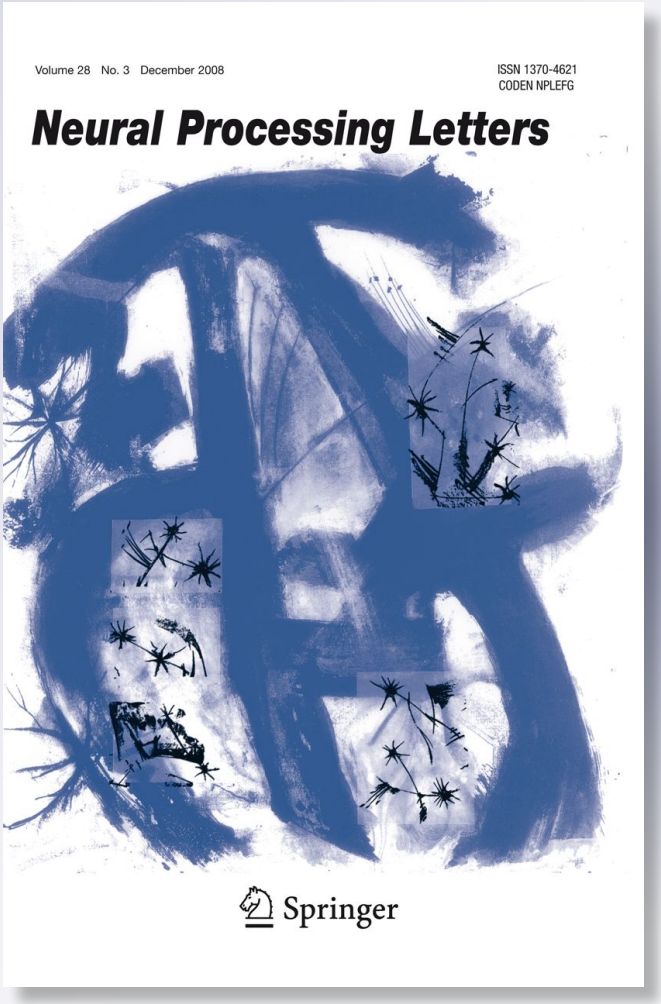
Making Diversity Enhancement Based on Multiple Classifier System by Weight Tuning

Mehdi Salkhordeh Haghighi, Abedin Vahedian & Hadi Sadoghi Yazdi

Neural Processing Letters

ISSN 1370-4621
Volume 35
Number 1

Neural Process Lett (2012) 35:61-80
DOI 10.1007/s11063-011-9204-y



Springer

Neural Process Lett (2012) 35:61–80
DOI 10.1007/s11063-011-9204-y

Making Diversity Enhancement Based on Multiple Classifier System by Weight Tuning

Mehdi Salkhordeh Haghighi · Abedin Vahedian · Hadi Sadoghi Yazdi

Published online: 9 November 2011

Abstract This article presents a new method to construct multiple classifier system by making diverse base classifiers using weight tuning. In the method presented, base classifiers are multilayer perceptions which creates diverse base classifiers using a three-step procedure. In the first step, base classifiers are trained for acceptable accuracy. In the second step, a weight tuning process tunes their weights such that each one can distinguish one class of input data from the others with highest possible accuracy. An evolutionary method is used to optimize efficiency of each base classifier to distinguish one class of input data in this step. In the third step, a new method combines the results of the base classifiers. As diversity is measured and monitored throughout the entire procedure, it is measured using a confusion matrix. Superiority of the proposed method is discussed using several known classifier fusion methods and known benchmark datasets.

Keywords Combining classifiers · Classifier fusion · Classifier diversity · Multiple classifier system diversity

1 Introduction

Combination of classifiers as a unified system has demonstrated to perform better than single classifiers [1,2]. As each classifier produces different errors in different regions of the input space, performance of each base classifier in some regions of input space would be better than other regions [3]. This property makes the combined system more accurate than single ones. The classifier system formed by combining multiple different classifiers is called multiple classifier system (MCS).

M. Salkhordeh Haghighi (✉) · A. Vahedian · H. Sadoghi Yazdi
Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
e-mail: haghighi@ieee.org

A. Vahedian
e-mail: vahedian@um.ac.ir

H. S. Yazdi
e-mail: h-sadoghi@um.ac.ir

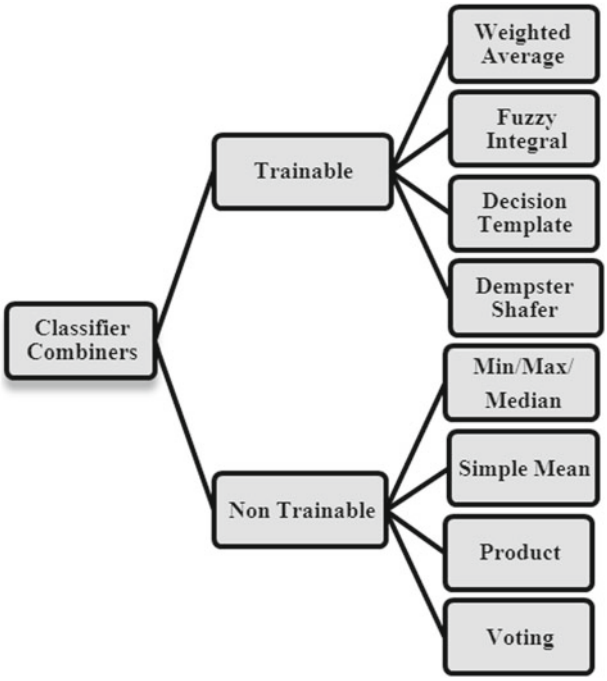


Fig. 1 Hierarchy of MCS fusion methods

Nevertheless, the proper strategy for combining simple classifiers is the main issue in MCS construction. As there is no best combination method to be used in all different applications, various fusion approaches have been reported [1,4]. While a majority vote may be suitable if only labels are available, continuous outputs such as posterior probabilities require average or other linear combinations [5]. If the classifier outputs are interpreted as fuzzy membership values, fuzzy approaches could be used. It is also possible to train the combiner separately, using outputs of the base classifiers as new features [6]. One of the earliest works on MCS dates back to Dasarathy and Sheela's 1979 work, which discussed the idea of partitioning the feature space using two or more classifiers [7].

Amongst the various approaches in categorizing MCS, one major category is the capability of being trained. They need additional training after the training process of the base classifiers is completed. With no training capability, combiner training is not required once the base classifiers have been trained individually. This is shown in Fig. 1.

Another class of MCS is those that develop the combiner during training of individual classifiers. Some known trainable combiners are Weighted Average, Fuzzy Integral, Decision Template, and Dempster Shafer. The most known non trainable combiners are Min, Max, Median, Simple Mean, Product, OWA, and Majority Vote.

