[b], Youngjo Lee[c], Jaewook Lee[b,*]

[a] Department of Applied Statistics, Gachon University, 1342 Seongnamdaero, Sujeong-gu, Seongnam-si, Gyeonggi 461-701, South Korea
[b] Department of Industrial Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 151-744, South Korea
[c] Department of Statistics, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 151-744, South Korea

ARTICLE INFO

Article history:
Received 13 March 2015
Received in revised form 28 August 2015
Accepted 26 April 2016
Available online 7 June 2016

Keywords:
Ensemble
Latent consensus model
Self-correction
Decision tree
Artificial neural network

ABSTRACT

Ensemble is a widely used technique to improve the predictive performance of a learning method by using several competing expert systems. In this study, we propose a new ensemble combination scheme using a latent consensus function that relates each predictor to the other. The proposed method is designed to adapt and self-correct weights even when a number of expert systems malfunction and become corrupted. To compare the performance of the proposed method with existing methods, experiments are performed on simulated data with corrupted outputs as well as on real-world data sets. Results show that the proposed method is effective and it improves the predictive performance even when a number of individual classifiers are malfunctioning.

1. Introduction

In the real world, people obtain professional advice from several experts before finally deciding on significant matters such as investing financially, seeking treatment for a disease, and buying products. Combining the opinions of several different specialists is natural. In artificial intelligence and data mining, ensemble systems are techniques that combine multiple experts opinions (e.g., classifier) to obtain better predictive performance than using a single opinion. Ensemble methods are also known as multiple classifier systems, committee of classifiers, or mixture of experts.

Using the ensemble technique has several advantages. First, ensemble learning improves accuracy and robustness better than a single model does. Each classifier in an ensemble may capture the big picture of the problem and the ensemble technique may obtain more sensitive results by making up for each weak learner. Combining diverse, independent multiple predictors reduces variance and bias because of less dependence on the outliers of the training sets, which increases functional flexibility. This combination may also reduce the total error when each error occurs in different directions. The ensemble technique reduces the risk of selecting a poor classifier by averaging the outputs.

Second, the ensemble technique is suitable for cases when the size of the data set to be analyzed is extremely large or extremely small. With the rapid development of hardware and software technologies, the size of data increases at a fast rate. Applying existing data mining techniques to large data is difficult [20]. An extremely small size of an available data set is problematic. Ensemble techniques can be useful when the size of the data set is extremely small because these techniques reproduce the training data set by using a resampling technique.

Third, ensemble systems provide the means to solve difficult problems. The complex decision boundary that divides data sets cannot be learned by using a simple linear model. However, an ensemble can learn the complex boundary or function by appropriately combining simple classifiers.

Lastly, ensemble systems can be applied to data sets with different data types, such as medical image data from different sources such as MRI, FDG, or PIB, as well as numerical data with text information. These data sets are more informative but difficult to analyze using a single model. In such cases, we can train different models for each data type and then collect them to create a final model.

Most ensemble learning systems consist of two phases. The first phase is the building model process wherein each classifier is diversified, and the second phase combines the outputs in specific ways. A number of different algorithms for the first phase of ensemble learning have been suggested. When each classifier that forms an ensemble system is highly diverse, the effect of ensemble systems can be maximized. A number of algorithms can achieve diversity by using resampling techniques or different training parameters for different models.

* Corresponding author.
E-mail addresses: nhkim@gachon.ac.kr (N. Kim), hand02@snu.ac.kr (Y. Son), youngjo@snu.ac.kr (Y. Lee), jaewook@snu.ac.kr (J. Lee).

http://dx.doi.org/10.1016/j.asoc.2016.04.037
1568-4946/© 2016 Elsevier B.V. All rights reserved.

Popular ensemble methods are bagging, random forest, boosting, and adaBoost. Several approaches for the second phase of ensemble combination are also available. Existing methods use majority voting, simple averaging of outputs, or the weighted sum of the votes of the weak learners. The weights are different for each component classifier of the ensemble system. The strategies used to calculate the weights are grouped as trainable and non-trainable methods. When weights depend on the performance or results of predictors, we do not have to train the weights. However, trainable methods require another training algorithm for computing the weights.

In this study, we propose a new ensemble combination scheme that uses a latent consensus function with related predictors. The proposed method aims to reveal better predictive power and functional reliability compared with the existing methods. We intend to show through simulation that the proposed method works effectively in situations when the component predictors in the ensemble fail or does not work well in predicting an output for a system malfunction.

The paper is organized as follows. In Section 2, we summarize related literature and algorithms. In Section 3, we describe the proposed method. Section 4 shows simulation results for predictor malfunction. We present the results of the empirical analysis and comparison of the proposed method with other ensemble methods in Section 5. We conclude the paper in Section 6.

2. Literature review

The concept of ensemble systems was proposed in [6], which discussed the division of feature space. Since then, several algorithms have been proposed [20,12,26,17,18,24] and applied to diverse fields [4,5]. Most of the contemporary ensemble methods develop an ensemble system based on the following equation:

$$E = \sum_{i=1}^{m} w_i c_i, \tag{1}$$

where E is an ensemble, c_i is an individual base predictor, and w_i is the weight for ith predictor. Fig. 1 shows that the predictions of the individual models c_i, $i = 1, 2, ..., m$ are combined using w_i. These weights can vary depending on combination scheme. As mentioned, ensemble systems obtain diversity by using a resampling technique, different training parameters, and different features. According to a combination scheme, ensembles are divided into two main classes, namely, ensembles combined by learning and ensembles combined by consensus.

Bagging is the simplest algorithm used to construct an ensemble [1], and is also known as bootstrap aggregating. Bagging creates individual predictors by training randomly selected training set. Each training set is generated by randomly with replacement, n examples. In bagging, the ensemble is formed by majority voting. Bagging does not consider the performance of each predictor. It

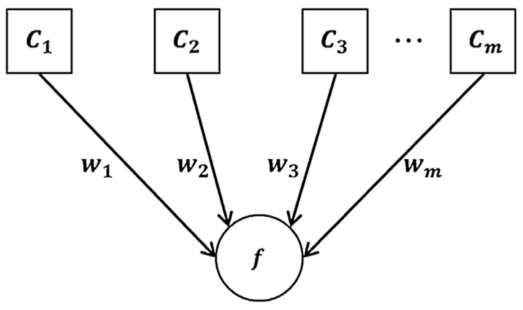


Fig. 1. Ensemble of classifiers.

averages the predictions of individual models in an unsupervised scheme. In bagging, the weights are uniform, i.e., the ensemble prediction is given by

$$E = \frac{1}{m}\sum_{i=1}^{m} c_i. \tag{2}$$

In classification, majority voting is used to predict the class of data. Thus, bagging is a simple but powerful method. A random forest is an ensemble technique for decision trees algorithm that combines the concept of bagging and the random selection of features [2].

