Int J Software Informatics, Vol.2, No.2, December 2008, pp. 181–197 E-mail: ijsi@iscas.ac.cn
International Journal of Software and Informatics, ISSN 1673-7288 http://www.ijsi.org
©2008 by Institute of Software, Chinese Academy of Sciences. All rights reserved. Tel: +86-10-62661040

Global and Local (Glocal) Bagging Approach for Classifying Noisy Dataset*

Peng Zhang[1], Zhiwang Zhang[1], Aihua Li[2], and Yong Shi[1,3]

[1](FEDS Research Center, Chinese Academy of Sciences,
Beijing 100190, China, {zhangpeng04, zzw06}@mails.gucas.ac.cn)

[2](Depart. of Manage. Sci. & Eng., Central Univ. of Finance & Economics,
Beijing 100080, China, neu_aihua@yahoo.com.cn)

[3](College of Inform. Sci. & Eng., Univ. of Nebraska at Omaha,
Omaha, NE 68182, USA, yshi@gucas.ac.cn)

Abstract Learning from noisy data is a challenging task for data mining research. In this paper, we argue that for noisy data both global bagging strategy and local bagging strategy suffer from their own inherent disadvantages and thus cannot form accurate prediction models. Consequently, we present a Global and Local Bagging (called Glocal Bagging: GB) approach to tackle this problem. GB assigns weight values to the base classifiers under the consideration that: (1) for each test instance I_x, GB prefers bags close to I_x, which is the nature of the local learning strategy; (2) for base classifiers, GB assigns larger weight values to the ones with higher accuracy on the out-of-bag, which is the nature of the global learning strategy. Combining (1) and (2), GB assign large weight values to the classifiers which are close to the current test instance I_x and have high out-of-bag accuracy. The diversity/accuracy analysis on synthetic datasets shows that GB improves the classifier ensemble's performance by increasing its base classifier's accuracy. Moreover, the bias/variance analysis also shows that GB's accuracy improvement mainly comes from the reduction of the bias error. Experiment results on 25 UCI benchmark datasets show that when the datasets are noisy, GB is superior to other former proposed bagging methods such as the classical bagging, bragging, nice bagging, trimmed bagging and lazy bagging.

Key words: bagging; ensemble learning; sampling

Zhang P, Zhang ZW, Li AH, Shi Y. Global and local (glocal) bagging approach for classifying noisy dataset. *Int J Software Informatics*, **2008, 2(2): 181–197. http://www.ijsi.org/1673-7288/2/181.pdf**

1 Introduction

Learning from noisy data is a realistic issue for data mining research. Under noisy environment, formulating accurate classification functions or rules is often difficult

* This work is supported by a grant from National Natural Science Foundation of China (#70621001, #70531040, #70501030, #10601064, #70472074), National Natural Science Foundation of Beijing #9073020, 973 Project #2004CB720103, Ministry of Science and Technology, China and BHP Billiton Co., Australia.
Corresponding author: Peng Zhang, Email: zhangpeng04@mails.gucas.ac.cn
Manuscript received 15 Oct., 2008; revised 4 Dec., 2008; accepted 20 Dec., 2008; published online 30 Dec., 2008.

because noisy data often deteriorates the classifier's performance by complicating the decision surfaces. Bagging, a well-known classifier ensemble method, can be used to reduce this type of error by combining together multiple base classifiers learnt from different versions of the training examples. The classical bagging method first generates a series of independent bags through bootstrap sampling from the training sample, then it trains a seriers of base classifiers from each bag, finally it predicts the class label of each test instance by using majority voting mechanism. A lot of studies have demonstrated that classical bagging method is powerful to improve the classifier's performance and thus has been widely used. Besides the classical bagging method, many other bagging methods have been proposed to deal with different kinds of data sources, which can be roughly categorzied into two different bagging strategies: global bagging strategy and local bagging strategy.

The global bagging strategy only utilizes training sample to build a classifier ensemble which tries to receive a minimal bias and variance error on the training sample, such that it also can performance well on testing sample which is assumed to share the same distribution with the training sample. Beriman's bagging[1] and its variants such as bragging[2], nice bagging[3], trimmed bagging[4] belong to this category. It is commonly agreed that the performance of the global bagging strategy critically depends on the base classifier's accuracy. For noise free training dataset, each base classifier will have low bias error, thus the global bagging strategy which can reduce the variance error will achieve a good performance. However, if the training sample contains a lot of noise, the base classifiers will have large bias error. Under this situation, although global bagging strategy can decrease the variance error, the high biased base classifiers will hurt its performance. In words, global bagging strategy shows limited effect on noisy dataset.

Taking consideration of the limitation of global bagging strategy, recently people proposed another local bagging strategy which delays the model building procedure until seeing a specific test instance. The idea behind the local bagging strategy is that when building models, we not only utilize the training sample's information, but also the test sample's. Unlike the global bagging strategy which makes great efforts to acquire minimal error on the whole training set, local bagging strategy tries to build a classifier ensemble that has a minimal error on each specific test instance. The most popular local learning method is to use a given number of the neighbors of the test instance to customize base classifiers, such that the classifier ensemble can achieve a minimal error on it. Recently, Zhu[5] proposed a lazy bagging (LB) method which adds some neighbors of the current test instance I_x into each bag to decrease its bias and variance error on I_x. Kotsiantis[6] proposed a local selective voting (LSV) approach that directly builds base classifiers on the k-nearest neighbors, and then only the classifiers that have statistically better accuracy (according to t-test with $p > 0.05$) are chosen as a base classifier in the ensemble. Comparing these two different approaches, we can observe that both LB and LSV aim to customize base classifiers that biased towards the current test instance I_x, such that it can assign the desired class label to I_x; moreover, they are all constructed on the assumption that I_x's nearest neighbors provide the correct information of I_x. In other words, I_x's neighbors should have the same class label with I_x. This may not be true when the dataset is saturated with noisy instances, where I_x is probably surrounded by instances with an opposite label. In conclusion, local bagging strategy is not capable to classify noisy dataset without modification.

Consider a two-class toy dataset in Fig.1, the symbol "." denotes instances belonged to group "-1" while "*" denotes instances belonged to "+1", the circle "o" means a randomly sampled bag (for simplification, we assume the bags are rather small, it may only contains 2 or 3 instances), the classification boundary is a linear one denoted by "/". Figure 1(a) shows that global bagging strategy tries to find the optimal global classification boundary. However, under noisy environment, finding global optimal boundary is very difficult since the base classifiers may have large bias error which deteriorates the classifier ensemble's performance. Figure 1(b) illustrates that to a specific noisy instance N_x denoted by an enlarged red dot in group "+1", by building classifiers on the nearest bags, we could correctly classify N_x. This is because Fig.1(b)'s situation accords with the local learning assumption that Nx's nearest neighbors share the same class label of N_x. In Fig.1(c), we can see that to a normal instance denoted by an enlarged blue "*" in group "+1", local bagging strategy will misclassify it since it is enclosed with noisy instances. Fortunately, such error can be corrected by taking account of the global optimal information, as shown in Fig.1(d) where "Bag1" and "Bag2" are with the same distance to the enlarged blue "*". When we assign more weight values to "Bag2", we could correctly classify it.