1.1 Diversity in MCS

It is well known that combination of the outputs of several classifiers is only useful if they disagree on some inputs [8,9]. We refer to the measure of disagreement as the *diversity* of the

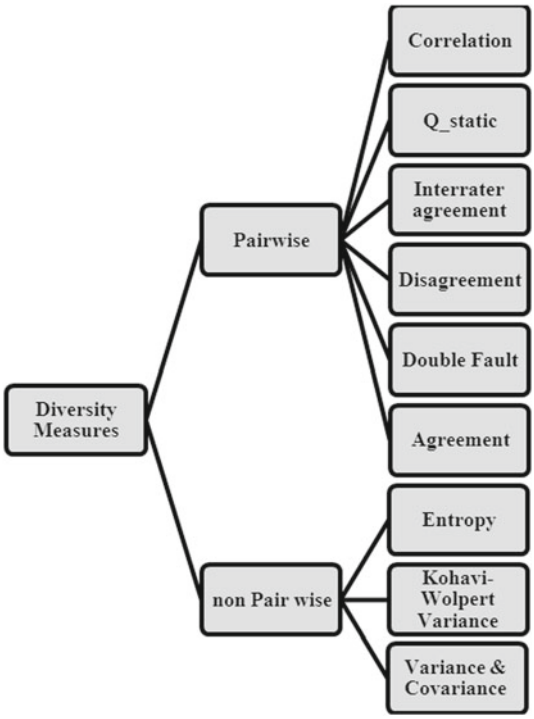


Fig. 2 Types of diversity measures

MCS. For regression problems, mean squared error is generally used to measure accuracy, and variance is used to measure diversity. Constructing a diverse committee in which each hypothesis is as different as possible, while maintaining consistency with the training data is known to be a theoretically important property of a good MCS method [10]. Although all successful construction methods encourage diversity to some extent, few have focused directly on the goal of maximizing diversity.

There is however, no general agreement about how to quantify the notion of diversity among a set of classifiers. As a starting point to quantify diversity, the measures can be categorized into two groups, pair wise and non pair wise [11] as shown in Fig. 2. To apply pair wise measures, it is necessary to compute averages over the set of paired classifiers diversities. Non pair wise measures attempt to measure diversity of a set of classifiers directly, for example based on variance entropy or some other measures.

On the other hand, diversity among classifiers can be analyzed from another point of view on what is considered as the population. Suppose we have K classifiers each classifying every one of the N items of input data (z_i). Thus, the population can be formed based on either the data or the classifiers [12]. In Data based population, each individual in the population is formed by outputs of base classifiers for one input data (z_i). In contrast, each individual in classifier based population is formed by outputs of one classifier for all input data z [13].

Therefore, diversity within data population is the diversity of the MCS with N population each one with K features with respect to a key item of input data. As a result, diversity can be calculated by averaging over all input data space. In contrast, in classifier based population, there are K populations each with N features; as a result, diversity would belong to

Table 1 Relationship table between classifiers i and j

	D_j is correct	D_j is incorrect
D_i is correct	a	b
D_i is incorrect	c	d

the outputs of one particular classifier for all different input samples. As a result, for the classifiers to be diverse enough, this type of population has little importance.

1.2 Types of diversity measures

Diversity measures are mainly groups as following:

1.2.1 Statistical methods

Consider classifiers with outputs 0 or 1. Output is 1 when input data x_i is classified correctly and 0 otherwise. In such classifiers, correlation is computed based on Table 1 as a measure of diversity. Entries of the table are the probabilities that each combination occurs. In this table, sum of all the probabilities is equal to 1 ($a+b+c+d=1$).

The correlation between classifiers D_i and D_j is calculated based on (1) [1].

$$\rho_{i,j} = \frac{ad - bc}{\sqrt{(a+b)\,(c+d)\,(a+c)\,(b+d)}} \tag{1}$$

Another statistical measure is Q_static. Using Table 1, the Q_static measure of diversity is computed by $Q_{i,j} = \frac{ad-bc}{ad+bc}$. For the classifiers that are statistically independent, the measure is 0 and $Q_{i,j} \in [-1, 1]$. Positive value of Q shows that the classifiers can recognize an object similarly.

Interrater agreement (k) is also used as a statistical measure. If C class labels are available, k is defined on a $C \times C$ coincidence matrix M. The entry $m_{k,s}$ would be the proportion of the dataset which classifier D_i labels as w_k while D_j labels it as w_s. The value of k for the two classifiers is calculated by $k_{i,j} = \frac{\sum_k m_{k,k} - \mathrm{ABC}}{1-\mathrm{ABC}}$. In the equation, the value of $\sum_k m_{kk}$ is the observed agreement between the classifiers. The value of ABC is the agreement by chance which is computed by $\sum_k \left(\sum_s m_{k,s}\right)\left(\sum_s m_{s,k}\right)$. Low values of k indicate higher disagreement and higher diversity.

1.2.2 Disagreement and double fault measures

In another view, disagreement measure is the probability that the two classifiers have disagreement on input data. Based on the Table 1, the diversity between classifier pair $D_{i,j}$ is measured by $D_{i,j} = b + c$. This measure is called disagreement measure. In contrast, double fault measure is the probability that both classifiers decide incorrectly. Based on the Table 1, it is computed by $\mathrm{DF}_{i,j} = d$.

1.2.3 Entropy measure

Diversity of ensemble for a particular input $z_j \in Z$ is maximized if $[L/2]$ of classifiers make the same decision on one class of input data while the other $L - [L/2]$ classifiers make different decisions on it. If they all agree or disagree in decision on some input, they cannot

be diverse. One of the measures of this type of diversity is computed using equation (2). In this equation, $E \in [0, 1]$, where 1 indicates the highest diversity.

$$E = \frac{1}{N}\frac{2}{L-1}\sum_{j=1}^{N}\min\left\{\left(\sum_{i=1}^{L} y_{i,j}\right),\left(L-\sum_{i=1}^{L} y_{i,j}\right)\right\} \tag{2}$$

1.2.4 Kohavi–Wolpert variance

A decomposition formula is introduced by Kohavi and Wolpert [14]. If y is the predicted class label for some input data x, the variance of y over different data sets is as (3). The value of $p(y = 1|z_j)$ is estimated as an average on different sets of test data.

$$Var_x = 1/2(1 - \sum_{i=1}^{c} P(y = \omega_i|x)^2) \tag{3}$$

1.2.5 Agreement measure

Agreement measure is defined as (1-D) where D refers to disagreement measure [15]. To define this measure, the following formulation is defined. The output of a classifier is defined to be 1 if a pattern is correctly classified and 0 otherwise. Let the jth classifier output under this labeling scheme be a k-dimensional binary vector given by $y_{m,j}$ where $m = 1 \ldots k$. Based on these definitions, equations (4)–(7) are defined. In these equations, $\overline{y}$ indicates logical complement.

$$N_{i,j}^{1,1} = \sum_{m=1}^{k} y_{m,i} \quad and \quad y_{m,j} \tag{4}$$

$$N_{i,j}^{0,0} = \sum_{m=1}^{k} \overline{y}_{m,i} \quad and \quad \overline{y}_{m,j} \tag{5}$$

$$N_{i,j}^{1,0} = \sum_{m=1}^{k} y_{m,i} \quad and \quad \overline{y}_{m,j} \tag{6}$$

$$N_{i,j}^{0,1} = \sum_{m=1}^{k} \overline{y}_{m,i} \quad and \quad y_{m,j} \tag{7}$$

Equations (4) and (5) are the number of agreements between classifiers i and j in correct and incorrect classifications respectively. Equations (6) and (7) are the number of disagreements between classifiers i and j. Therefore, agreement measure is computed by (8):

$$A_{i,j} = 1 - \frac{N^{0,1} + N^{1,0}}{N^{1,1} + N^{0,0} + N^{0,1} + N^{1,0}} \tag{8}$$

1.2.6 Variance and covariance measures

The research done by Krogh and Vedelsby [10] has revealed that in a single data point the quadratic error of the MCS estimator is guaranteed to be less than or equal to the average quadratic error of the component estimators or base classifiers (9).

$$(f_{ens} - d)^2 = \sum_i w_i \, (f_i - d)^2 - \sum_i w_i \, (f_i - f_{ens})^2 \tag{9}$$

In (14), f_{ens} is a convex combination of component estimator f computed as $f_{ens} = \sum_i w_i f_i$ and $\sum_i w_i = 1$.

