The Effectiveness of A New Negative Correlation Learning Algorithm for Classification Ensembles

Shuo Wang and Xin Yao
The Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA)
School of Computer Science, University of Birmingham
Edgbaston, Birmingham B15 2TT, UK
Email: {s.wang, x.yao}@cs.bham.ac.uk

***Abstract*—In an earlier paper, we proposed a new negative correlation learning (NCL) algorithm for classification ensembles, called AdaBoost.NC, which has significantly better performance than the standard AdaBoost and other NCL algorithms on many benchmark data sets with low computation cost. In this paper, we give deeper insight into this algorithm from both theoretical and experimental aspects to understand its effectiveness. We explain why AdaBoost.NC can reduce error correlation within the ensemble and improve the classification performance. We also show the role of the amb (penalty) term in the training error. Finally, we examine the effectiveness of AdaBoost.NC by varying two pre-defined parameters – penalty strength λ and ensemble size T. Experiments are carried out on both artificial and real-world data sets, which show that AdaBoost.NC does produce smaller error correlation along with training epochs, and a lower test error comparing to the standard AdaBoost. The optimal λ depends on problem domains and base learners. The performance of AdaBoost.NC becomes stable as T gets larger. It is more effective when T is comparatively small.**

***Keywords*-ensemble learning; classification; negative correlation learning; diversity**

I. Introduction

Since diversity has been recognized as a main reason for the success of an ensemble model, a number of researchers are encouraged to develop negative correlation learning (NCL) algorithms and related theoretical studies. Aiming at better generalization performance, NCL is an ensemble learning technique that takes into account the diversity of the ensemble explicitly during training [1], so as to balance individual errors and the ensemble covariance.

However, the theoretical support of those NCL algorithms is achieved only in the regression context. There is no single theoretical framework or agreed definition of diversity for classification [2]. It is still a challenging problem to introduce diversity explicitly into classification ensembles. Besides, those NCL algorithms are restricted to using base learners that are capable of producing posterior probabilities. So far, only neural networks have been applied in literature. More flexible model is desirable. For the above reasons, we proposed a NCL algorithm for classification ensembles recently, called AdaBoost.NC [3]. A diversity-related term (amb) is decomposed from the 0-1 error function, which is introduced into the AdaBoost training framework as penalty.

By integrating the error correlation information into the weights of the training data, AdaBoost.NC shows very encouraging empirical results in both effectiveness and efficiency aspects. In this paper, we study why it can improve the classification accuracy. Both theoretical and empirical evidences are given. Due to the sequential training procedure of AdaBoost.NC, we consider AdaBoost [4] as the baseline model, from which our analysis starts.

First, we review AdaBoost.NC algorithm and explain several important issues with regard to the algorithm design and implementation. We investigate the role of the penalty term computed by amb in training error, and derive the expression of α_t, the weight of each classifier.

Then, since AdaBoost.NC is claimed to be a "negative correlation" version of AdaBoost, it should be capable of reducing the classifier correlation within the ensemble, or in other words, introducing diversity. We show AdaBoost.NC can produce lower error correlation and better classification boundaries than AdaBoost. We also prove why reducing error correlation in the sequential training context is beneficial to the classification performance theoretically.

Finally, we examine the effectiveness of AdaBoost.NC by varying two pre-defined parameters – penalty strength λ and ensemble size T. Comprehensive experiments are carried out on UCI benchmarks [5].

The rest of the paper is organized as follows: section II describes AdaBoost.NC algorithm. Section III explains why it can reduce error correlation and improve classification performance. Section IV discusses the effectiveness of AdaBoost.NC with respect to the two parameters. Finally, section V makes the conclusions.

II. An NCL Algorithm for Classification Ensembles

To encourage diversity explicitly in classification ensembles, several issues need to be answered: 1) what is the diversity-related term that can penalize high positive correlation among classifier members? 2) How should this term be used in the ensemble training procedure without losing

flexibility and efficiency? 3) Why is our algorithm capable of reducing error correlation and improving classification performance? Our earlier work addressed the first two issues [3], which will be reviewed briefly in this section. A few algorithm-related points that were not specified in [3] will also be clarified here. The third question will be discussed in the next section.