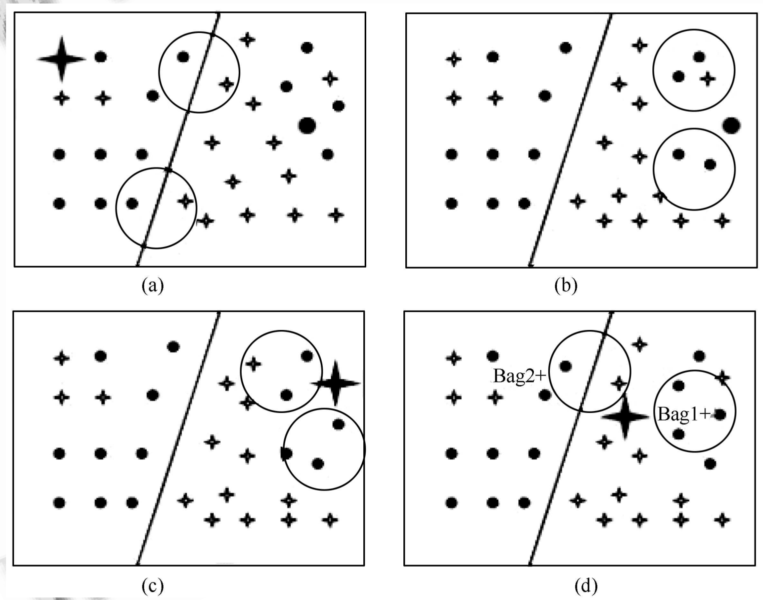

Figure 1. (a) Global optimal bagging tries to find the global optimal classification boundary, which is always difficult due to the noisy instances such as the red enlarged start and circle. (b) Local optimal bagging may help us to correctly classify the noisy instance (denoted by a enlarged red solid circle) if only the neighbors has the same class label; (c) However, if we merely use the neighbors' information, we may misclassify a normal instance which enclosed by noisy instances; (d) To an arbitrary instance, we should consider both the local information and the global information.

In short, under noisy environment, the global bagging strategy tries to build a global unbiased classifier ensemble, which is difficult due to the impact of the noisy training instances (an alternative way is cleaning the training sample before building models, i.e., Yan Zhang's ACE method[7]), while the local bagging strategy which uses the neighbors' information to build base classifiers are too sensitive to the noise. These observations motivate our research on Glocal Bagging (GB) method. The idea behind GB is to combine the advantages of the global bagging strategy and the local bagging strategy. To a specific test instance Ix, GB assigns each base classifier weight value that is in proportion to the base classifier's accuracy on out-of-bags (to get the global information) meanwhile in reversely proportion to the distance between each bag and Ix (to get the local information). By doing so, we expect to increase the prediction accuracy on both normal and noisy instances in test set. We will study why GB works better than the previously proposed bagging methods on noisy dataset by diversity/accuracy analysis and bias/variance analysis.

It should be pointed out that GB neither directly adding the current test instance I_x's neighbors into bags as LB doing, nor using the neighbors to build base classifiers as LSV doing, it calculates the distance between I_x and each bag to avoid the base classifiers trapped into a local optimal solution. Since the performance of GB partly relies on whether the bags near I_x are indeed provide helpful information, we need to calculate the weight values of each attribute such that the weighted distance function can indeed capture informative bags. We will discuss this in Section 2.

The remainder of this paper is organized as follows: in Section 2, we will present the formulation of GB method in detail; in Section 3, we will study the rationale of GB by using both diversity/accuracy analysis and bias/variance analysis on several synthetic datasets with different levels of noise. In Section 4, we will report our experiment resluts on 25 UCI benchmark datasets. In Section 5, we will conclude our paper with discussing the future works.

2 Glocal Bagging

Consider a training set $T_r = \{x_i, y_i\}_{i=1}^n$ and a testing set $T_s = \{x_i, y_i\}_{i=1}^m$, $x_i \in \mathbb{R}^d$ and $y_i \in c$. where c is the set of class labels, GB first builds s independent bags B_i $(i = 1, ..., s)$ and s out-of-bags $S_i = Tr\backslash B_i$ $(i = 1, 2, ..., s)$, then it builds base classifiers using learning algorithm L on each bag B_i and gets $f_i = L(B_i)$. Since many researches have revealed that the unstable classifiers such as decision tree, neural networks probably receive a big improvement when using bagging method, we will use decision tree as GB's learning algorithm. Finally, GB adds weights w_i to each base classifier f_i to formulate a new classifier ensemble $f_E = \Sigma w_i f_i$ to predict the test set Ts. As we discussed above, GB's weights w_i $(i = 1, ..., s)$ mainly come from two parts: the base classifier's accuracy on the corresponding out-of-bag (we denote p below), and the distance between the current test instance I_x to the bags (we denote d below).

To get the first part of w_i's weight, we test each base classifier f_i's accuracy on the corresponding out-of-bag S_i which can be denoted by $p_i = f_i(S_i)$, and then we normalize all the p_i into the range [0,1]. As we said above, w_i will be proportional to p_i.

To get the second part of w_i's weight, we need to calculate the distance between the test instance and the distribution center of each bag. Taking a test instance $I_x \in T_s$ for consideration, to help I_x find the nearest bag, many attributes weighted method, such as Information-gain Ratio (IR)[8] and ReliefF [9] can be employed. Here we use IR method, interested readers can refer to Quinlan[8] for more information about IR. After calculating of the IR value for each attribute, GB normalizes all the IR values into range [0,1], then it uses the Euclidian distance in *Equ.*(1.1) to calculate the distance between I_x to B_i,

$$d(I_x, B_i) = \frac{1}{n}\sum_{k=1}^{n} d(I_x, B_{ik}) = \frac{1}{nr}\sum_{k=1}^{n}\sqrt{\sum_{j=1}^{r} IR'(A_j)(I_x^{A_j} - B_{ik}^{A_j})^2} \qquad (1)$$