In these equations, f_i is the ith MCS output and d is the desired output. The value of $\sum_i w_i (f_i - d)^2$ is the weighted average quadratic error of ensemble members. The term $\sum_i w_i (f_i - f_{ens})^2$ indicates the amount of variability among the MCS member's response for a specific pattern.

Geman et al. [16] showed that the Bias/variance decomposition for quadratic loss states that the generalization error of an estimator can be broken down into two components: bias and variance. These two usually work in opposition to one another meaning that attempts to reduce the bias component will cause an increase in variance, and vice versa. The decomposition is shown in (10).

$$\begin{aligned} E\left\{(f - < d >)^2\right\} &= E\left\{(f - E\{f\})^2\right\} + (E\{f\} - < d >)^2 \\ \rightarrow MSE\,(f) &= var\,(f) + bias(f)^2 \end{aligned} \tag{10}$$

In the above equations, $<d>$ is the expected value of the target point given the noise. If the estimator is a convex combined MCS, the variance component can be broken down further, as represented in equations (11)–(13) [17].

$$\overline{bias} = \frac{1}{M} \sum_i (E\{f_i\} - < d >) \tag{11}$$

$$\overline{var} = \frac{1}{M} \sum_i \left[E\left\{(f_i - E\{f_i\})^2\right\}\right] \tag{12}$$

$$\overline{covar} = \frac{1}{M(M-1)} \sum_i \sum_{j \neq i} E\left\{(f_i - E\{f_i\})\left(f_j - E\{f_j\}\right)\right\} \tag{13}$$

Therefore, mean square error is decomposed as (14).

$$E\left\{\left[\left(\frac{1}{M}\sum_i f_i\right) - < d >\right]^2\right\} = \overline{bias}^2 + \frac{1}{M}\overline{var} + \left(1 - \frac{1}{M}\right)\overline{covar} \tag{14}$$

It is therefore obvious that the *MSE* of an MCS depends on the amount of error correlation between MCS members, summarized in the covariance component. It is expected to decrease the covariance without any increases in the bias or variance components. In (14) bias and variance are both positive, while the covariance can be negative.

1.3 Diversity creation methods

Diversity creation methods are categorized into explicit and implicit as shown in Fig. 3. Explicit methods have special attention to optimize some diversity metrics during MCS creation. For instance, Boosting [18] is an explicit method as it directly manipulates the training data distributions to ensure some form of diversity in the MCS, even if there is no guarantee to be the proper diversity.

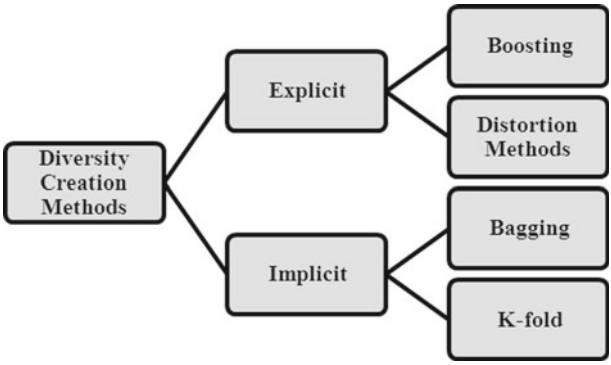


Fig. 3 Examples of diversity creation methods

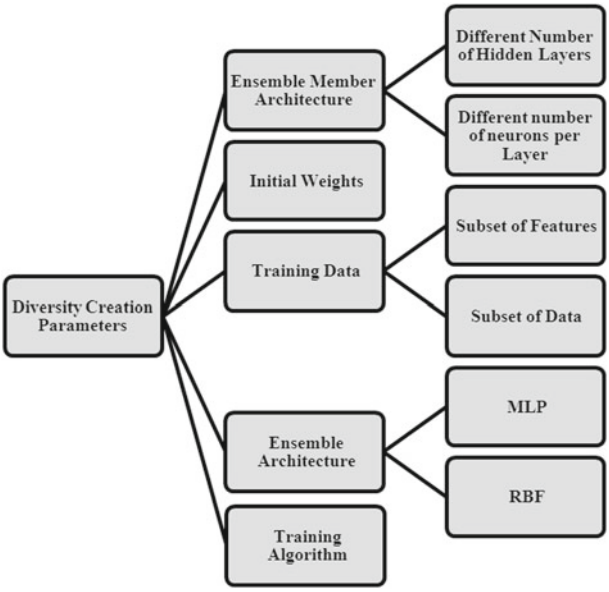


Fig. 4 Various parameters affecting diversity

Implicit diversity creation methods have neither special attention to diversity nor diversity measures during MCS creation. An example of implicit methods is bagging [19]. This method randomly samples training data for training of each MCS member with replacement to produce different MCS members. There is no measuring process to ensure having acceptable diversity.

1.3.1 Diversity parameters

Various categories of the parameters affecting diversity are namely initial weights, training data used, architecture of the base classifiers, and training algorithms [20] which are shown in Fig. 4.

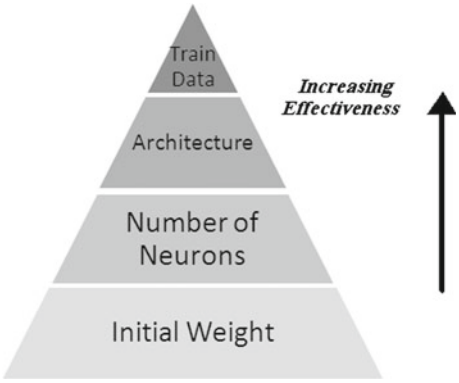


Fig. 5 Effectiveness of different parameters in diversity creation

The methods using different initial weights, introduce different starting points; therefore the points they converge at are different. Starting each MCS member with different random initial weights will increase the probability of following a different path with respect to the other members. This is the most common way to generate an MCS, but is not necessarily an effective way of generating enough diversity.

Some of the methods used for creating diversity attempt to use different training data to train each MCS member. This approach divides training data space by two different strategies. One strategy selects all features but a subset of input samples for training each MCS member. The other strategy selects all input data samples but a subset of features for training each MCS member. However, some other methods use a pre processing function on features, for example logarithm, to map input data on a different plane which are called distortion methods [21]. Duin and Tax [22] found that combining the results of one type of classifier on different feature sets is far more effective than combining the results of different classifiers on one feature set.