A. Ambiguity Term

Let $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ be a set of training examples, taken by the algorithm as input. Each x_i belongs to an instance space X, and each label y_i is in a finite label space Y. This paper considers binary classification, so we assume $Y = \{-1, +1\}$. The base learning algorithm is called repeatedly with T rounds ($t = 1, 2, \ldots, T$) to construct an ensemble model. A base classifier f_t is built on round t with the form: $X \rightarrow R$. $h_t(x)$ is defined as the sign of $f_t(x)$, which is the predicted label (-1 or +1) to be assigned to example x by the t-th classifier. The final hypothesis $H(x)$ is the sign of the weighted majority vote of the T base classifiers: $H = sign\left(\sum_{t=1}^{T} \alpha_t f_t\right)$, where α_t is the weight assigned to f_t, and $\sum_{t=1}^{T} \alpha_t = 1$.

Given the idea of existing NCL algorithms proposed in the regression context, we were looking for an "ambiguity" term (amb) that can measure the disagreement among the individual classifiers as penalty p_t in the classification context. Based on the 0-1 error function, Eq.(1) is obtained [3] [6]. For any input x,

$$amb = \frac{1}{2} y \sum_{t=1}^{T} \alpha_t (H - h_t). \tag{1}$$

Similar to the ambiguity for regression ensembles [7], Eq.(1) measures the average difference between the ensemble and individuals. However, it is not an "absolute" difference. It also depends on the real label y. In other words, the sign of amb is related to the classification accuracy h_t.

B. AdaBoost.NC

AdaBoost.NC was proposed as a NCL algorithm for classification ensembles [3] based on the AdaBoost training framework. To introduce the amb term into an ensemble training procedure, AdaBoost is selected with the following advantages. First, the sequential training mechanism provides a chance to consider both diversity and accuracy at the same time. Second, it is independent of the choice of base learners. Thus, flexibility is guaranteed. Any classifier can be applied according to the solving problems. Third, diversity information measured by amb is introduced into the weight-updating rule of AdaBoost, so that there is no need to modify training examples themselves.

From another viewpoint, it is reasonable and meaningful to improve the original AdaBoost algorithm, in that degradation of the generalization performance has been reported [8] [9] empirically. In some cases, the weight vectors can become very skewed, which may lead to undesirable bias towards some limited groups of data. Freund and Schapire attribute it to overfitting [10]. They restrict training epoch T to keep final hypothesis simple and achieve lower generalization error. Some related studies also find that AdaBoost can produce diverse ensemble at the first few training epochs, but diversity drops as more classifiers are added in. So, they suggest that it could be beneficial to stop training progress early [11].

Theoretically, Schapire et al. derive an upper bound on the generalization error of AdaBoost [12]. They prove that AdaBoost is aggressive at increasing margins of the training examples, even after the training error reaches zero, which effects a reduction of this bound. However, this bound is rather weak [13]. Breiman finds that better margin distribution does not lead to a lower test error necessarily. If they try too hard to make the margins larger, then overfitting sets in [14]. Murua presents an improved error bound for linear classifier combination by introducing "mutual weak dependence" [15], as an important progress. It is shown that both the low dependence between classifiers and the large expected margins play an important role in achieving low error rates, and there is a trade-off between these two. It provides the evidence that only considering increasing margins is not sufficient when building ensembles. They also suggest looking for a training procedure that can keep the dependence between the classifiers low with large margins.

AdaBoost.NC works in the way for this purpose. Both theoretical and empirical studies have provided enough reasons for AdaBoost.NC to give good performance. Table I presents the pseudo-code of AdaBoost.NC.

Table I: AdaBoost.NC algorithm for binary classification

Given training data set $\{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_m, y_m)\}$, initialize data weights $D_1(x_i) = 1/m$; penalty term $p_1(x_i) = 1$.

For training epoch $t = 1, 2, \ldots, T$:
Step 1. Train weak classifier f_t using distribution D_t.
Step 2. Get weak classifier f_t: $X \rightarrow R$.
Step 3. Calculate the penalty value for every example x_i:
$p_t(x_i) = 1 - |amb_t(x_i)|$.
Step 4. Calculate f_t's weight α_t by error and penalty.
Step 5. Update data weights D_t and obtain new weights D_{t+1} by error and penalty:
$D_{t+1}(x_i) = \frac{(p_t(x_i))^{\lambda} D_t(x_i) exp(-\alpha_t f_t(x_i) y_i)}{Z_t}$,
where Z_t is a normalization factor.