where n is the number of instances in B_i, r is the number of attributes, B_{ik} is the k^{th} instance in B_i, A_j denotes the j^{th} attribute, $I_x^{A_j} - B_{ik}^{A_j}$ is the number of value difference on attribute A_j. If A_j is a nominal attribute, $I_x^{A_j} - B_{ik}^{A_j}$ equals 0 iff both I_x and B_{ik} have the same value on A_j. Otherwise it equals 1. w_i is of reverse proportion to d. The whole procedure of GB is shown in Fig.2.

```
Input: a training set $T_r = \{(x_i, y_i)\}_{i=1}^{n}$ , a testing set $T_s = \{(x_i, y_i)\}_{i=1}^{m}$ , $x_i \in R^r$, $y_i \in C$,
learning algorithm $L$, given number of bags $S$
Output: prediction accuracy $Acc$
Begin{Glocal Bagging}
1. Initialize the prediction accuracy $Acc = 0$;
2. Generate $S$ bags $\{B_i\}_{i=1}^{s}$ and out-of-bags $\{S_i\}_{i=1}^{s}$ ;
3. Build models on $B_i$ using $L$, get the base classifiers $f_i$ $(i = 1, , s)$;
4. Calculate each $f_i$'s accuracy on $S_i$ to get the accuracy $p_i$, then normalize $p_i$ to $p'_i = \frac{p_i}{\Sigma_i p_i}$;
5. For each test instance $I_x \in T_r$
    5.1 Calculate the distances $d_i$ between $I_x$ and each $B_i$, and then normalize $d_i$ to $d'_i = \frac{d_i}{\Sigma_i d_i}$ ;
    5.2 Calculate each base classifier's weight $w_i = \frac{p'_i}{d'_i}$, and then normalize $w_i$ to $w'_i = \frac{w_i}{\Sigma_i w_i}$;
    5.3 Construct a classifier ensemble $f_E = \Sigma_{i=1}^{s} w'_i f_i$;
    5.4 if $f_E(I_x) = I_{y_x}$, then $Acc = Acc + 1$ ;
End For;
6. return the prediction accuracy $Acc = Acc/m$;
End Glocal Bagging.
```

Figure 2. The generic framework of Glocal Bagging

3 The Rationale of GB

To investigate why and how GB works, in this section, we will discuss the rationale of GB method. Firstly, we design some synthetic datasets with different levels of noise, and then we explain GB by both diversity/accuracy analysis and bias/variance analysis.

3.1 Synthetic datasets

We generate four synthetic datasets with different levels of noise, i.e., 5%, 10%, 20%, 30% of noise. Each dataset is a 2-dimensional 2-class problem with 5000 instances. All of the generated instances x comply with the Gaussian distribution

$x \sim N(\mu, \Sigma)$, where μ is the mean vector and Σ is the covariate matrix. The classification boundary is defined as a quadric surface $\|x\|^2 = 1$. Instances that satisfy will be assigned a class label "-1" while those satisfy will be assigned a class label "+1". We can see that in these synthetic datasets, instances in each group distinguish each other by its Euclid distance (the smaller the distance, the higher probability to belong to the same group). Although we only generate two groups' classification datasets, it is easy to extend them to multiple groups circumstance.

3.2 Benchmark methods

As we discussed above, all the previously proposed bagging methods can be categorized into two groups: global bagging strategy and local bagging strategy. To investigate whether GB does work, we will compare it with other five bagging methods. Four of them belong to global bagging strategy (classical bagging, bragging, nice bagging, and trimmed bagging) and the remained one belongs to local bagging stratege (Lazy Bagging). Assume we have s bags B_i $(i = 1, ..., s)$, on which we build s base classifiers f_i $(i = 1, ..., s)$, the classical bagging method gives test instance the most frequent y_i as

$$G_{bag}(x) = argmax_{[y\in c]} \sum_{i=1}^{s} f_i(x) \tag{2}$$

where x is the test set. A variant of the classical bagging method is bragging[2], which computes a robust location. More specifically, it use the median location among the s classifiers as

$$G_{brag}(x) = f_{s/2}(x) \tag{3}$$

We can observe that bragging selects a special classifier which locates in the middle of all the generated s classifiers, so the prediction error is equals to bias error (this will be shown in the bias/variance analysis section). The third bagging method is nice bagging[3], which chooses $s'(s' < s)$ base classifiers that has lower error rate than a given value α on the whole training set,

$$G_{nicebag}(x) = argmax_{[y\in c]} \sum_{j=1}^{s'} g_j(x) \tag{4}$$

where $g_j(x) \in \{f_i(x)|Err(f_i(T_r)) < \alpha\}$. The latest proposed global bagging method is Trimmed bagging[4], which chooses the best k base classifiers, k usually sets 75% of the whole s bags.

$$G_{trimmedbag}(x) = argmax_{[y\in c]} \sum_{j=1}^{0.75*s} g_j(x) \tag{5}$$

where $g_j(x) \in \{top \quad 75\% \quad f_i(S_i)\}$. To the local bagging strategy, we choose the most recent lazy bagging method as the benchmark method. Instances in LB's bag are from two parts, the first part is randomly selected from with replacement, the second part is the knn instances (k is usually 3%-5% of the training samples). It is reported that LB, by adding knn instances into each bag, can reduce its base classifier's bias and variance error, such that it is better than many other sample selection methods. For ease of presentation, we use the shorten symbol for each bagging methods in Table 1.

Table 1 Symbols for each bagging method

Symbol	Description
B	Classical Bagging
BR	Bragging
NB	Nice bagging
TB	Trimmed bagging
LB	Lazy bagging
GB	Glocal bagging

3.3 Diversity and accuracy analysis

Lots of research works[11−14] have shown that the performance of a classifier ensemble is decided by its individual base classifier's accuracy and diversity. Thus if we want to improve the performance of a classifier ensemble, we need either to improve its base classifier's accuracy or to improve the diversity between base classifiers. A good classifier ensemble always consists of highly accurate base classifiers (accuracy) which at the same time disagree with each other as much as possible (diversity). But this condition, in fact, is a trade-off between diversity and accuracy. For example, to a specific test instance Ix, if all the base classifiers give it a same correct prediction, then the accuracy of base classifiers are high but the diversity is low. To investigate how diversity and accuracy affect GB's performance, we will discuss them separately:

Diversity is important to ensemble mechanics because even if the performance of base classifiers are not good, we can still acquire an excellent classifier ensemble as long as the diversity is plenty enough. Generally speaking, diversity is increasing with the number of bags. So many bagging methods generate a large number of bags. For example, both the classical bagging and bragging methods need no less than 50 bags, and trimmed bagging uses even more bags. However, due to the trade-off between diversity and accuracy, generating a large number of bags will reduce the base classifier's accuracy, thus deteriorating the classifiers ensemble's performance. What's more, generating a lot of bags will take enormous computation time. That is to say, there is another trade-off between diversity and computational complexity. To tackle these two types of trade-offs, we define the following two concepts to measure the increasing of diversity:

Definition 1. (Diversity Gain): *Given a classifier ensemble f_E and a new coming classifier f, the quantity of diversity that f brings to f_E is defined by $DG_f = E_xE_l(f_E^l(x) - f^l(x))$.* where subscribe l denotes class labels, x denotes test instance, $f_E^l(x)$ denotes the expected value of the previous classifier ensemble on test instance x at label l, $f^l(x)$ denotes the current classifier f's prediction on x at label l, $f_E^l(x) - f^l(x)$ can be calculated by adding up all the discrepancy on each class lable l between classifier f and each base classifier f_i in f_E. The result DG_f is the expectation of the discrepancy of the current classifier f and the previous classifier ensemble f_E.