Nevertheless, some of the methods used for creating diversity, use different structures for each member. Works have been done on the number of hidden layers and the number of neurons in each layer. Partridge and Yates [23] showed that the number of hidden nodes after initial weights attains least positive effect on creating diversity in neural network MCS. They investigated the effect of using different MCS members utilizing MLP and RBF which were found to be more effective than hidden nodes adjustments. Figure 5 shows effectiveness of different strategies in making diversity.

To gain more diverse classifiers, one type of Bagging is called Random Forest [24]. The MCS consists of decision trees built on bootstrap samples. Another method for building diverse classifiers is Random Subspaces method [25]. In this method, each MCS member is constructed based on a different subset of features randomly selected from the original feature set. In this method, the individual classifiers can be built in parallel, independent of each other.

1.3.2 Special diversity creation methods

Rodriguez and Kuncheva [26] introduced a combined method for creation of accurate and diverse MCS based on principal component analysis (PCA) called Rotation Forest. In this method, input feature set F is split randomly into preferably K disjoint subsets. For each of these subsets, PCA produces weights for features of these K sets stored in a separate matrix called rotation matrix R_α. The matrix is used to generate different training sets by rearranging

its columns. Then, training data set for each classifier D_i is produced by equation $X \times R_{\alpha_i}$, where X is the training data matrix and R_{α_i} is the same as R_α except that the columns of R_α are rearranged to produce training data for ith classifier. The method uses PCA to increase both accuracy and diversity.

However, some methods use special properties of fuzzy clustering to produce training data. It is to be noted that the MCS produced should be not only accurate but also diverse enough in almost all input space. Since accuracy and diversity are inversely related to each other, increasing accuracy alleviates diversity and vice versa. Therefore, MCS creation methods should balance between these two properties. One of the fuzzy methods used for creating MCSs is introduced by Zhang and Lu [27]. In their research, bootstrap weights for training data are produced based on FCM method and definition of entropy such that new training data for each classifier can be obtained. Hence, the amount of fuzzy information in each input data X_i is determined by entropy measure $Info\,(X_i) = -\sum_{j=1}^{c} \mu_{i,j} \log_2 \mu_{i,j}$.

After calculating membership values $\mu_{i,j}$ by FCM, the value of $info\,(x_i)$ is zero if one $\mu_{i,j}$ equals 1 while all the others are zero. Therefore, x_i can be easily labeled. If each $\mu_{i,j}$ of x_i is $1/c$, worst case happens and $info\,(x_i)$ is maximized causing difficulty in labeling x_i. If one weight is calculated for each training instance according to $info\,(x_i)$ then each base classifier is built based on the weighted bootstrap samples and the original dataset.

In this article, a new combined method is introduced to increase diversity during MCS construction. This article is organized in the following sections. Section 2 describes details of the proposed new method. Section 3 examines the efficiency based on some known datasets while comparing the results with those obtained by other fusion methods. Section 4 provides a conclusion and some ideas for future work.

2 Disjoint class diversity enhancement method (DICDEM)

The proposed method is categorized as explicit as during MCS creation, diversity is continuously monitored based on some criteria. The MCS is incrementally created by training one base classifier in each step.

2.1 Motivation

As mentioned, accuracy and diversity are two important items during MCS construction which are in contrast to each other. In the method, the motivation is that both accuracy and diversity are considered during MCS creation. In contrast to existing methods, in our new method, emphasis is on both diversity and accuracy. To do this, in a hierarchical approach, each base classifier is trained first such that desired accuracy is achieved. Second, each base classifier is assigned to one of the classes of input data. Third, weights of each base classifier are tuned such that maximum accuracy in detecting the assigned class of data is gained while keeping accuracy of detecting the other classes. This adjustment process increases diversity of MCS because each base classifier is trained more to detect one class of data. This process is shown in Fig. 6.

2.2 Description of the DICDEM method

Since different training sets have the most influence on diversity, in the first step, structure of all the base classifiers is the same. Therefore, all the base classifiers in the MCS have the same number of hidden layers and nodes. For example, monitoring diversity and accuracy

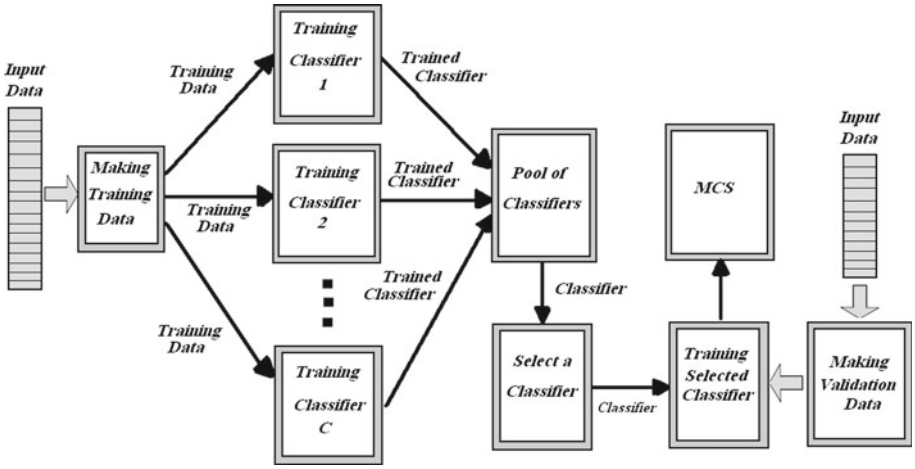


Fig. 6 Construction of diverse base classifiers

during creation of ensemble systems may use expression $Fitness\,(i) = accuracy\,(i) + \alpha \times diversity(i)$ to measure fitness of classifier i according to these parameters [28] in which α indicates degree of influence of diversity. The value of $diversity(i)$ is the contribution of classifier i diversity to the total MCS diversity.

This method basically focuses on the distribution of data in each class, training procedure of base classifiers and their outputs. One key idea to have more diverse classifiers is to train them such that each base classifier can distinguish data of one class from all others more accurately. For training such diverse classifiers, DICDEM uses a three step procedure. In the method, accuracy and diversity are considered in a stepwise manner. After gaining acceptable accuracy, focus will go on diversity. This stepwise procedure ensures enough attention to both of them.

In step 1 of DICDEM, each base classifier is trained with training data. In this step, it is expected to increase efficiency or accuracy of each base classifier for training data. To have more diverse classifiers while keeping accuracy, in step 2 each base classifier is tuned for distinguishing one class of training data more efficiently than others. As a result, if one classifier is adjusted for one class of data, for a C class dataset, C base classifiers are required. In step 2, the process of tuning each base classifier for distinguishing one class among others is done by tuning network weights obtained in step 1. A special evolutionary algorithm is used for tuning the weights of each trained base classifier. Basic strategy is to keep accuracy while increasing diversity. After tuning each base classifier, in step 3, the final combiner is trained to combine results of the tuned base classifiers. Training of the combiner is necessary if one of the trainable combiners is used for combining outputs of the base classifiers. This procedure is shown in Fig. 7.