Boosting combines multiple base predictors by learning. It uses the information of the performance of previous predictors to select the most informative data as a training data set. This kind of ensemble is also known as supervised scheme. Boosting algorithms learn iteratively weak classifiers using a training set selected according to the previous classified results, and then combine these classifiers with different weights to create an ensemble [10]. The weights are determined by the performance of the weak learners. Boosting results sometimes show poor performance because of overfitting the training set. Boosting method for updating the probabilities may be overemphasizing noisy data. Therefore, if noise exists in the data, boosting performance may perform poorly. Other ensemble systems proposed include stacked generalization and mixture of experts model. These systems are similar in such a way that both have another learning phase for computing the weights. In stacked generalization, the outputs of each classifier are used as inputs to learn the relationship between the ensemble outputs and actual classes [27]. Similarly, the mixture of experts model also uses a second level classifier, but the inputs are the training data instances rather than the outputs of classifiers [17].

In this study, we assume that the classifier outputs are already given. Thus, the diversity of the classifiers is not our focus. We propose a new trainable ensemble combination method for computing weights. In the following section, we describe our proposed method.

3. Proposed method

In this paper, we propose an ensemble combination scheme. In contrast to the conventional ensemble combination scheme that follows (1) (or Fig. 1), the proposed method extracts a latent consensus from opinions of experts. To be specific, we build a latent consensus model where each expert predictor c_i with $\mathbb{E}[c_i(\mathbf{x})] = \mu_i$ reflects the so-called latent consensus function f as follows:

$$c_i(\mathbf{x}) - \mu_i = \lambda_i(f(\mathbf{x}) - \mu_f) + \theta_i\delta_i(\mathbf{x}), \quad i = 1, 2, \ldots, m \tag{3}$$

where λ_i and θ_i are constants, and δ_i is a specific noise term with zero mean and unit variance. δ_i is assumed to be uncorrelated with f and δ_j's for $j \neq i$, and $f(\cdot)$ is assumed to have mean μ_f and variance σ^2. The conceptual model which extends and generalizes a latent one-factor model is described in Fig. 2. Given a data set $\mathcal{D}$, it will be shown in this section that our final ensemble combination model of experts c_i can share the behavior of (1) but approximately as follows.

$$E = \hat{f}(\mathbf{x}) \simeq \sum_{i=1}^{m} w_i^{\mathcal{D}} c_i(\mathbf{x}), \quad i = 1, 2, \ldots, m \tag{4}$$

where $w_i^{\mathcal{D}}$ is a weight to be learned from data $\mathcal{D}$.

We next describe how to calculate $w_i^{\mathcal{D}}$ as well as λ_i measuring a relative consensus of each individual experts. Eq. (3) can be equivalently written in the vector form

$$\mathbf{c}(\mathbf{x}) - \boldsymbol{\mu} = (f(\mathbf{x}) - \mu_f)\boldsymbol{\lambda} + \mathbf{D}_{\boldsymbol{\theta}}\boldsymbol{\delta}(\mathbf{x}), \quad i = 1, 2, \ldots, m. \tag{5}$$

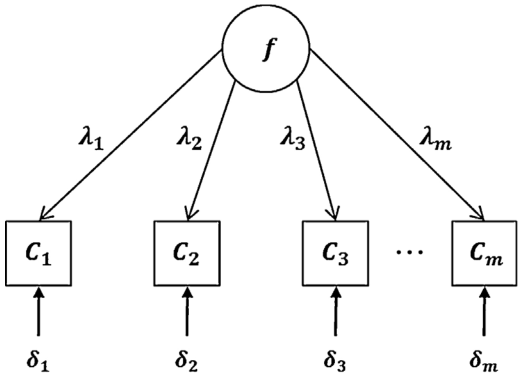


Fig. 2. Concept of the proposed method.

where $\boldsymbol{\lambda}=(\lambda_1,\ldots,\lambda_m)^\top$, $\boldsymbol{\theta}=(\theta_1,\ldots,\theta_m)^\top$, $\boldsymbol{\delta}=(\delta_1,\ldots,\delta_m)^\top$, and $\mathbf{D}_\theta$ is the diagonal matrix with θ_i in the ith diagonal position. Then we observe that the slope coefficient λ_i represents the scaled covariance of the c_i with the real function f, i.e.

$$\begin{aligned}\lambda_i &= \mathrm{cov}[c_i(\mathbf{x}),f(\mathbf{x})]/\sigma^2\\ &= \mathbb{E}[(f(\mathbf{x})-\mathbb{E}[f(\mathbf{x})])(c_i(\mathbf{x})-\mathbb{E}[c_i(\mathbf{x})])]/\sigma^2,\quad i=1,2,\ldots,m.\end{aligned}\tag{6}$$

Moreover, we have

$$\begin{aligned}\boldsymbol{\Sigma} &:=\mathbb{E}[(\mathbf{c}(\mathbf{x})-\boldsymbol{\mu})(\mathbf{c}(\mathbf{x})-\boldsymbol{\mu})^\top]\\ &=\mathbb{E}[(f(\mathbf{x})-\mu_f)^2]\boldsymbol{\lambda}\boldsymbol{\lambda}^\top+\mathbf{D}_{\boldsymbol{\theta}}\mathbb{E}[\boldsymbol{\delta}\boldsymbol{\delta}^\top]\mathbf{D}_{\boldsymbol{\theta}}=\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top+\mathbf{D}_{\boldsymbol{\theta}^2},\end{aligned}\tag{7}$$

where $\boldsymbol{\theta}^2=(\theta_1^2,\ldots,\theta_m^2)^\top$.

Now to estimate the unknown parameters from data $\mathcal{D}=\{\mathbf{x}_1,\ldots,\mathbf{x}_n\}$, we set $f^{(j)}=f(\mathbf{x}_j)$, $c_i^{(j)}=c_i(\mathbf{x}_j)$, and $\delta_i^{(j)}=\delta_i(\mathbf{x}_j)$, we can write the above equation as

$$\begin{aligned}&[c_i^{(1)}-\mu_i,c_i^{(2)}-\mu_i,\ldots,c_i^{(n)}-\mu_i]\\ &\quad=\lambda_i[f^{(1)}-\mu_f,f^{(2)}-\mu_f,\ldots,f^{(n)}-\mu_f]\\ &\qquad+\theta_i[\delta_i^{(1)},\delta_i^{(2)},\ldots,\delta_i^{(n)}],\quad i=1,2,\ldots,m,\end{aligned}$$

or equivalently in vector form

$$\mathbf{c}_i-\mu_i\mathbf{1}=\lambda_i(\mathbf{f}-\mu_f\mathbf{1})+\theta_i\boldsymbol{\delta}_i,\quad i=1,2,\ldots,m,$$

where $\mathbf{c}_i=(c_i^{(1)},c_i^{(2)},\ldots,c_i^{(n)})^\top$, $\mathbf{f}=(f^{(1)},f^{(2)},\ldots,f^{(n)})^\top$, and $\boldsymbol{\delta}=(\delta_i^{(1)},\delta_i^{(2)},\ldots,\delta_i^{(n)})^\top$. We can then rewrite the above equation in matrix form as following.