From this definition, we can observe that the diversity gain describes how many diversity (in quatity) we can acquire if adding a classifier f to a classifier ensemble fE. Besides DGf , the changing tendency of DGf is also useful to describe the diversity. Thus we give a diversity gain rate (DGR) definition as follows:

Definition 2. (Diversity Gain Rate): *Given a classifier ensemble f_E and a new classifier f, the diversity gain rate is defined by $DGR_f = \frac{DG_f}{DG_f + DG_{f_E}}$.*

In Definition 2, calculation of can be regarded as a recursion procedure which adds base classifiers one by one. Figure 3 shows the experimental results of DG, we can see that DG, on all of the four synthetic datasets, increases dramatically when the number of bags is increasing from 0 to 10 (where DG increases from 0 to average 0.21), and then becomes nearly flat (where DG has nearly no improvment from the 10th bag to the 100th bag). In words, after 10 bags, the diversity of DG nearly keeps stable. Figure 4 exhibits the changing tendency of DG. We can see that the changing rate of diversity is a monotonic decreasing function. It tells us that when adding bags to a classifier ensemble, although the diversity receives improvment, the extent of improvement becomes marginal. Combining Figs.3 and 4, we can come to a conclusion that when the number of bags is less than 10, diversity increases

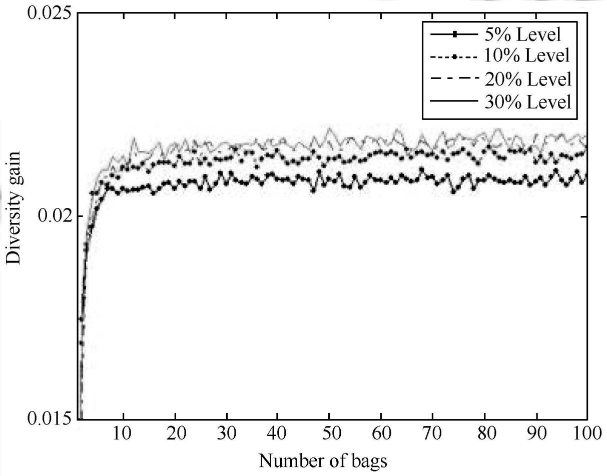

Figure 3. Diversity Gain under different levels of noise. We can see that when the number of bags increases from1 to 10, there is a big improvement of DG. After 10, even if we add more bags, the DG curve is smoothly flat, which means no big improvement of DG.

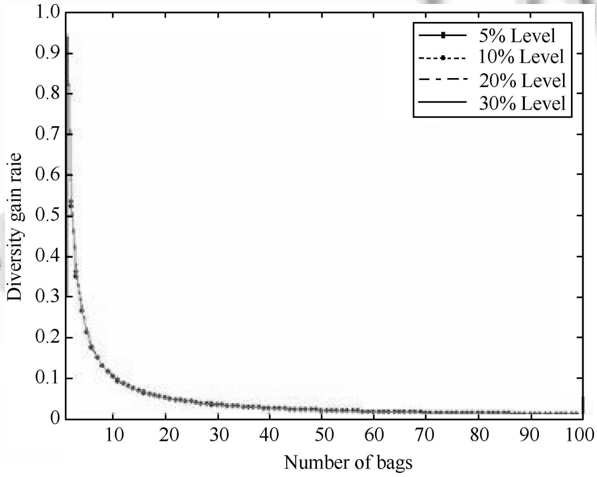

Figure 4. Diversity Gain Rate under different levels of noise. We can observe that the four curves under the four levels of noise are almost overlapped, which tells us the fact that DGR nearly affected by noisy levels. Another important property of DGR is that it is a monotonic

decreasing function. Increasing speed of diversity goes down with the number of bags increasing.

significantly; however, when the number of bags exceeds 10, the increasing speed becomes slower. This also tells us that 10 bags probably is a satisfactory solution of the trade-off between the accuracy, diversity and computational complexity.

Accuracy is another important factor to a classifier ensemble. The prediction accuracy of a classifier ensemble is, in fact, proportional to the comprehensive accuracies of its constituent member. Given the same diversity, whether GB can outperform other bagging methods critically depends on whether it can enhance its base classifier's accuracy. To investigate this, we give the following *Lemma*1 as:

Lemma 1. *Given a diversity d, GB can achieve a more accurate classifier ensemble than the former model averaging mechanics.*

Proof: Assume we have a classifier ensemble f_E which contains s base classifiers f_i $(i = 1, 2, ..., s)$. To a specific test instance I_x, each base classifier has a probability p_i to correctly predict its class label. Then the model average mechanism on I_x can be denoted as $f_{avg}(I_x) = \Sigma_{i=1}^{s} u_i f_i(I_x)$, where each base classifier has the same weight value $u_i = \frac{1}{s}$. To GB, it adds each base classifier a normalized but not equal weight value w_i $(\Sigma_i w_i = 1$), where a much more accurate classifier f_i $(p_m > p_i)$ is assigned heavier weight value. Thus the prediction of GB on I_x is

$$\begin{aligned} f_{CB}(I_x) &= \sum_{i=1}^{s} w_i f_i(I_x) = w_m p_m + \sum_{i\neq m} w_i p_i \\ &= (1 - \sum_{i\neq m} w_i) p_m + \sum_{i\neq m} w_i p_i \\ &= [u_m + \sum_{i\neq m} (u_i - w_i)] p_m + \sum_{i\neq m} w_i p_i \\ &= [u_m p_m + \sum_{i\neq m} u_i p_m \\ &> \sum u_i p_i = f_{avg}(I_x) \end{aligned} \tag{6}$$

So we can get

$$E_x[f_{CB}(x)] > E_x[f_{avg}(x)]. \tag{7}$$

Figure 4 lists the results of the six bagging methods on different noise levels. We can see that GB always performs better than the other five bagging methods. The superiority of GB varies with noise levels. When the noise level is 5%, GB is not significantly superior to the other five methods. This may due to the fact that under as low as 5% noise environment, both global and local bagging strategies perform well. But when adding more noise, neither global bagging strategy nor local bagging strategy performs well, which makes GB much more attractive. In words, GB is superior to other bagging methods under noise environment.