Output the final ensemble:
$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t f_t(x)\right)$.

By introducing the penalty term p_t computed by amb term into the weight updating rule, the main effect is the misclassified examples by the current classifier that receive more same votes from all built individual classifiers get a larger weight increase; the correctly classified examples that get

more disagreement opinions have a larger weight decrease. Therefore, both accuracy and diversity are considered during training.

It is worth noting that the penalty term for each training example uses the magnitude of amb in step 3, for the reason that the sign of amb actually depends on the classification accuracy of the base classifiers as we have explained. Within the weight updating rule in step 5, the accuracy is already considered by the exponential term. There is no point to penalize it repeatedly. In addition, using the absolute value to calculate the penalty term leads to consistently better models than applying amb itself through some preliminary experiments [3]. For the same consideration about the choice of α_t in Eq.(1), uniform weights ($\frac{1}{t}$ on round t) are simply assigned to compute amb. Handling it in this way can also avoid the cyclic computing dilemma in AdaBoost.NC between steps 3 and 4.

$$p_t = 1 - \frac{1}{2t}\left|\sum_{j=1}^{t}\left(H - h_j\right)\right|. \tag{2}$$

It ranges in $[0.5, 1]$. The example weights are updated in step 5. λ is a pre-defined parameter, to control the strength of the penalty term. It is not necessary to range within $[0, 1]$. In many cases, a comparatively large λ is required to achieve better performance [3]. More discussion about the parameter settings will be given in Section IV.

C. Training Error of AdaBoost.NC and the Choice of α_t

Schapire et al. gave an upper bound on the training error of the final ensemble for the generalized version of AdaBoost [16]. Following the notations in Table I, the training error of $H(x)$ produced by AdaBoost is

$$\frac{1}{m}\left|\left\{i : H\left(x_i\right) \neq y_i\right\}\right| \leq \prod_{t=1}^{T} Z_t.$$

Z_t is the normalization factor of updating examples' weights in each round. In order to minimize training error, AdaBoost greedily minimizes this bound sequentially. It also gives the idea for the choice of α_t, the weight of each classifier. Analogously, we induce the upper bound on the training error for AdaBoost.NC by applying the same inference method.

Theorem 1. *The overall training error of AdaBoost.NC is bounded by*

$$\frac{1}{m}\left|\left\{i : H\left(x_i\right) \neq y_i\right\}\right| \leq \left(\prod_{t=1}^{T} Z_t\right) \sum_i \frac{D_{T+1}\left(x_i\right)}{\prod_{t=1}^{T}\left(p_t\left(x_i\right)\right)^{\lambda}}$$

The learning objective of AdaBoost.NC is to minimize the training error of the ensemble, and meanwhile, maximize the ensemble diversity over the training examples. Encouraging diversity is equivalent to reducing p_t. Besides, p_t is always smaller than 1. From theorem 1, therefore, the role of p_t in the training error is to delay the convergence of the training error to some extent and make the algorithm balance diversity and the overall accuracy.

To bound the overall training error by minimizing Z_t [17], the weight of the t-th classifier α_t in step 4 is uniquely selected as

$$\alpha_t = \frac{1}{2}\log\frac{\sum_i D_t\left(x_i\right)\left(p_t\left(x_i\right)\right)^{\lambda}\left(1 + f_t\left(x_i\right) y_i\right)}{\sum_i D_t\left(x_i\right)\left(p_t\left(x_i\right)\right)^{\lambda}\left(1 - f_t\left(x_i\right) y_i\right)}. \tag{3}$$

III. Why Can AdaBoost.NC Reduce Error Correlation and Improve Classification Performance?

So far, we have explained why we proposed AdaBoost.NC and how it works in details. It shows effectiveness and efficiency on many benchmark data sets, comparing to AdaBoost and some other NCL algorithms [3]. In this section, we give some theoretical understandings and more empirical evidence, to demonstrate that AdaBoost.NC can reduce the error correlation within an ensemble and bring performance improvement. Because AdaBoost.NC utilizes AdaBoost training framework, our analysis here focuses on the sequential training context with a majority voting combination rule.