$$\mathbf{C}-\mathbf{1}\boldsymbol{\mu}^\top=(\mathbf{f}-\mu_f\mathbf{1})\boldsymbol{\lambda}^\top+\boldsymbol{\Delta}\mathbf{D}_{\boldsymbol{\theta}},\tag{8}$$

where

$$\mathbf{C}=\begin{bmatrix}c_1^{(1)} & c_2^{(1)} & \cdots & c_m^{(1)}\\ c_1^{(2)} & c_2^{(2)} & \cdots & c_m^{(2)}\\ \vdots & & & \\ c_1^{(n)} & c_2^{(n)} & \cdots & c_m^{(n)}\end{bmatrix}\in\Re^{n\times m},\quad \mathbf{f}=\begin{bmatrix}f^{(1)}\\ f^{(2)}\\ \vdots\\ f^{(n)}\end{bmatrix}\in\Re^n,\quad \boldsymbol{\lambda}=\begin{bmatrix}\lambda_1\\ \lambda_2\\ \vdots\\ \lambda_m\end{bmatrix}\in\Re^m,\quad \boldsymbol{\Delta}=\begin{bmatrix}\delta_1^{(1)} & \delta_2^{(1)} & \cdots & \delta_m^{(1)}\\ \delta_1^{(2)} & \delta_2^{(2)} & \cdots & \delta_m^{(2)}\\ \vdots & & & \\ \delta_1^{(n)} & \delta_2^{(n)} & \cdots & \delta_m^{(n)}\end{bmatrix}\in\Re^{n\times m},\quad \boldsymbol{\mu}=\begin{bmatrix}\mu_1\\ \mu_2\\ \vdots\\ \mu_m\end{bmatrix}\in\Re^m.$$

To find the appropriate parameters, we follow and extend the approach of [14] and observe that the matrix of partial correlations for the specific noise term $\boldsymbol{\delta}(\mathbf{x})$ given by

$$\begin{aligned}\mathbb{E}[\boldsymbol{\delta}(\mathbf{x})\boldsymbol{\delta}(\mathbf{x})^\top] &= \mathbf{D}_{\boldsymbol{\theta}}^{-1}\mathbb{E}[(\mathbf{c}(\mathbf{x})-\boldsymbol{\mu}-(f(\mathbf{x})\\ &\quad-\mu_f)\boldsymbol{\lambda})(\mathbf{c}(\mathbf{x})-\boldsymbol{\mu}-(f(\mathbf{x})-\mu_f)\boldsymbol{\lambda})^\top]\mathbf{D}_{\boldsymbol{\theta}}^{-1}\end{aligned}$$

should have zeroes in its off-diagonal elements by the uncorrelated properties of different δ_i's. From our assumptions on $\mathbf{f}$ and $\boldsymbol{\Delta}$, i.e.

$$\frac{1}{(n-1)}(\mathbf{f}-\mu_f\mathbf{1})^\top(\mathbf{f}-\mu_f\mathbf{1})\approx\sigma^2,\qquad \frac{1}{(n-1)}(\mathbf{f}-\mu_f\mathbf{1})^\top\boldsymbol{\Delta}\approx 0$$

the sample covariance can be approximated as

$$\begin{aligned}\mathbf{S} &= \frac{1}{n-1}(\mathbf{C}-\mathbf{1}\boldsymbol{\mu}^\top)^\top(\mathbf{C}-\mathbf{1}\boldsymbol{\mu}^\top)\\ &= \frac{1}{n-1}((\mathbf{f}-\mu_f\mathbf{1})\boldsymbol{\lambda}^\top+\boldsymbol{\Delta}\mathbf{D}_{\boldsymbol{\theta}})^\top((\mathbf{f}-\mu_f\mathbf{1})\boldsymbol{\lambda}^\top+\boldsymbol{\Delta}\mathbf{D}_{\boldsymbol{\theta}})\\ &= \frac{1}{n-1}(\boldsymbol{\lambda}(\mathbf{f}-\mu_f\mathbf{1})^\top(\mathbf{f}-\mu_f\mathbf{1})\boldsymbol{\lambda}^\top+\mathbf{D}_{\boldsymbol{\theta}}[\boldsymbol{\Delta}^\top(\mathbf{f}-\mu_f\mathbf{1})\boldsymbol{\lambda}^\top\\ &\quad+\boldsymbol{\Delta}(\mathbf{f}-\mu_f\mathbf{1})^\top\boldsymbol{\Delta}+\boldsymbol{\Delta}^\top\boldsymbol{\Delta}]\mathbf{D}_{\boldsymbol{\theta}})\\ &\approx\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top+\frac{1}{n-1}\mathbf{D}_{\boldsymbol{\theta}}\boldsymbol{\Delta}^\top\boldsymbol{\Delta}\mathbf{D}_{\boldsymbol{\theta}}.\end{aligned}$$

Therefore the sample counterparts of the partial correlations is given by

$$\frac{1}{n-1}\boldsymbol{\Delta}^\top\boldsymbol{\Delta}=\mathbf{D}_{\boldsymbol{\theta}}^{-1}[\mathbf{S}-\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top]\mathbf{D}_{\boldsymbol{\theta}}^{-1}$$

and its off-diagonal elements should be as small as possible.

Notice that if we let $\nu_1,\nu_2,\ldots,\nu_N$ be eigenvalues of a correlation matrix $\mathbf{A}\in\Re^{N\times N}$, then $\mathrm{tr}(\mathbf{A})=N=\sum_{i=1}^N\nu_i$, so we have

$$\ln\det\mathbf{A}=\ln\prod_{i=1}^N\nu_i=N\cdot\frac{1}{N}\sum_{i=1}^N\ln\nu_i\le N\cdot\ln(\frac{1}{N}\sum_{i=1}^N\nu_i)=N\cdot\ln(1)=0$$

by the negative convexity of ln functions and Jensen's inequality. Therefore, we have $\det\mathbf{A}\le 1$ for any correlation matrix $\mathbf{A}$. Using this fact, since the off-diagonal elements of the matrix $\frac{1}{n-1}\boldsymbol{\Delta}^\top\boldsymbol{\Delta}$ shrink to zeroes, their determinant approaches its maximum value of one. This motivates us to solve the following problem to estimate the parameters:

$$\max\quad |\mathbf{D}_{\boldsymbol{\theta}}^{-1}[\mathbf{S}-\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top]\mathbf{D}_{\boldsymbol{\theta}}^{-1}|=\frac{|\mathbf{S}-\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top|}{|\mathbf{D}_{\boldsymbol{\theta}^2}|},\tag{9}$$

where $|\mathbf{A}|$ denotes the determinant of a square matrix $\mathbf{A}$.