3.4 Bias and variance analysis

The bias and variance analysis is widely used to asset classifier's performance. It is known that the smaller the bias and variance errors, the better the performance. There have been a line of works on bias variance decomposition[16−19]. According to Domingo's decomposition[16], the expected loss of a learner L on a test instance x can be decomposed as noise error $N(x)$, bias error $B(x)$ and variance error $V(x)$:

$$EL(L, x) = c_1 N(x) + B(x) + c_2 V(x) \tag{8}$$

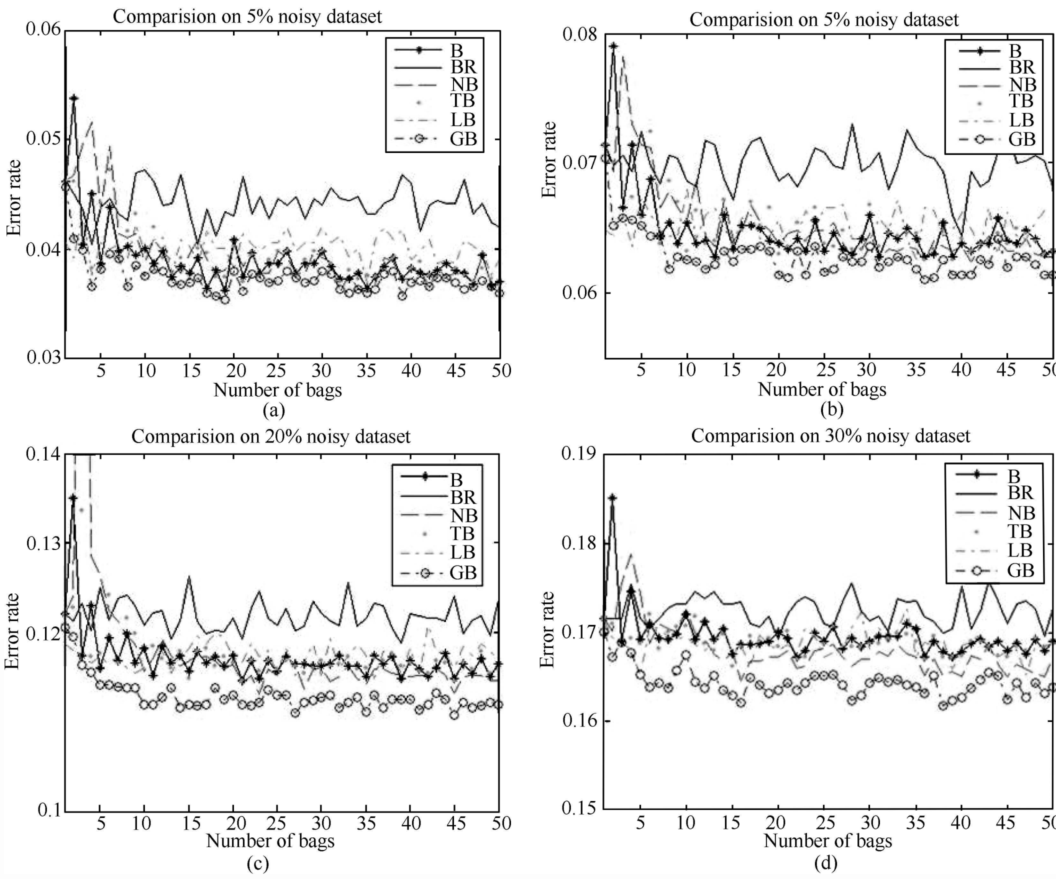

Figure 5. Comparative study on the six different bagging methods, we can observe that GB (depicted in read with circle curve) always has the least prediction error. (a) is the comparison on 5% noise level; (b) is on 10% noise level; (c) is on 20 noise level; (d) is on 30 noise level. The margin between GB and others enlarges with the level of noise increasing. That is to say, compared with others, GB is more professional in classifying noisy dataset.

Before calculating Equ.(8), we would like to introduce two important concepts first: ***optimal prediction*** and ***main prediction***. Consider a two-class (group c1 and group c2) noisy datasets, instance x_i and instance x_j $(i\ j)$ may share the same attribute but having different labels. We define the optimal prediction to be the most frequent observed label t as

$$y_x^* = argmax_{[t \in c]} p(t|x). \tag{9}$$

We then define the main prediction y_x^m to be the most frequent class label that each base learner gives to x as

$$y_x^m = argmax(p(c_1|x), p(c_2|x)). \tag{10}$$

After defining y_x^* and y_x^m, we can calculate the noise error of instance x by counting the number of discrepancies between class label t and y_x^*:

$$N(x) = \sum_t \|t \neq y_x^*\| p(t|x), \tag{11}$$

where $\|z\| = 1$ if z is true, otherwise 0. Then bias error can be calculated by

$$B(x) = \begin{cases} 1, & y_x^m \neq y_x^* \\ 0, & y_x^m = y_x^* \end{cases} \tag{12}$$

The variance error is the discrepancies between each base classifier's prediction and the main prediction, which can be denoted as

$$V(x) = \frac{1}{s}\sum_{i=1}^{s} \|f(x_i) - y_x^m\|. \tag{13}$$

And the expected loss can be calculated as Equ.(14),

$$E(Loss) = E[N(x)] + E_x[B(x)] + E_x[V(x)]. \tag{14}$$

where

$$E[N(x)] = \frac{1}{n}\sum_{i=1}^{n} N(x_i) = \frac{1}{n}\sum_{i=1}^{n}\sum_{t} \|t_{x_i} \neq y_{x_i}^*\| p(t|x_i) \tag{15}$$

$$E_x[B(x)] = \frac{1}{n}\sum_{i=1}^{n} B(x_i) = \frac{1}{n}\sum_{i=1}^{n} |\frac{y_{x_i}^m - t_i}{2}| \tag{16}$$

and

$$E_x[V(x)] = \frac{1}{n}V(x_i) = \frac{1}{ns}\sum j = 1^s \sum_{i=1}^{n} \|y_{x_i}^m \neq f(x_i)\| \tag{17}$$

More information can refer to Domingo[16] and Giorgio Valentini[18]'s work.