2.3 Evolutionary algorithm

Once the base classifiers are trained, it is time to tune each of them for distinguishing one class of input data from the others. Since the base classifiers are NNs, the number of hidden layers and the number of neurons in hidden layers are also effective in diversity. These parameters are, therefore, set for all the base classifiers.

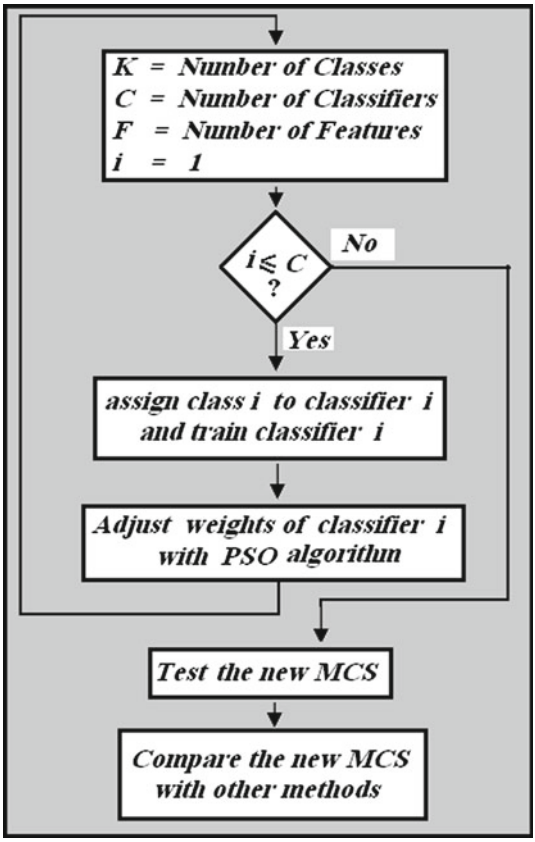


Fig. 7 Basic steps of DICDEM method

2.3.1 PSO parameters and steps

In the following algorithm, C is the number of classifiers, K is the number of classes, and F is the number of features in input data space. If one base classifier is tuned for one class of input data, total number of base classifiers necessary for making MCS is K, such that $C = K$. It is possible to tune more classifiers for each class (for example, m classifiers per class), in this case, total number of classifiers in the MCS is $C = m \times K$. In another approach, it is possible to tune more classifiers for complex classes than simple ones. In this case, the number of classifiers is more than the number of classes ($C > K$).

Since PSO is a population based algorithm, basic parameters of the algorithm are population size, structure and size of each individual, and fitness function. Figure 8 shows the structure of one individual with details. In this figure, each base classifier has one input, two hidden and one output layers.

Network weights are grouped in three disjoint matrices $W_{1,2}$, $W_{2,3}$ and $W_{3,4}$. Each individual is a string formed by concatenation of all the rows of these three matrices as shown in Fig. 8. If L is the size of each individual, and n_1 through n_4 are the number of neurons in each layer, L is computed based on (15):

$$L = n_1 \times n_2 + n_2 \times n_3 + n_3 \times n_4 \tag{15}$$

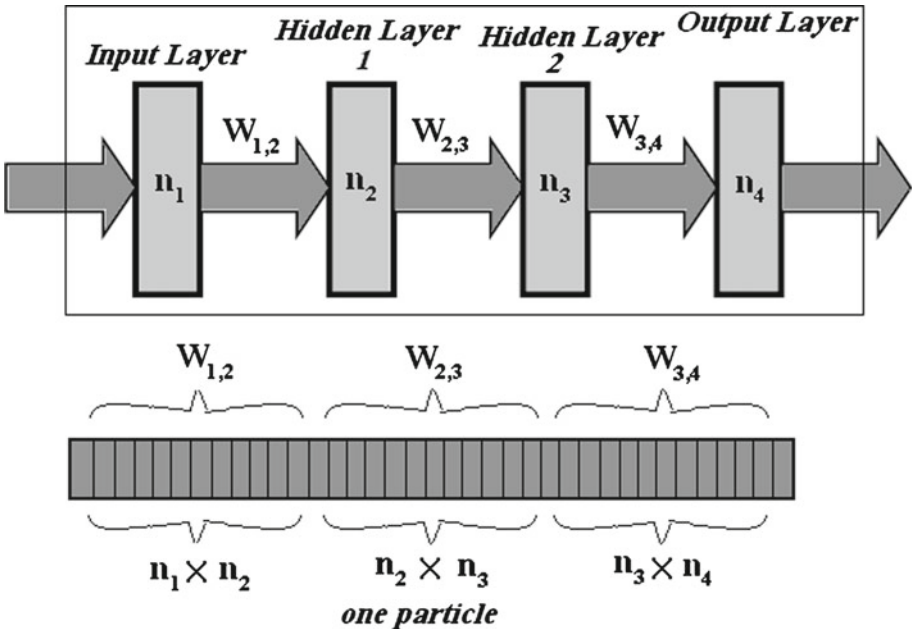


Fig. 8 Structure of each individual in the population

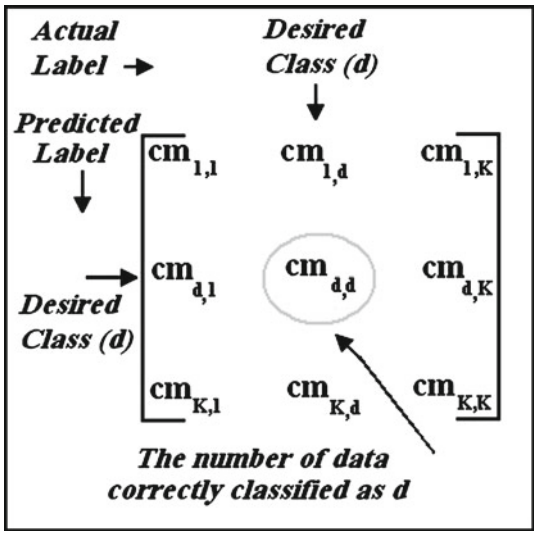


Fig. 9 Structure of CM for a classifier

In Fig. 8 the assumptions used are $n_1 = F + 1$ because one neuron for each feature in input layer and a bias are needed, $n_2 = K$ because output vector is used, $W_{1,2} = \{w_{i,j}\}_{n_1 \times n_2}$, $W_{2,3} = \{w_{i,j}\}_{n_2 \times n_3}$, and $W_{3,4} = \{w_{i,j}\}_{n_3 \times n_4}$.

Therefore, with initial population size P, the population is initialized with P copies of the weights obtained following the training process of the base classifiers. As mentioned above, each individual includes all weights of a base classifier; therefore, each weight in the individual is a particle. As a result, velocities of particles form a matrix V as $V = \{V_{i,j}\}_{P \times L}$.

```
PSO algorithm
1-   Input: W1,2, W2,3, W3,4 weights of the selected base classifier;
2-   Output: W'1,2, W'2,3, W'3,4 adjusted weights of the selected classifier;
3-     Initialize parameters of PSO:
4-   L=size(W1,2)+size(W2,3)+size(W3,4)  ; size of each particle;
5-   PopulationSize=100;
6-   X=generate random population;
7-   For  iteration = 1: maximum iteration
8-         Fx=compute_ fitness();
9-         Compute_displacement_for_particles();
10-       Update_particle_values_based_on_displacement();
11-  End for
12-  [W1,2, W2,3, W3,4]= particle with maximum fitness();
end
```

Fig. 10 PSO algorithm for adjusting weights of one classifier

This value is used in line 10 in Fig. 10 to compute new values for particles. Initially, all the velocities are set randomly. Therefore, it causes random distribution of the particles in input space.