Next to find the maximum likelihood estimate, we need to differentiate the right hand side of Eq. (9) by the unknown parameters.

$$\frac{\partial}{\partial\lambda_i}\frac{|\mathbf{S}-\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top|}{|\mathbf{D}_{\boldsymbol{\theta}^2}|}=-\frac{1}{|\mathbf{D}_{\boldsymbol{\theta}^2}|^2}\frac{\partial|\mathbf{D}_{\boldsymbol{\theta}^2}|}{\partial\lambda_i}|\mathbf{S}-\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top|+\frac{1}{|\mathbf{D}_{\boldsymbol{\theta}^2}|}\frac{\partial|\mathbf{S}-\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top|}{\partial\lambda_i}\tag{10}$$

Since

$$\mathbf{D}_{\boldsymbol{\theta}^2}=\mathrm{diag}(\sigma_i^2-\sigma^2\lambda_i^2)$$

and

$$|\mathbf{D}_{\boldsymbol{\theta}^2}|=\prod_i(\sigma_i^2-\sigma^2\lambda_i^2),$$

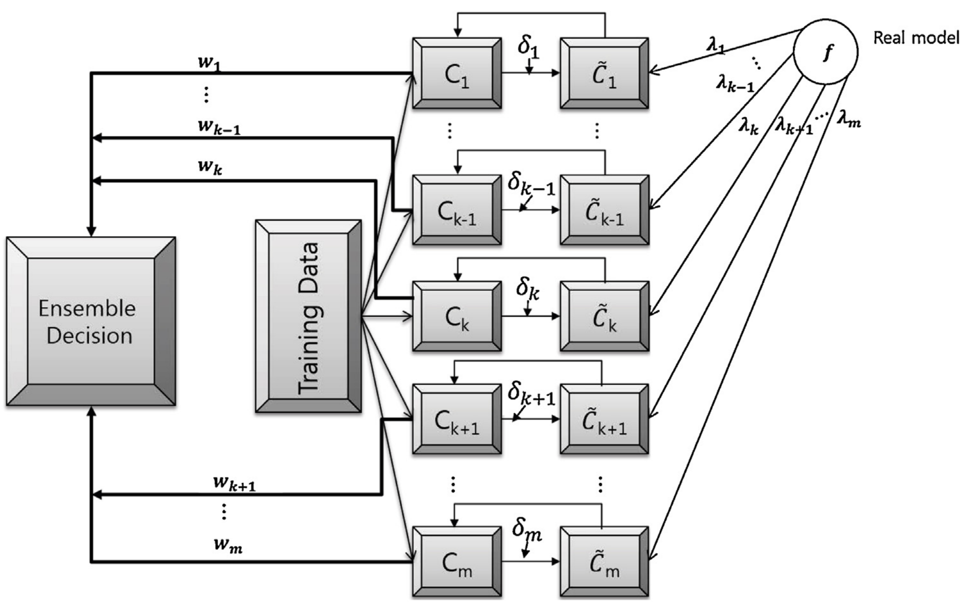


Fig. 3. Proposed method.

where σ_i^2 is the i'th diagonal element of $\boldsymbol{\Sigma}$, we have

$$\frac{\partial|\mathbf{D}_{\theta^2}|}{\partial\lambda_i} = -2\sigma^2\lambda_i\prod_{j\neq i}(\sigma_j^2-\sigma^2\lambda_j^2) = -2\sigma^2\lambda_i\frac{|\mathbf{D}_{\theta^2}|}{\theta_i^2}. \tag{11}$$

Also, we have

$$\begin{aligned}\frac{\partial|\mathbf{S}-\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top|}{\partial\lambda_i} &= \sum_{g=1}^{m}\sum_{h=1}^{m}|\mathbf{S}-\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top|_{gh}\frac{\partial(s_{gh}-\sigma^2\lambda_g\lambda_h)}{\partial\lambda_i}\\ &= \sigma^2\sum_{g=1}^{m}|\mathbf{S}-\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top|_{gi}(-\lambda_g)\\ &\quad+\sigma^2\sum_{h=1}^{m}|\mathbf{S}-\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top|_{ih}(-\lambda_h)\\ &= -2\sigma^2\sum_{g=1}^{m}|\mathbf{S}-\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top|_{gi}\lambda_g,\end{aligned} \tag{12}$$

where s_{gh} is the gth row and hth column element of $\mathbf{S}$ and $|\mathbf{A}|_{ij}$ is the ijth cofactor of the matrix $\mathbf{A}$.

Inserting (11) and (12) into (10), we have

$$\frac{\lambda_i}{\theta_i^2} = \sum_{g=1}^{m}\frac{|\mathbf{S}-\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top|_{gi}}{|\mathbf{S}-\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top|}, \quad \forall i=1,\ldots,m,$$

or, in matrix form,

$$\mathbf{D}_{\theta^2}^{-1}\boldsymbol{\lambda} = (\mathbf{S}-\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top)^{-1}\boldsymbol{\lambda}.$$

After simple manipulation, we get the maximum likelihood solution for $\boldsymbol{\lambda}$:

$$\sigma^2\boldsymbol{\lambda}(\boldsymbol{\lambda}^\top\mathbf{D}_{\theta^2}^{-1}\boldsymbol{\lambda}) = \mathbf{S}\mathbf{D}_{\theta^2}^{-1}\boldsymbol{\lambda}-\boldsymbol{\lambda}. \tag{13}$$

Then, from Eq. (7), we can calculate

$$\mathbf{D}_{\theta^2} = \boldsymbol{\Sigma}-\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top.$$

Eventually, from Eq. (5), we obtain the following predicted model of conditional multivariate normal distribution form assuming $\mu_f=0$ and $\boldsymbol{\mu}=0$:

$$f(\mathbf{x})|\mathbf{c}(\mathbf{x})\sim\mathcal{N}(\boldsymbol{w}^{\boldsymbol{D}^\top}\mathbf{c}, \sigma_{\mathcal{D}}^2) \tag{14}$$

$$\boldsymbol{w}^{\boldsymbol{D}} = \sigma^2\boldsymbol{\lambda}^\top(\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top+\mathbf{D}_{\theta^2})^{-1} \tag{15}$$

$$\sigma_{\mathcal{D}}^2 = \sigma^2-\sigma^4\boldsymbol{\lambda}^\top(\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top+\mathbf{D}_{\theta^2})^{-1}\boldsymbol{\lambda}. \tag{16}$$

Now from Eq. (15), we can derive Eq. (4).

$$E=\hat{f}(\mathbf{x})\simeq\sum_{i=1}^{m}w_i^{\mathcal{D}}c_i(\mathbf{x}), \quad i=1,2,\ldots,m \tag{17}$$

where $w_i^{\mathcal{D}}=\sigma^2\boldsymbol{\lambda}^\top(\sigma^2\boldsymbol{\lambda}\boldsymbol{\lambda}^\top+\mathbf{D}_{\theta^2})^{-1}$ is a weight to be learned from data $\mathcal{D}$. One issue to be discussed is that the solution of Eq. (13) is not unique, and is called a rotation problem. We can select the value of $\boldsymbol{\lambda}$ and $\mathbf{D}_{\theta^2}$ for example by using *Kaiser's varimax rotation* as in [19]. Fig. 3 illustrates the whole process of the proposed method, where the outputs of individual predictors are input of the weight training system. After calculating λ and w, the final ensemble decisions based on a constructed latent consensus function are made.