Table 2 Results of Bias variance analysis

Measure	Algorithm	5%	10%	20%	30%
Acc	B	0.96	0.9346	0.8817	0.828
	BR	0.9528	0.9314	0.8792	0.8268
	NB	0.9612	0.9344	0.8844	0.8274
	TB	0.9623	0.9368	0.8842	0.8292
	LB	0.9603	0.9320	0.8840	0.8320
	GB	0.9625	0.9373	0.8875	0.8342
BI	B	0.0326	0.0578	0.1117	0.1664
	BR	0.0472	0.0686	0.1208	0.1732
	NB	0.0315	0.0586	0.1086	0.1667
	TB	0.0309	0.056	0.1095	0.1654
	LB	0.0387	0.0668	0.1138	0.1654
	GB	0.0305	0.0553	0.1057	0.1598
VA	B	0.0074	0.0076	0.0066	0.0056
	BR	0	0	0	0
	NB	0.0073	0.007	0.007	0.0059
	TB	0.0068	0.0072	0.0063	0.0054
	LB	0.001	0.0012	0.0022	0.0026
	GB	0.007	0.0074	0.0068	0.006

Table 2 reports the 10-folder cross validation of bias variance decomposition results on the synthetic datasets. The first column lists three measurements including accuracy, bias and variance. The second column shows the comparative algorithms. All of the bagging methods are using 10 bags. The 3th to 6th column list the experiment results. We won't report the noise error $N_(x)$ here because noise error is independent of classifiers, it is only related to the nature of dataset. Since comparisons are on the same synthetic datasets, the noise error will be the same and can be removed. From Table 2, we can observe that, GB always has the highest accuracy and the minimal bias error (we darkle them in Table 2). That is to say, GB achieves much better results than other bagging methods by significantly reducing its bias error.

4 Experiments

4.1 Experimental settings

The whole system is implemented in Java with an integration of WEKA data mining tool[18]. As we mentioned above, we use decision tree (WEKA J4.8 implementation with default parameters) as our base classifier. The objective of these experiments is two folder: (1) comparing GB with B, BR, NB, TB and LB on 25 UCI benchmark datasets[19] to investigate whether GB is better; (2) Ranking all of the bagging methods. To meet the needs of the noisy data sources, we produce different levels of noise by randomly selecting 5%, 10%, 20%, 30% instances from the 25 UCI datasets and assign them an arbitrary label which is not equal to their original label. It should be pointed out that to a nominal attributes, the value will be 1 when their attribute value is the same, otherwise 0.

4.2 Assessment by ranking

To investigate the performance of each bagging method, we rank all of the bagging methods and calculate their number of winning chances and losing chances. Consider a instance x in a test set, if the predicted class label of x is the same with its highest posterior probability, then x is correctly predicted. Accuracy (acc) is defined as the proportion of the number of correctly classified test instances. Furthermore, according to each classifier's accuracy, we rank all of the algorithms in the range of 1 to 6, classifiers with the highest accuracy will be ranked as 1 and the worst classifiers will be ranked as 6. It is should be noticed that there may be more than one classifiers rank 1 (or rank 6) at the same time if these classifiers have the same highest accuracy (or the lowest accuracy). We define other two measurements Winner (#W) and Loser (#L) as follows: if a classifier is ranking 1, then we increase its #W by 1; on the contrary, if ranking 6, we add its #L by 1. Repeating this procedure on the whole 25 UCI data set, we get the Average Ranking (AR), standard deviation of Ranking (SR) and the total number of #W and #L for each bagging methods. A good classifier should have an average rank AR close to 1, more #W, less #L. Meanwhile, if an algorithm is with smaller SR, it is more stable.

4.3 Experimental results

Table 3 reports the experiment results of the comparisons of the six algorithms on different levels of noise. The first column list the name of the used dataset (listed

Table 3 Comparisions on 25 UCI datasets

	B	BR	NB	TB	LB	GB
balance	0.6682	0.6191	0.5546	0.4925	0.6619	0.6352
	0.5440	0.5271	0.6603	0.5882	0.5720	0.4617
	0.7126	0.6470	0.5813	0.5382	0.7350	0.6673
	0.6024	0.5318	0.7476	0.6823	0.6186	0.5419
breast	0.7142	0.6580	0.5911	0.5594	0.7107	0.6451
	0.5676	0.4918	0.7071	0.6483	0.5676	0.5540
	0.7321	0.6709	0.6406	0.5918	0.7178	0.6419
	0.6250	0.5324	0.6964	0.6812	0.6506	0.6189
car	0.9051	0.8998	0.9012	0.912	0.901	0.9241
	0.8105	0.8063	0.8078	0.8221	0.8247	0.8373
	0.7333	0.7294	0.7425	0.7808	0.7526	0.7632
	0.6651	0.6584	0.6758	0.6861	0.675	0.7001
cmc	0.4121	0.4716	0.4189	0.4614	0.4817	0.5054
	0.3481	0.4308	0.3858	0.4067	0.4308	0.4438
	0.3028	0.3852	0.3068	0.3664	0.3977	0.4039
	0.268	0.3643	0.2837	0.3099	0.3282	0.3534
ecoli	0.7882	0.7852	0.7882	0.8117	0.7794	0.8147
	0.6837	0.6729	0.7027	0.7054	0.7162	0.7513
	0.625	0.6	0.6024	0.64	0.565	0.6825
	0.5325	0.4813	0.5465	0.5581	0.486	0.6
glass	0.5952	0.6	0.5999	0.6523	0.5799	0.6952
	0.5391	0.5347	0.5739	0.6043	0.6	0.6782
	0.408	0.516	0.4599	0.472	0.4319	0.548
	0.3444	0.374	0.3703	0.3851	0.3814	0.4666
Halloffame	0.9111	0.9081	0.9051	0.9155	0.897	0.9125
	0.8244	0.8136	0.8299	0.8346	0.821	0.8272
	0.7337	0.7006	0.7412	0.7493	0.6912	0.7619
	0.6298	0.5862	0.6224	0.6614	0.6195	0.7074
hayes	0.6153	0.723	0.6307	0.6692	0.7076	0.6846
	0.6142	0.6142	0.6	0.6357	0.6477	0.6571
	0.5333	0.5666	0.5466	0.5666	0.58	0.6
	0.5117	0.5529	0.4999	0.5526	0.5176	0.5529
ionosphere	0.8828	0.86	0.8971	0.8942	0.8742	0.8857
	0.8472	0.7944	0.8444	0.8555	0.7916	0.8666
	0.7894	0.7526	0.7842	0.8184	0.7473	0.8236
	0.5822	0.5466	0.5577	0.6244	0.511	0.6522
Kr-vs-kp	0.9832	0.9779	0.9838	0.9844	0.9829	0.9838
	0.9008	0.8943	0.9011	0.9019	0.8968	0.9025
	0.824	0.8018	0.8206	0.825	0.8156	0.8261
	0.746	0.7108	0.7373	0.7474	0.7231	0.7506
lymph	0.7333	0.7066	0.76	0.7666	0.7333	0.7666
	0.6251	0.65	0.6312	0.6625	0.6437	0.7
	0.5529	0.5764	0.5235	0.5764	0.5705	0.5882
	0.4631	0.526	0.4578	0.5052	0.4789	0.5263
promoters	0.79	0.72	0.76	0.82	0.74	0.85
	0.6454	0.6545	0.6727	0.6818	0.7	0.7363
	0.525	0.5166	0.6083	0.5916	0.5583	0.5666
	0.4538	0.5	0.4769	0.523	0.523	0.5307
prostate	0.8538	0.8615	0.8384	0.8538	0.8461	0.8538
	0.7533	0.7466	0.7466	0.7466	0.6866	0.76
	0.6	0.575	0.5875	0.6375	0.5687	0.6187
	0.5294	0.5764	0.5352	0.547	0.4647	0.5647
segment	0.9523	0.9377	0.9515	0.9549	0.9532	0.957
	0.8673	0.8393	0.8598	0.874	0.8511	0.8826
	0.7779	0.731	0.7714	0.7866	0.8028	0.7555
	0.6786	0.6289	0.6526	0.696	0.6423	0.719