One of the key points in the PSO algorithm is to determine proper fitness function. Since primary goal of this algorithm is to adjust weights of the selected classifier such that it can distinguish one class of input data called desired class from others with higher accuracy, total diversity of MCS is also enhanced. To design a fitness function such that both accuracy and diversity are considered, two vital requirements for the fitness function are validating data and confusion matrix (CM). Structure of the CM is shown in Fig. 9.

Nevertheless, according to the CM obtained for the validation data, three criteria is defined such that the fitness function is properly designed. These three criteria for the selected classifier are total accuracy (TA), desired class accuracy (DCA) and other class accuracy (OCA). As Fig. 9 shows, with K classes, numbered 1 through K, if d is the desired class for the selected classifier, $cm_{i,j}$ indicates the number of validation data in class j which are classified in class i. if i and j are the same, the value of $cm_{i,j}$ is the number of validation data that correctly classified by the selected classifier. Since the CM is a square matrix, $cm_{i,j}$ with $i = j$ is a diagonal element. If $i \neq j$, $cm_{i,j}$is total number of validation data in class j that incorrectly classified in class i.

In the PSO algorithm, the classifier assigned to the desired class is adjusted such that accuracy of determining the desired class is enhanced. As a result, TA indicates ratio of the number of correctly classified validation data (sum of diagonal elements of CM) with respect to total number of validation data (sum of all elements of CM) as defined by equation (16).

$$\mathrm{TA} = \frac{\sum_{i=1}^{K} cm_{i,i}}{\sum_{i=1}^{K}\sum_{j=1}^{K} cm_{i,j}} \tag{16}$$

On the other hand, it is expected that the fitness function returns higher values for the individuals having higher accuracy in detecting desired class of validation data (d). For this to happen, another criterion called DCA is defined. For class d, DCA is the ratio of the number of correctly classified validation data in class d $\left(cm_{d,d}\right)$ to total validation data in class d (sum of the elements in column p of CM) as shown by equation (17). It is obvious that for the individuals with higher DCA, the fitness function should also return higher values.

$$\mathrm{DCA} = \frac{cm_{d,d}}{\sum_{i=1}^{K} cm_{i,d}} \tag{17}$$

Nevertheless, DCA is not sufficient for determining fitness of individuals. This is because increasing the accuracy of detecting the desired class may also result the accuracy of detecting other classes to decrease. Therefore, another criterion called OCA is defined. The need for OCA in the fitness function is due to the fact that for the selected classifier to be tuned to distinguish the desired class with the highest possible accuracy, the accuracy of detecting other classes of data should not be decreased. Since OCA shows accuracy of detecting other classes of data except the desired class, this parameter should be considered in the fitness function. Based on the structure of confusion matrix, OCA is the ratio of the number of correctly classified validation data except desired class d to total number of validation data except desired class as shown in (18).

$$\mathrm{OCA} = \frac{\sum_{\substack{i=1,\\ i\neq d}}^{K} cm_{i,i}}{\sum_{\substack{i=1,\\ i\neq d}}^{K} \sum_{j=1}^{K} cm_{i,j}} \tag{18}$$

Typically, fitness function should combine DCA and OCA in a proper manner. Since the main goal of PSO is to maximize DCA while taking OCA into consideration, the fitness function is defined as equation (19).

$$fitness = \alpha \times \mathrm{DCA} + (1 - \alpha) \times \mathrm{OCA} \tag{19}$$

where, α is a parameter that reflects the influence of OCA.

As a result, defining such fitness function not only increases accuracy of detecting desired class, but also increases the diversity of MCS. Accuracy of each base classifier in detecting the desired class is satisfied by the first term in fitness function (DCA). The second term in the fitness function tries to keep accuracy of detecting the other classes by factor α. The fitness function forces the base classifiers to cover input data space for detecting each class of input data with maximum possible accuracy while increasing total diversity of MCS. Details of the PSO algorithm is shown in Fig. 10.

In Fig. 10, inputs to the algorithm are weights of the base classifier selected for tuning.

Since the base classifiers have one input, two hidden and one output layer, matrix $W_{1,2}$ contains weights of connections between input layer and first hidden layer. In the same way, $W_{2,3}$ contains weights of the connections between first and second hidden layer. Finally, $W_{3,4}$ have weights of the connections between second hidden layer and output layer. Therefore, Size of each individual is also determined by equation (15). As it is possible to assign more than one classifier for detecting each class of input data, variable 'm' is defined to show the number of classifiers per class. On the other hand, the algorithm tunes weights of one classifier in each call. Therefore, for each class of input data, the number of calls of the algorithm and the number of classifiers for adjusting is 'm'.

3 Experiments and results analysis

In this section, accuracy of the new classifier fusion method based on special diversity enhancement strategy is examined and compared with some other known methods of classifier fusion. The datasets are selected from UCI repository of machine learning databases [29]. The datasets selected with specification details are listed in Table 2.

Table 2 Specification of the datasets

	# of samples	# of features	# of classes
Iris	150	4	3
Glass	214	9	6
Wine	178	13	3
Wpbc	198	32	2
Ecoli	336	8	8
Liver	345	7	2
Solar	1389	10	12

3.1 Experiment and test procedure

Validation procedure of the proposed method has three basic steps. It should be noted that the number of classifiers used in the new method depends on the number of classes in each dataset. Therefore, in the MCS, the default number for classifier per class is 1. On the other hand, for each classifier, a unique class of data is assigned such that the classifier is responsible for distinguishing the class from others with maximum accuracy.

In step 1, once the dataset is determined, the number of classifiers is set to the number of classes for the selected dataset. Next, three groups of data are defined namely as train, test, and validation data. Using training data, base classifiers are trained. Since the base classifiers are MLP, one of the defined training methods such as back propagation is used. This training process continues until acceptable accuracy for all the training data is achieved. Therefore, at the end of step 1, final weights of the classifier are saved for next step.

As a result, the weights determined at the end of step 1, are used as input to step 2. The selected classifier is assigned to one of the classes of input data. Moreover, the classifier should be tuned such that it can distinguish the class of data assigned to the classifier with higher accuracy than other classes. In this step, validation data are used to validate accuracy of the classifier.

3.1.1 PSO algorithm initialization

The weights determined in the step 1, are used as primary weights of the classifier at the beginning of step 2. In this step, the PSO algorithm is used to tune these initial weights such that accuracy of detecting the assigned class gains the possible maximum while keeping the accuracy of detecting other classes.

First, PSO parameters should be initialized. With population size P and individual length L which is the number of weights of the selected classifier, P copies of the individual is produced to form initial population. To randomly distribute the individuals in the input space, each individual is assigned a velocity vector of size L. Initially, these vectors are assigned randomly. During successive iterations of PSO, the vectors are updated.