4. Simulation for predictor malfunction

In certain cases, we may have some corrupted predictor outputs because of malfunction. If these predictors are determined in advance, we may ignore and delete the outputs of these predictors or not use them in the current ensemble methods. This situation may decrease the performance of the ensemble because of fewer available predictors. The proposed method, however, can employ these corrupted predictors by using self-correction to improve its performance. To verify the robustness of the proposed method, we perform an experimental simulation on a number of artificially designed data sets. We begin the simulation with the assumption that the outputs of individual predictors are ready to use because the algorithm that we focus on combines the outputs after generating the predictors. We assume that we have 15 classifiers ($m=15$) for a two-class classification problem with 100 instances ($n=100$) with 50 of Class 1 and 50 of Class 2. The outputs of individual classifiers are generated from Model A. The outputs for a given class are assumed to be continuous value; thus, these outputs can be interpreted as estimates of the posterior probability for that class. The following three cases are considered in the simulation.

- Case I: all $0.8<\lambda_i<1$

Table 1
Experimental simulation using toy data sets: average classification error with standard deviation in parentheses.

	Bagging	Proposed
Case I	0.0004 (0.0020)	0.0004 (0.0020)
Case II	0.0073 (0.0095)	0.0047 (0.0073)
Case III	0.1098 (0.0305)	0.0012 (0.0033)

Table 2
The average of the estimated weights w_i from the proposed method.

	Case I	Case II	Case III
w_1	0.0691	0.0111	−0.2126
w_2	0.0646	0.0115	−0.2381
w_3	0.0644	0.0114	−0.2282
w_4	0.0630	0.0125	−0.2137
w_5	0.0669	0.0105	−0.2127
w_6	0.0668	0.0941	0.2100
w_7	0.0654	0.0926	0.2069
w_8	0.0635	0.0895	0.1983
w_9	0.0671	0.0945	0.2072
w_{10}	0.0681	0.0960	0.2157
w_{11}	0.0669	0.0928	0.2046
w_{12}	0.0699	0.0979	0.2194
w_{13}	0.0656	0.0933	0.2080
w_{14}	0.0665	0.0923	0.2094
w_{15}	0.0721	0.0999	0.2258

- Case II: there exist some $0 < \lambda_i < 0.3$ and $0.8 < \lambda_j < 1$
- Case III: there exist some $\lambda_i < 0$ and $0.8 < \lambda_j < 1$

In Case I, all classifiers in the ensemble system are normal and represent the true model well. In Case II, a number of classifiers weakly represent the true model. In Case III, a number of classifiers fail to represent the true model. We generate 100 different sets of classifier outputs for each case. The value of λ_i is randomly generated from the uniform distribution between the given interval for each classifier. The noise term is assumed to follow a normal distribution with zero mean and unit variance. For Cases II and III, the first five classifiers are selected as corrupt predictors. Then, the performance of the proposed combination method is compared with that of the bagging method in which all of the weights are the same, that is, $1/m$. The results are shown in Table 1.

In Case I, no difference exists between the performance of the proposed method and bagging. However, in Cases II and III where corrupted predictors exist, the proposed method performs better than bagging with a smaller standard deviation. The estimated weights for the individual classifiers from the proposed method are presented in Table 2. The weights are distributed almost equally among the classifiers in Case I. The proposed method yields smaller or negative weight values to the first five corrupted classifiers, which increases the performance of the ensemble. The results show that the proposed method captures corrupted predictors correctly and adjusts the weights for the individual predictors. Thus, the proposed method is compatible under normal circumstances and better when some base learners malfunction and become corrupted.

5. Experimental results

This section describes the experimental results of the proposed method and the comparative methods used on real data sets. The methods used in this experiment use neural networks and decision trees as the base model.

5.1. Data sets

The proposed approach is used on 19 real data sets. To evaluate the performance of the proposed method, we use both regression and classification data sets from UCI Machine Learning Repository [9]. Table 3 shows the summary of the data sets used. The data shown in Table 3 are widely used real-world problems in the machine learning community. The data sets present different characteristics with regard to the number and type (continuous and discrete) of features, as well as the number of cases and the type of problems (classification and regression).

5.2. Experimental settings

In this study, neural networks and decision trees are used as the base model. Neural networks and decision trees are unstable learners with large variance, so the ensemble technique is more effective on these learners. Classifier diversity can be achieved by adjusting the parameters and the resampling technique. There are many kinds of neural networks and decision trees. Among them we use multilayer neural network and binary decision tree.

Multilayer neural networks consist of a series of input, hidden, and output layers. Neural networks are generally presented as systems of neurons linked together with other neurons. The connections have numeric weights that can be tuned based on experience. The goal of the neural network algorithm is to determine a set of weights that minimize the total sum of squared errors. To learn the weights of a neural network model, we need an efficient algorithm that converges to the right solution. A common technique of training neural networks known as backpropagation has been developed in [25]. There are two phases in each iteration of the algorithm: the forward phase and the backward phase. During the forward phase, the weights obtained from the previous iteration are used to compute the output value of each neuron in the network. These neural networks are required to set a number of parameters. We trained the neural networks with a learning rate of 0.15, a momentum constant of 0.9, and randomly selected weights ranging from −0.5 to 0.5. The size of the hidden layer is determined according to the number of input and output units with five hidden units as a minimum. The number of epochs is based on the number of cases. We use a larger value of epochs as the size of data decreases. The training parameters used in the neural network experiments are shown in Table 3.

Another base model is binary decision tree. Binary decision tree is one of the decision tree models, in which only two child nodes can be generated from one parent node at maximum. In usual, the binary decision tree uses all variables simultaneously to find the optimal split boundary and so does one used in this paper. Like other decision trees, a binary decision tree requires some preset options and parameters. One of the important options is the split criterion. For the classification tasks, we set Gini diversity index, $1-\sum_{k=1}^{K} p_k^2$ where K is the number of classes and p_k is the portion of the class k, as the split criterion and all variables are used for each split. For regression tasks, we set the split criterion as minimizing mean squared error, $\sum_{i=1}^{N}(y_i-\hat{y}_i)^2$ where y_i is ith true value of the output and $\hat{y}_i$ is its predicted value. The other parameters are set the same for both classification and regression tasks. The minimum number of data points included in each leaf is one but for each parent node, the minimum required number of data points is 10. Also, every tree is pruned with the reduced error pruning scheme [3].