Table 3(Continued) Comparisions on 25 UCI datasets

Sonar	0.738	0.6523	0.7333	0.7523	0.7476	0.7714
	0.6476	0.6	0.6428	0.6857	0.6095	0.6904
	0.4833	0.5	0.4791	0.5333	0.5708	0.5249
	0.3703	0.4962	0.3925	0.4666	0.4037	0.4555
splice	0.9291	0.9167	0.9273	0.9332	0.9251	0.9366
	0.8376	0.8222	0.8304	0.8432	0.8304	0.8982
	0.7945	0.7647	0.7907	0.8019	0.7756	0.8464
	0.752	0.7159	0.7481	0.7628	0.7659	0.8092
Staheart	0.7185	0.674	0.6962	0.7222	0.7404	0.7407
	0.7285	0.7035	0.6821	0.75	0.7214	0.7607
	0.6161	0.6032	0.6064	0.6612	0.558	0.6387
	0.46	0.5	0.4657	0.4971	0.4457	0.5028
ta	0.5	0.5266	0.4733	0.5333	0.5133	0.5933
	0.4187	0.4062	0.4187	0.4937	0.425	0.5125
	0.3833	0.4333	0.3944	0.3999	0.4333	0.4666
	0.2526	0.3578	0.2842	0.321	0.3789	0.3842
Tic-Tac-toe	0.8864	0.8218	0.8645	0.9052	0.8708	0.9208
	0.7742	0.719	0.7571	0.7971	0.7847	0.8028
	0.6782	0.64	0.6765	0.7147	0.6704	0.7139
	0.5766	0.5693	0.5661	0.6161	0.5959	0.5967
Vehicle	0.667	0.6894	0.6917	0.7023	0.6811	0.7352
	0.5688	0.5698	0.586	0.6172	0.5752	0.6526
	0.4792	0.5217	0.492	0.5336	0.498	0.5801
	0.3927	0.439	0.4181	0.4445	0.4063	0.49
vowal	0.775	0.7369	0.756	0.806	0.8029	0.883
	0.6541	0.6321	0.6376	0.6944	0.6761	0.7853
	0.5457	0.5228	0.5347	0.5983	0.5923	0.6915
	0.4421	0.4531	0.4445	0.4875	0.4765	0.5992
wdbc	0.8762	0.8627	0.8627	0.8813	0.8423	0.8898
	0.8225	0.7887	0.8193	0.8403	0.7822	0.8435
	0.7738	0.7261	0.7461	0.7907	0.7184	0.7923
	0.6594	0.6175	0.6594	0.6932	0.604	0.6783
wine	0.9222	0.9277	0.9	0.9277	0.8833	0.9222
	0.7736	0.7578	0.7947	0.8105	0.7473	0.821
	0.7285	0.6904	0.7142	0.7428	0.5761	0.7571
	0.5869	0.5608	0.5347	0.6217	0.4826	0.6391
yeast	0.4644	0.5033	0.4348	0.5161	0.5865	0.5275
	0.3822	0.4374	0.411	0.4368	0.4453	0.5282
	0.3084	0.3769	0.3179	0.3516	0.3567	0.4393
	0.2367	0.3341	0.2694	0.286	0.3176	0.3808
zoo	0.9	0.9099	0.89	0.91	0.93	0.9202
	0.8181	0.7818	0.8181	0.8363	0.835	0.8736
	0.725	0.7333	0.75	0.75	0.7333	0.7583
	0.623	0.6692	0.6692	0.6923	0.6384	0.7

by alphabet order), the 2th to 7th column list the results with different algorithms. Each cell in the table contains four numeric values which denotes the results on 5%, 10%, 20% and 30% noise levels. The results are averaged on 10-folder cross validation. Since the variances are rather small with little discrepancy, to save space, we won't report them here. We can observe that on the 25 datasets with four different noise levels, GB performs the best on 85 results out of 100. Out of 25 datasets, 12 datasets are with the best accuracy reports on all the four different levels of noise, and we observe 8 weaknesses come from 5% level of noise, which takes the biggest part. This is because under 5% noise level, it is not significant to discriminate a noisy dataset and a normal one, so the global bagging strategy may also perform well. When the noise level increases, the improvement between GB and B is significantly, i.e., on the "car" dataset, when the noise level is 5%, the improvement on GB is 1.9%. When 10% noise, the improvement is 2.68%. When 20%, it achieves 3.0%. And when 30%, it reaches 3.5%. This demonstrates that GB can achieve a better result, especially when noise level is large. Table 4 to Table 7 report the ranking results of the six bagging methods under different levels of noise. As we talked above, we use average ranking, standard deviation of ranking, number of winners and losers as the evaluation methods. In Table 4, we can observe that GB with the highest rank (rank 1.64), the most winners (17 winners) and the least losers (1 losers). TB ranks the second (rank 2.16), with 6 winners, and 1 losers. LB ranks the third (rank 3.8), with 2 winners and 5 losers. BR ranks the last (rank 4.72), with the most losers (10 losers). The whole ranks of the six algorithms under 5% noise level can be concluded as:

$$AR_{GB} < AR_{TB} < AR_{LB} < AR_{B} < AR_{NB} < AR_{BR}. \tag{18}$$

Table 5 lists the results under 10% noise level. We can observe that GB is with the highest rank (rank 1.04), the most winners (17 winners). TB ranks the second, with average rank (rank 2.36) and the least losers (0 losers). LB ranks the third, with average rank 3.80, B ranks after LB, with an average rank 4.16, BR ranks the last, with the most losers (11 losers). The ranks can be orders as:

$$AR_{GB} < AR_{TB} < AR_{LB} < AR_{B} < AR_{NR} < AR_{BR}. \tag{19}$$

In Table 6, we report the results on 20% noise level. As we can see, GB also performs the best on all the four evaluation methods. It has the highest rank (rank 1.48), the least deviation 0.96, the most winners (18 winners) , and the least losers (0 losers). TB, with an average rank 2,12, 4 winners, 0 losers, ranks the second. Following TB is LB, which ranks 3.88, with 2 winners, 6 losers. The ranks of this table can be written as :

$$AR_{GB} < AR_{TB} < AR_{LB} < AR_{B} < AR_{NR} < AR_{BR}. \tag{20}$$

In Table 7, we exhibit the results on 30% noise level, GB is still with the best performance. It has the highest rank (rank 1.29), the least SR (0.62), the most winners (19 winners) and the least losers (0 losers). TB still lists behind GB, with an average rank 2.37, 2 winners and 0 losers. BR lists the third, with an average rank 3.75, 4 winners and 6 losers. NB ranks after BR (rank 4.5), and B ranks the last (rank 4.79), with 0 winners and 10 losers.