3.1.2 Fitness function evaluation

Since each individual in the PSO is formed with all weights of the selected classifier, fitness function for each individual should construct CM for the selected classifier based on both weights of the classifier determined by the individual and the validation data. Then, according

to the equation (19), fitness value of the individual is determined. Therefore, in each call of the fitness function, all validation data should be evaluated by the classifier with the weights of the individual.

Nevertheless, the PSO algorithm computes fitness of all individuals during iterations where velocity matrix should be updated such that new values for individuals are evaluated. Finally, after completion of all iterations, the individual with highest fitness value is selected as final result. Therefore, the weights of the best individual are assigned to the selected classifier which is adjusted for the class of data assigned to.

3.1.3 Test process of MCS

During iterations of step 2, one of the classifiers trained in step 1 is selected for weight tuning. Next, one of the unassigned classes is assigned to the selected classifier. The PSO algorithm is run with the weights of the selected classifier to tune them for detecting data of the assigned class with highest accuracy. The process of step 2 is repeated for all the trained base classifiers. After tuning of each base classifier for one of the classes, step 3 starts. In this step, overall MCS, constructed with the adjusted classifiers, is tested with the defined test data where a combiner is needed to combine outputs of the adjusted base classifiers.

As a result, for better comparison of efficiency of the new method with other classifier fusion methods, different methods are tested with the test data described in Table 2. Next subsection presents more details.

3.2 Experiments and analysis

Based on the procedure described in previous subsection, test is carried out with different known benchmark datasets from UCI repository of machine learning as described in Table 2 [29].

Since the main idea of the new method called DICDEM is to increase total diversity of the MCS, it is worth to measure the diversity before and after tuning process of the algorithm described in Fig. 10. Different measures of diversity are used to show how the diversity increases by the new method. Different diversity measures used here are Correlation ($\boldsymbol{\rho}$), Q_static (Q), Disagreement (D), Double fault (DF), Interrater agreement (k), Entropy (Ent), Measure of difficulty ($\boldsymbol{\theta}$), and Kohavi–Wolpert variance (KW).

It should also be noted that some of these measures are directly related to the diversity such that by increasing the diversity, the value of these measures are increased. On the other hand, some of the measures inversely related to the diversity because by increasing diversity, the value of these measures decreases. Moreover, each measure has special increasing or decreasing rate with special range determined by the equations described in Sect. 1. In order to show direct or inverse relations, symbols $\uparrow$ and $\downarrow$ are used respectively. In Table 3, the results of the experiments done on different datasets before and after adjustment process done by algorithm of Fig. 7 is shown.

By more analyzing the DICDEM method, it should be noted that the primary goal of the method to increase diversity is mainly based on margin theory [30]. The margin for an object is related to the certainty of its classification. The larger the margin of an input data, the more certain is its label. In contrast, narrow margins increase incorrect classification. A small margin will cause instability of the classification label, that is, one input data might be assigned to different classes by two similar classifiers. One of the ways to measure the margin for an input data x is by degree of support of the data. For a dataset with C classes,

Table 3 Different measures of diversity before and after tuning

Dataset	Method							
	$\rho(\downarrow)$	$Q(\downarrow)$	$D(\uparrow)$	DF$(\downarrow)$	$k(\downarrow)$	$Ent(\uparrow)$	$\theta(\downarrow)$	$KW(\uparrow)$
Iris before →	0.4426	0.8915	0.0844	0.0333	0.8720	0.1266	0.04197	0.0281
Iris after →	0.4008	0.8847	0.0888	0.0311	0.8653	0.1333	0.0404	0.0296
Glass before →	0.3760	0.7799	0.1585	0.0694	0.7454	0.2523	0.0608	0.0660
Glass after →	0.3647	0.7645	0.1607	0.0676	0.7443	0.2542	0.0593	0.0670
Wine before →	0.6227	0.9027	0.1872	0.4925	0.6000	0.2808	0.18	0.0624
Wine after →	0.4979	0.7951	0.2471	0.4681	0.5170	0.3707	0.1600	0.0823
Wpbc before→	0.1840	0.8529	0.0404	0.0050	0.8850	0.0808	0.0145	0.0101
WPBC after→	0.1435	0.7725	0.0505	0.0050	0.8562	0.1010	0.0168	0.0126
Ecoli before →	0.6931	0.9649	0.0837	0.1180	0.8506	0.1241	0.0980	0.0366
Ecoli after →	0.6861	0.9634	0.0847	0.1154	0.8501	0.1250	0.0963	0.0370
Liver before →	0.8573	0.9914	0.0521	0.2144	0.8831	0.1043	0.1701	0.0130
Liver after →	0.8440	0.9894	0.0579	0.2173	0.8712	0.1159	0.1716	0.0144
Solar before →	0.6819	0.9442	0.1195	0.1863	0.7907	0.1770	0.1361	0.0497
Solar after →	0.6668	0.9315	0.1304	0.1917	0.7739	0.1882	0.1370	0.0543

$\mu_k(x), k = 1 \ldots C$ is the support value of x to class k. By this definition, the margin for input data x is defined by equation (20), where ω_k is known class label of x [1].

$$m(x) = \mu_k(x) - \max_{j \neq k}\{\mu_j(x)\}, \sum_{j=1}^{C} \mu_j(x) = 1 \tag{20}$$

However, in the new method presented in this article to increase diversity, each base classifier is trained more to distinguish one class of data more than others. The measures used here, namely DCA and OCA, are computed based on the CM. In order to label input data x, if k is the actual class label for x, after tuning each base classifier for one of the input classes, the decision for label of $x$is made based on decision profile (DP) of x. therefore, the label of x is determined by equation (21) for a dataset with C classes.

$$Label(x) = k | DP_{k,k} = \max\left\{DP_{i,i}, i = 1 \ldots C\right\} \tag{21}$$

To increase certainty of the decision, a defined margin should be maximized. Therefore, a new definition for margin is done by equation (22):

$$m(x) = DP_{k,k} - \max\left\{DP_{j,j}\right\}, \quad j \neq k \tag{22}$$

Moreover, the algorithm shown in Fig. 10 for tuning the base classifiers is the main source of increasing the margin defined by equation (22). Therefore, for each dataset, all the methods are tested and compared with the new method after tuning is done by algorithm of Fig. 10 and using equation (21). The results of these experiments are shown in Table 4.

Taking into account the equation (22) it is clearly identified that an expansion in margin is achieved resulting in significance in decision making. In addition, as equation (23) comprises two components namely accuracy and diversity, both components are carefully taken care off.