For the comparison, other popular ensemble algorithms, such as bagging, boosting, and single model, are applied to the same data sets. The results achieved by the proposed method, of which

Table 3
Summary of the data sets.

Data set	Cases	Class	Features		Neural network			
			Cont.	Disc	Inputs	Outputs	Hiddens	Epochs
Credit-a	690	2	6	9	47	1	10	35
Ionosphere	351	2	34	–	34	1	10	40
Statlog_aust	690	2	6	8	40	1	10	35
Statlog_german	1000	2	7	13	45	1	10	20
Hepatitis	155	2	–	16	16	1	5	40
Diabetes	768	2	9	–	8	1	5	30
Adult	30,162	2	6	8	96	1	20	20
Magic gamma telescope	19,020	2	10	–	10	1	5	20
Gisette	6000	2	–	5000	5000	1	100	20
Internet_ad	2359	2	3	1555	1558	1	50	20
Ailerons	13,750	regression	40	–	40	1	10	20
Housing_boston	506	regression	13	–	13	1	5	35
Pole telecom	5000	regression	48	–	48	1	10	20
Abalone	4177	regression	7	1	9	1	5	20
Wine quality_red	1599	regression	11	–	11	1	5	20
Wine quality_white	4898	regression	11	–	11	1	5	20
Elevator	8752	regression	18	–	18	1	5	30
Concrete slump	103	regression	10	–	10	1	5	70
Communities and crimes	1994	regression	99	–	99	1	25	20

variance parameter σ^2 is set to be unity, are compared with those obtained using the techniques detailed below:

- A single neural network and a decision tree (single).
- An ensemble where individual neural networks and decision trees are trained using randomly resampled training sets [1]. Given a set of responses related to an unknown sample, this method averages the responses and assigns the sample to the class with the largest value in the classification and the average value in the regression (bagging).
- An ensemble where the networks and decision trees are trained using weighted resampled training sets (boosting). In classification experiments, we use two kinds of boosting algorithms: Arcing method [10] and Ada method [11]. Boosting for regression is based on [8]. To perform a fair comparison, we set the maximum number of predictors in each ensemble to the same number in bagging and in the proposed method.

We obtain the average of the test error of each method over five standard 10-fold cross validation experiments. The data set is divided into 10 equal-sized sets, and each set is alternately used as test and training sets. For all ensemble methods, we create 25 predictors for each fold of an ensemble. The proposed method is used to the same predicted outputs of the bagging ensemble where an individual predictor is trained using randomly resampled training sets. Thus, the proposed method and bagging use the same predictors but have different weights.

5.3. Results

Comparison results are reported in Tables 4–7. In particular, Tables 4 and 5 show the classification error for classification problems when neural networks and decision trees are used as base learners, respectively. The columns in both tables indicate the different methods, namely, a single model (single), bagging (bagging), arcing (Arc), adaboosting (Ada), and the proposed method. Each row shows the results for each data set.

Tables 6 and 7 indicate the root mean squared error for regression problems when neural networks and decision trees are used, respectively. In the regression problems, the single model, bagging, boosting, and proposed method are used. Each row shows the results for each data set.

A statistical significance test is conducted to statistically validate the comparison results. We perform the non-parametric Wilcoxon signed rank test [7] ($\alpha = 0.05$) over 50 runs(five 10-fold CV). Wilcoxon tests the null hypothesis that the difference between two forecasting results comes from a distribution with a median of zero based on the ranks of the predicting errors.

According to the Wilcoxon test, the values in bold in the tables highlight the results that are significantly better than the second best result (values starred in the table) for each data set. When

Table 4
Classification error (%) for the data sets using (1) single neural network classifier; (2) bagging ensemble of neural networks; (3) arcing ensemble of neural networks; (4) Ada-boosting ensemble of neural networks; and (5) proposed method. Bold values represent the best statistically significant results, while starred values represent the second best results in this case. If the best result of the data set is insignificant, both the best and second best results are starred.

Data set	Neural networks				
	Single	Bagging	Boosting		Proposed method
			Arc	Ada	
Credit-a	14.17	13.94*	15.04	14.52	13.88*
Ionosphere	12.91	11.71	10.17*	10.86	**8.46**
Statlog_aust	15.48	13.83*	15.22	14.75	13.57*
Statlog_german	26.66	24.68*	27.48	24.96	**23.78**
Hepatitis	36.80	34.13*	36.40	36.00	**31.47**
Diabetes	24.16	23.21*	24.82	23.16*	24.13
Adult	16.62	16.37	16.80	16.25*	**14.83**
Magic gamma telescope	17.09	16.47	15.49*	15.83	**13.28**
Gisette	3.63	2.85	2.69	2.67*	**2.23**
Internet_ad	4.94	4.02	**3.00**	3.34*	3.54

Table 5
Classification error (%) for the data sets using (1) single decision tree; (2) bagging ensemble of decision trees; (3) arcing ensemble of decision trees; (4) Ada-boosting ensemble of decision trees; and (5) proposed method. Bold values represent the best statistically significant results, while starred values represent the second best results in this case. If the best result of the data set is insignificant, both the best and second best results are starred.

Data set	Decision tree				
	Single	Bagging	Boosting		Proposed method
			Arc	Ada	
Credit-a	18.9	13.42*	14.3	14.72	**13.22**
Ionosphere	12.23	9.03	6.29*	**6.17**	8.63
Statlog_aust	17.25	13.45*	14.46	13.88	**13.36**
Statlog_german	29.20	23.50*	25.10	26.10	**23.30**
Hepatitis	41.87	39.47	38.67*	44.27	**38.27**
Diabetes	28.95	24.37*	25.79	26.18	**24.29**
Adult	18.14	15.58	16.16	16.44	**15.52**
Magic_gamma_telescope	18.43	13.24	13.22	13.28	**13.17**
Gisette	6.80	4.28	2.68*	2.65*	3.43
Internet_ad	3.42	2.98	2.68	2.63*	**2.37**

Table 6
Regression results (RMSE) for the data sets using (1) single neural network; (2) bagging ensemble of neural networks; (3) boosting ensemble of neural networks; and (4) proposed method. Bold values represent the best statistically significant results, while starred values represent the second best results in this case. If the best result of the data set is insignificant, both the best and second best results are starred.