$$AR_{GB} < AR_{TB} < AR_{BR} < AR_{LB} < AR_{NR} < AR_{B}. \tag{21}$$

Table 4 Ranking on 5% noise level

	B	BR	NB	TB	LB	GB
AR	3.92	4.72	4.48	2.16	3.8	1.64
SR	0.99	1.48	1.04	1.06	1.68	1.18
#W	0	1	0	6	2	17
#L	1	10	4	1	5	1

Table 5 Ranking on 10% noise level

	B	BR	NB	TB	LB	GB
AR	4.16	5	4.28	2.36	3.8	1.04
SR	1.18	1.19	0.94	0.7	1.52	0.2
#W	0	0	0	1	0	24
#L	4	11	3	0	5	1

Table 6 Ranking on 20% noise level

	B	BR	NB	TB	LB	GB
AR	4.28	4.56	4.12	2.12	3.88	1.48
SR	1.4	1.47	1.19	0.73	1.67	0.96
#W	0	0	1	4	2	18
#L	6	9	2	0	6	0

Table 7 Ranking on 30% noise level

	B	BR	NB	TB	LB	GB
AR	4.79	3.75	4.5	2.37	4.04	1.29
SR	1.25	1.82	0.93	0.82	1.33	0.62
#W	0	4	0	2	0	19
#L	10	6	4	0	3	0

Comparing (21) with (18), (19) and (20), we can find a slight change of orders, LB falls from the 3^{th} to 4^{th}, BR replaces LB, becoming the 3th. B drops to the last rank. This is because when noise level is large, LB probably contains many noisy neighbor instances into each bag, which increases its base classifiers' bias errors and thus reduces its accuracy on prediction. BR, which always finds a robust location, may not increase its bias error as fast as other bagging methods. And due to its minimal variance error, we can observe a big improvement when the noise level is increasing.

4 Conclusions

Under noise environment, the global bagging strategy is not able to receive a good performance because the noise aggravates its base classifier's bias error. On the other hand, the local bagging strategy which uses k nearest neighbor instances to trim bootstrap bags neither can not achieve a good result because the k nearest neighbors may provide a wrong information to classify the test instance due to the noise. In this paper, we present a Global and Local Bagging (GB) method which combines the strength of both global and local bagging strategies. GB adds weight to each base classifier which is both proportional to the base classifier's accuracy on the out-of-bags and reversely proportional to the distance between the bags and the test instance. Experimental results on UCI benchmark dataset have demonstrated that GB performs better than the existing bagging, bragging, nice bagging, trimmed bagging, lazy bagging methods.

Acknowledgements This research has been partially supported by a grant from National Natural Science Foundation of China (#70621001, #70531040, #70501030, #10601064, #70472074), National Natural Science Foundation of Beijing #9073020, 973 Project #2004CB720103, Ministry of Science and Technology, China and BHP Billiton Co., Australia.

References

[1] Breiman L. Bagging predictors. Journal of Machine Learning, 1996, 24: 123–140.

[2] Buhlmann P. Bagging, subbagging and bragging for improving some prediction algorithms. In: Akritas MG, Politis DN, eds. Recent Advances and Trends in Nonparametric Statistics.

Elsevier, Amsterdam, 2003. 9–34.

[3] Skurichina M, Duin B. Bagging for linear classifiers. Pattern Recognition 1998, 31: 909–930.

[4] Christophe Croux, Kristel Joossens, Aurelie Lemmens. Trimmed Bagging. Computational Statistics & Data Analysis, 2007, 52: 362–368.

[5] Zhu XQ. Lazy Bagging for Classifying Imbalanced Data. In: Proc. of IEEE ICDM 2007. 2007. 763–768, 2007.

[6] Sotiris Kotsiantis, Dimitris Kanellopoulos. Local selective voting. In: Proc. of IEEE ICCIT 2007. 1621–1626.

[7] Zhang Y, Zhu XQ, Wu XD, Bond JP. ACE: An aggressive classifier ensemble with error detection, correction, and cleansing. In: Proc. of 17th IEEE ICTAI. 2005. 310–317.

[8] Quinlan JR. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.

[9] Konoeneko I. Estimating attributes, analysis and extension of RELIEF. In: Proc. of the ECML Conference. 1994.

[10] Hanstie L, Salamon P. Neural networks ensembles. IEEE Trans. on Pattern Anal. Mach. Intell., 1990, 12(10): 993–1001.

[11] Optiz DW, Shavlik JW. Generating accurate and diverse members of a neural-network ensemble. Neural Inform. Proc. System, 1996, 8: 535–41.

[12] Dietterich TG. An experimental comparison of three methods for constructing ensembles of decision trees, bagging, boosting and randomization. Mach. Learn., 2000, 40(2): 139–157.

[13] Rodriguez JJ, Kuncheva LI, Alonso CJ. Rotation forest: A new classifier ensemble method. IEEE Trans. on Pattern Anal. Mach. Intell.,2006, 28(10): 1619–1630.

[14] Kong EB, Dietterich TG. Error-Correcting output coding corrects bias and variance. In: Proc. of the 12th ICML Conf. 1996.

[15] Domingos P. A Unified bias-variance decomposition and its applications. In: Proc. of ths Sventeeth Interntional Conf. on Machine Learning. Stanford, CA, Morgan Kaufmann., 2000. 231–238.

[16] Domingos P. A Unifed bias-variance decomposition for zero-one and squared loss. In: Proc. of the 17th National Conference on Artificial Intelligence. Austin, TX, AAAI Press, 2000.

[17] Valentini G, Dietterich TG. Bias-Variance analysis of support vector machines for the development of SVM-based ensemble methods. Journal of Machine Learning Research, 2004, 5: 725–775.

[18] Witten I, Frank E. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, 2005.

[19] UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/.