Table 4 Errors in DICDEM vs. others methods

	DS	DT	NB	BG	ADB	DICDEM
Iris	2.96	2.96	5.19	5.93	5.19	*2.22*
Glass	9.52	9.52	9.05	10.57	7.62	9.52
Wine	0	0	0	0	1.31	1.58
Wpbc	9.47	9.47	6.32	6.32	5.85	*2.34*
Ionosphere	0.63	0.95	0.95	1.9	0.92	0.95
Ecoli	17.27	17.27	18.48	18.79	13.47	*11.78*
Liver	18.30	17.65	18.30	8.30	17.67	*17.32*
Solar	27.28	27.78	23.26	22.92	25.78	*20.49*

$$e_{MCS} = \frac{1}{2}\left(\overline{f} - t\right)^2 = \frac{1}{B}\sum_i \frac{1}{2}\left(f_i - t\right)^2 - \kappa \frac{1}{B}\sum_i \frac{1}{2}\left(f_i - \overline{f}\right)^2 \tag{23}$$

The first term in the right side of this equation is the weighted average error of the base classifiers. The second term is the amount of variability among the base classifiers indicating the diversity among them. The $\kappa \in [0, 1]$ is a scaling coefficient indicating the emphasis on either diversity or error [31].

Basically, the classifier fusion methods used for testing in Table 4 are Dempster Shafer (DS), Decision Template (DT), Bagging (BG), AdaBoost (ADB), and Naïve Bayes (NB). It should also be noted that in the algorithm of Fig. 10, weights of each classifier are adjusted such that minimum decrease in recognition rate of all other classes except the desired class is taken place. Moreover, in train, test and tuning procedures, k-fold cross validation is used for more accurate results with $k = 10$. As a result, by evaluating Table 4 it is obvious that tuning procedure can effectively increase performance of total MCS. Although in some cases, reduction in error rate is not significant in general, however, the results are acceptable even with large datasets.

Moreover, there are some key points about the results obtained in Table 4. In Table 4, the right most columns indicates the error results obtained after tuning weights of the base classifiers in the MCS by the algorithm of Fig. 10. The algorithm tries to change weights of each base classifier based on equation (19) such that each base classifier is tuned for one of the classes of input data by a search strategy. Moreover, AdaBoost results are produced with the same number of base classifiers used for the other methods in the table. It is obvious that by using more base classifiers better results for the method is obtained.

4 Conclusion

In this article, a new method based on enhancing diversity is introduced for classifier fusion. As a general rule, using multiple classifiers in an MCS can be more efficient than a single classifier if the base classifiers used are diverse enough in the input space. Using identical classifiers makes no major improvement in efficiency of the system. On the other hand, diversity and accuracy in their general form are against each other, therefore, increasing diversity, decreases accuracy of the MCS. As a result, both diversity and accuracy of the system should be considered simultaneously.

Our new method not only increases diversity of a MCS, but also monitors accuracy such that no decrease in accuracy with respect to initially trained system is resulted. To do this, each base classifier is assigned a unique class of data such that the classifier is responsible to detect the class of data with highest reachable accuracy. Therefore, having several different base classifiers each one responsible for detecting one class of data, makes the classifiers as diverse as possible while it is possible to have maximum accuracy.

Since base classifiers are neural networks, tuning each one for detecting one of the classes on input data is implemented by manipulating network weights. Besides, a new measure of diversity is also introduced based on the method used for increasing diversity. Further research can be done to evaluate diversity enhancement by using various types of base classifiers.

References

1. Kuncheva LI (2004) Combining pattern classifiers: methods and algorithms. Wiley, Hoboken
2. Partridge D, Griffith N (2002) Multiple classifier systems: software engineered, automatically modular leading to a taxonomic overview. Pattern Anal Appl 5(2):180–188
3. Alexandre LA, Campilho AC, Kamel M (2001) On combining classifiers using sum and product rules. Pattern Recognit Lett 22:1283–1289
4. Kittler J et al (1998) On combining classifiers. IEEE Trans Pattern Anal Mach Intell 20(3):226–239
5. Chen L, Kamel MS (2009) A generalized adaptive ensemble generation and aggregation approach for multiple classifier systems. Pattern Recognit 42:629–644
6. Valdovinos RM, Sɲnchez JS, Barandela R (2005) Dynamic and static weighing in classifier fusion. Pattern recognition and image analysis. Springer-Verlag, Berlin, pp 59–66
7. Dasarathy BV, Sheela BV (1979) Composite classifier system design: concepts and methodology. Proc IEEE 67(5):708–713
8. Hansen LK, Salamon P (1990) Neural network ensembles. IEEE Trans Pattern Anal Mach Intell 12(10):993–1001
9. Turner K, Ghosh J (1996) Error correlation and error reduction in ensemble classifiers. Connect Sci 8(3):385–403
10. Krogh A, Vedelsby J (1995) Neural network ensembles, cross validation and active learning. Adv Neural Inform Process Syst 7:231–238
11. Kuncheva LI, Whitaker CJ (2003) Measures of diversity in classifier ensembles. Mach Learn 51:181–207
12. Kuncheva LI (2003) That elusive diversity in classifier ensembles. In: IbPRIA, S.n., Mallorca, pp 1126–1138
13. Aksela M, Laaksonen J (2006) Using diversity of errors for selecting members of a committee classifier. Pattern Recognit 39:608–623
14. Kohavi R, Wolpert DH (1996) Bias plus variance decomposition for zero–one loss functions. In: 13th international conference on machine learning, Edinburgh, pp 275–283
15. Windeatt T (2005) Diversity measures for multiple classifier system analysis and design. Inform Fusion 6:21–36
16. Geman S, Bienenstock E, Doursat R (1992) Neural networks and the bias/variance dilemma. Neural Comput 4(1):1–58
17. Kong Z, Cai Z (2007) Advances of research in fuzzy integral for classifier's fusion. In: 8th ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing, Qingdao, pp 809–814
18. Freund Y, Schapire RE (1996) Experiments with a new boosting algorithm. In: 13th international conference on machine learning, Edinburgh, pp 48–156
19. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140
20. Sharkey AJC (1999) Combining artificial neural nets: ensemble and modular multi-net systems. Multi-net systems. Springer-Verlag, Berlin, pp 1–30
21. Sharkey AJC et al (2000) The test and select approach to ensemble combination. In: International workshop on multiple classifier systems, LNCS, S.n., Calgiari, pp 30–44
22. Duin RPW, Tax DMJ (2000) Experiments with classifier combining rules. In: International workshop on multiple classifier systems, LNCS, S.n., Calgiari, pp 16–29
23. Partridge D, Yates WB (1996) Engineering multiversion neural-net systems. Neural Comput 8(4): 869–893

24. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
25. Ho TK (1998) The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell 20(8):832–844
26. Rodriguez JJ, Kuncheva LI (2006) Rotation forest: a new classifier ensemble method. IEEE Trans Pattern Anal Mach Intell 28(10):1619–1630
27. Zhang H, Lu J (2009) Creating ensembles of classifiers via fuzzy clustering and deflection. Fuzzy Sets Syst 161(13):1790–1802
28. Tsymbal A, Pechenizkiy M, Cunningham P (2005) Diversity in search strategies for ensemble feature selection. Inform Fusion 1(1):83–98
29. Blake CL, Merz CJ. UCI repository of machine learning databases. Department of Information and Computer Sciences, University of California. Irvine, s.n., 2007, http://www.ics.uci.edu/~mlearn/MLRepository.html
30. Schapire RE et al (1998) Boosting the margin: a new explanation for the effectiveness of voting method. Ann stat 25(5):1651–1686
31. Brown G, Wyatt JL, Tino P (2005) Managing diversity in regression ensembles. J Mach Learn Res 6:1621–1650