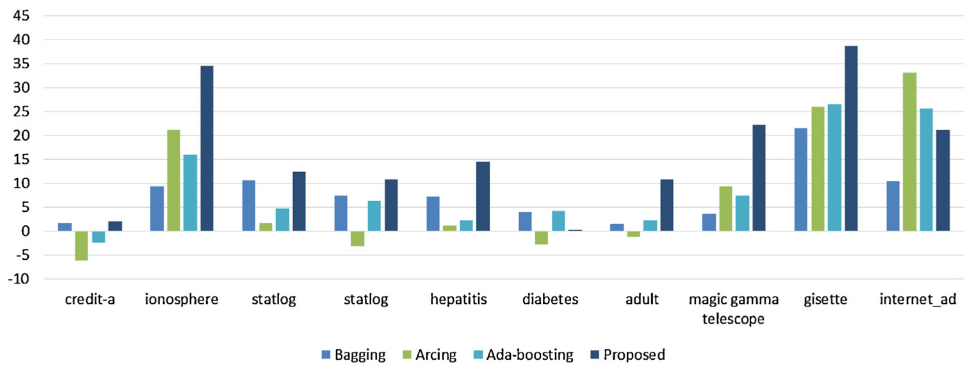
Data set	Neural networks			
	Single	Bagging	Boosting	Proposed method
Ailerons	0.0005	0.0003*	0.0007	**0.0002**
Housing_boston	3.9512	3.3357*	3.2844*	3.3458
Pole telecom	13.9010	10.5752*	10.8982	**10.3555**
Abalone	2.1020	2.0764*	2.9201	2.0732*
Wine quality_red	0.6520	0.6263*	0.6928	**0.6255**
Wine quality_white	0.7179	0.7042*	0.7589	**0.7037**
Elevator	0.0024	0.0023*	0.0025	0.0023*
Concrete slump	3.0202	2.0131*	2.3461	**1.6876**
Communities and crimes	0.3069	0.1491*	0.1552	**0.1490**

Table 7
Regression results (RMSE) for the data sets using (1) single decision tree; (2) bagging ensemble of decision trees; (3) boosting ensemble of decision trees; and (4) proposed method. Bold values represent the best statistically significant results, while starred values represent the second best results in this case. If the best result of the data set is insignificant, both the best and second best results are starred.

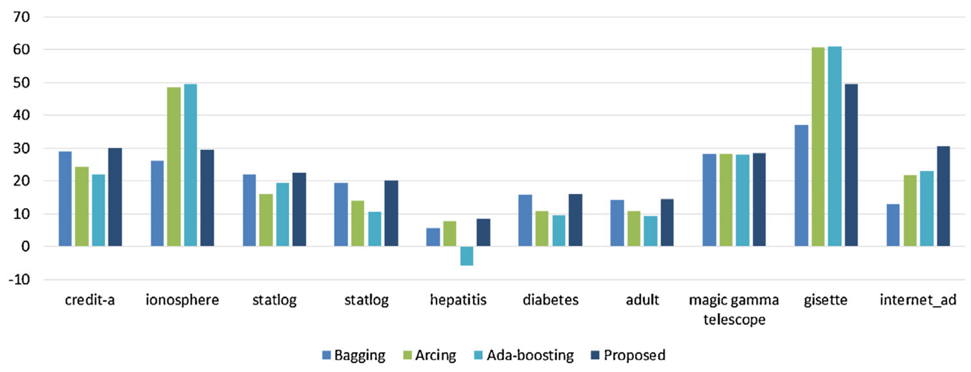
Data set	Decision tree			
	Single	Bagging	Boosting	Proposed method
Ailerons	0.0021	0.0016*	0.0020	0.0016*
Housing_boston	4.6282	3.3712	3.3709*	**3.3613**
Pole telecom	8.7607	6.3975*	6.4457	**6.3716**
Abalone	2.7769	2.2045	2.1970*	2.2045*
Wine quality_red	0.7523	0.5865*	0.5884	0.5864*
Wine quality_white	0.7975	0.6162*	0.6162	0.6162*
Elevator	0.0039	0.0029*	0.0038	**0.0029**
Concrete slump	6.3331	5.2187	5.1890*	5.1286*
Communities and crimes	0.1884	0.1392*	0.1395	0.1391*

the results do not present a statistically significant difference, the two best results are both starred. Table 4 indicates that the proposed method achieves the best performance for eight out of the ten data sets in classification problems. Among these data sets, six have results that are significantly better than the second best results (ionosphere, statlog german, hepatitis, adult, magic gamma telescope and gisette). In most cases, ensemble methods show better performance than the single model, except for some boosting algorithms on credit-a, statlog_german, diabetes and adult. In these cases, the error rates are increased when the boosting algorithm is applied. Sometimes, boosting shows the best performance but their results are not significant in this experiment except for internet_ad. When decision trees are used as the base learner (Table 5), the proposed method shows the best results for eight out of the ten data sets, but these results are different from those of the case using the neural networks. Boosting performs better in the ionosphere and gisette data sets and the proposed method and bagging outperform the other methods in the other cases. Tables 4 and 5 show that the performance of the ensemble methods depends on both the data set and the base learner used.

Table 6 shows that the proposed approach outperforms the other ensembles for eight out of the nine data sets in regression problems. Among these data sets, six have results that are significantly better than the second best results at $\alpha = 0.05$. For instance, for the concrete slump data set, the proposed method achieves a root mean square error (RMSE) of 1.6876. The second best result for this data set is obtained by bagging, which has achieved an RMSE of 2.0131. A similar result can be observed for the remaining data sets except for the housing_boston data set in which boosting shows the best performance, followed by bagging. As shown in Table 7,

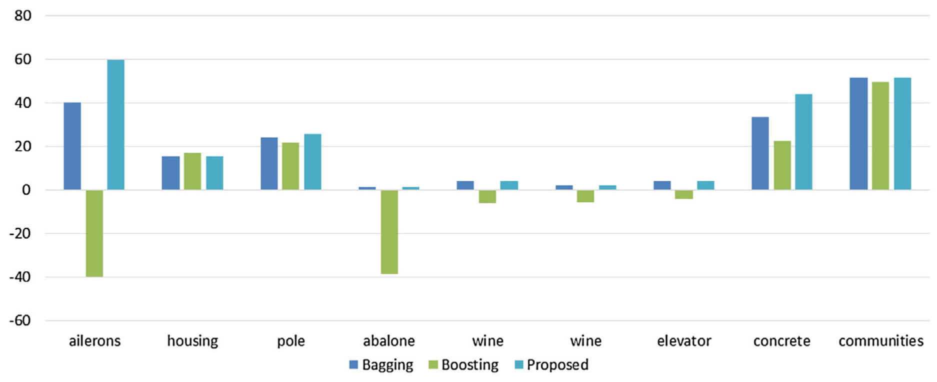


(a) Neural Networks

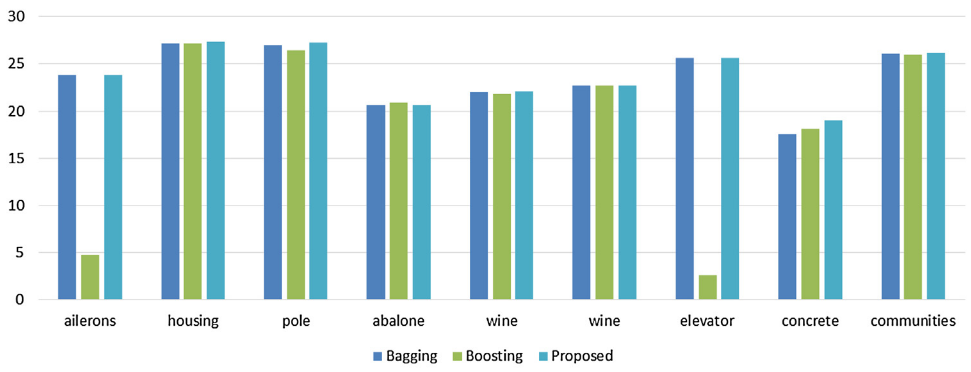


(b) Decision Trees

Fig. 4. Improvement ratio in classification error relative to single model.



(a) Neural Networks



(b) Decision Trees

Fig. 5. Improvement ratio in regression error (RMSE) relative to single model.

Table 8
Performance of different methods. Values are average rank of test error across the nine problems (low is good).

Method	Classification average rank	Regression average rank
Single	4.7	3.7
Bagging	2.85	1.89
Boosting		
Arc	3.15	2.83
Ada	2.9	
Proposed	1.4	1.17

except for abalone, for all of the data sets, the proposed method outperforms the others, and three data sets have significant results.

The average ranks of test errors across the nineteen problems using both neural networks and decision trees for classification and regression are presented in Table 8. In both classification and regression problems, the proposed method is ranked first and bagging is second. The boosting method is followed by the single model. All ensemble methods outperform the single model.

To better analyze Tables 4–7, we plot the percentage reduction in error for each method as a function of the single model error rate, as illustrated in Figs. 4 and 5. A 50% error reduction means that the error rate is reduced by half. For example, in the ionosphere data presented in Table 4, the error reduction of the proposed method compared with that of the single model has increased by 34% (12.91-8.46%).

Based on the experimental results on real data sets, the effectiveness of the proposed method can be validated through comparison with the most popular ensemble methods, such as bagging and boosting, under normal condition, that is, without the corrupted outputs.

6. Conclusion

In this study, we have proposed a novel approach for ensemble combination scheme using a latent consensus function that relates individual predictors. Our basic idea is that the predicted value of individual predictors is composed of the reflection of the real function value and a specific error term. According to this assumption, we determine weights for the ensemble combination using a separate training algorithm.

We have presented a comprehensive evaluation of the proposed method on simulated data set as well as real world data sets using neural networks and decision trees as base models. The experimental results show that the proposed method further improves its prediction performance by using self-correction for the malfunctioning base learners. By analyzing the results of corrupted toy data sets, we have shown that an ensemble can adjust weights by detecting the corrupt predictors in the learning process. Therefore, the proposed method can improve its performance even when a number of individual predictor outputs are corrupted.

Future research will investigate two aspects. First, the predictor selection ability of the proposed method can be enhanced. Second, the effectiveness of the proposed method can be enhanced when different base learners like a learning model that uses the concepts of unsupervised models for supervised tasks [13,16,15,22,21,23], data sizes, and distributions are used.

Acknowledgments

This research was supported partly by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MEST) (No. 2011-0030810 and 2011-0030811), Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2014R1A1A3051321), and partly by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MEST) (No. 2011-0017657).

References

[1] L. Breiman, Bagging predictors, Mach. Learn. 24 (1996) 123–140.
[2] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[3] C.A. Brunk, M.J. Pazzani, An investigation of noise-tolerant relational concept learning algorithms., in: Proceedings of the 8th International Workshop on Machine Learning, 1991, pp. 389–393.
[4] C. Catal, S. Tufekci, E. Pirmit, G. Kocabag, On the use of ensemble of classifiers for accelerometer-based activity recognition, Appl. Soft. Comput. 37 (2015) 1018–1022.
[5] Q. Dai, T. Zhang, N. Liu, A new reverse reduce-error ensemble pruning algorithm, Appl. Soft. Comput. 28 (2015) 237–249.
[6] B.V. Dasarathy, B.V. Sheela, Composite classifier system design: concepts and methodology, Proc. IEEE 67 (1979) 708–713.
[7] F.X. Diebold, R.S. Mariano, Comparing predictive accuracy, J. Bus. Econ. Stat. 20 (1995) 134–144.
[8] N. Duffy, D. Helmbold, Boosting methods for regression, Mach. Learn. 47 (2002) 153–200.
[9] M. Lichman, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2013 http://archive.ics.uci.edu/ml.
[10] Y. Freund, R. Schapire, Experiments with a new boosting algorithm, in: Proceedings of the Thirteenth International Conference on Machine Learning, Bari, Italy, 1996, pp. 148–156.
[11] Y. Freund, R. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1997) 119–139.
[12] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. 12 (1990) 993–1001.
[13] H. Heo, H. Park, N. Kim, J. Lee, Prediction of credit delinquents using locally transductive multi-layer perception, Neurocomputing 73 (2009) 169–175.
[14] W.G. Howe, Some Contributions to Factor Analysis, Oak Ridge National Laboratory, Oak Ridge, TN, 1955.
[15] K.-H. Jung, N. Kim, J. Lee, Dynamic pattern denoising method using multi-basin system with kernels, Pattern Recognit. 44 (8) (2011) 1698–1707.
[16] K.-H. Jung, J. Lee, Probabilistic generative ranking method based on multi-support vector domain description, Inf. Sci. 247 (2013) 144–153.
[17] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of local experts, Neural Comput. 3 (1991) 79–87.
[18] M.J. Jordan, R.A. Jacobs, Hierarchical mixtures of experts and the EM algorithm, Neural Comput. 6 (1994) 181–214.
[19] H.F. Kaiser, The varimax criterion for analytic rotation in factor analysis, Psychometrika 23 (1958) 187–200.
[20] N. Kim, K.-H. Jung, Y.S. Kim, J. Lee, Uniformly subsampled ensemble (USE) for churn management: theory and implementation, Expert Syst. Appl. 39 (2012) 11839–11845.
[21] K. Kim, J. Lee, Sentiment visualization and classification via semi-supervised nonlinear dimensionality reduction, Pattern Recognit. 47 (2) (2014) 758–768.
[22] K. Kim, J. Lee, Nonlinear dynamic projection for noise reduction of dispersed manifolds, IEEE Trans. Pattern Anal. Machine Intell. 36 (11) (2014) 2303–2309.
[23] K. Kim, Y. Son, J. Lee, Voronoi cell-based clustering using a kernel support, IEEE Trans. Knowl. Data Eng. 27 (4) (2015) 1146–1156.
[24] B. Krawczyk, M. Woźniak, G. Schaefer, Cost-sensitive decision tree ensembles for effective imbalanced classification, Appl. Soft. Comput. 14 (2014) 554–562.
[25] D. Rumelhart, G. Hinton, R. Williams, Learning internal representations by error propagation, in: D. Rumelhart, J. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, 1986, pp. 318–363.
[26] R.E. Schapire, The strength of weak learnability, Mach. Learn. 5 (1990) 197–227.
[27] D.H. Wolpert, Stacked generalization, Neural Netw. 5 (1992) 241–259.