A. Error Correlation and Classification Accuracy

Before our discussion, we use the conventional probability formalism to reformulate our problem first [18] [19]. For any single data point (x, y), y belongs to the discrete and finite target space, and we assume that the target is noise-free. Ensemble output H combines the discrete outputs from the classifiers $\{h_1, \ldots h_T\}$ with equal weights assigned. Let $p\left(h_i\left(x\right) = y\right)$ denote the average probability that h_i makes guess y for input x in response to the training subset S_i. When the context is clear, $p\left(h_i\right)$ will be used instead. It is also the accuracy for a given x. Let $p\left(\overline{h_i}\right) = 1 - p\left(h_i\right)$. Since our focus is on the error correlation among the individuals, we define the correlation between any h_i and h_j in Eq.(4), which has a very straightforward form [20], as

$$\begin{aligned} cov\left(h_i, h_j\right) &= p\left(h_i = y, h_j = y\right) - p\left(h_i = y\right) p\left(h_j = y\right) \\ &= cov\left(\overline{h_i}, \overline{h_j}\right) \end{aligned} \tag{4}$$

The first term presents the probability of making the same decision. The second term is the value obtained in a statistically independent manner. Apparently, a negative cov implies negatively correlated errors. The analysis in [20] has demonstrated empirically that it is better for the individuals to make errors in a negatively correlated way rather than just in an independent way by discussing $cov\left(h_i, h_j\right)$.

Using the above formulation, let's put our attention back to the sequential training context with a majority voting

combination rule. We assume that each h_t is only correlated with the previous one h_{t-1}, to simplify the problem. It is a reasonable assumption for AdaBoost training framework, because the weights of examples are updated by only considering the accuracy of the current classifier, rather than the accuracy given by the whole ensemble. In other words, for any $i \neq t-1\,(i<t)$, we have $cov\,(h_t, h_i) = 0$. In fact, h_t and h_i may have indirect correlation which is carried through h_{t-1}, but it is supposed that this indirect correlation is much weaker than $cov\,(h_t, h_{t-1})$. For binary classification, a theorem is achieved to describe the accuracy of the current ensemble H^t consisting of the first t base classifiers.

Theorem 2. *When building the t-th classifier h_t, reducing the correlation between h_t and H^{t-1} can increase the probability of H^t producing the correct result if the accuracy of h_t does not drop, where H^q denotes the combined output given by the first q classifiers.*

Proof: Consider a voting combination with uniform weight assignment to each classifier. If H^t needs at least n correct votes from the t classifiers to produce the right decision, then

$$p\left(H^t\right) = p\left(H^{t-1}\right) + \sum^{C_{t-1}^{n-1}} \left[p\left(h_t = y, H^{\left(\frac{n-1}{t-1}\right)} = y\right) - p\left(h_t = y, H^{\left(\frac{n-1}{t-1}\right)} = y, H^{t-1} = y\right)\right].$$

This equation considers two situations. In order to let H^t make the right decision, one possibility is that the first $(t-1)$ classifiers already contain at least n correct votes. The other possibility is that the first $(t-1)$ classifiers have exactly $(n-1)$ correct votes and the t-th one provides the last correct vote. $p\left(H^{\left(\frac{n-1}{t-1}\right)} = y\right)$ denotes the probability of all $(n-1)$ classifiers producing the correct label, which are randomly chosen from the first $(t-1)$ ones.

Now, we only discuss the second term that contains h_t. According to cov's definition, we obtain

$$\begin{aligned} & p\left(h_t, H^{\left(\frac{n-1}{t-1}\right)}\right) - p\left(h_t, H^{\left(\frac{n-1}{t-1}\right)}, H^{t-1}\right) \\ =& p\,(h_t)\left[p\left(H^{\left(\frac{n-1}{t-1}\right)}\right) - p\left(H^{\left(\frac{n-1}{t-1}\right)}, H^{t-1}\right)\right] + cov\left(h_t, H^{\left(\frac{n-1}{t-1}\right)}\right) \\ & - cov\left(h_t, H^{\left(\frac{n-1}{t-1}\right)} H^{t-1}\right). \end{aligned}$$

Now by applying the assumption made on the error correlation, it can be rewritten into two possible cases,

$$\begin{aligned} & \text{if } h_{t-1} \notin H^{\left(\frac{n-1}{t-1}\right)}: \\ & p\,(h_t)\left[p\left(H^{\left(\frac{n-1}{t-1}\right)}\right) - p\left(H^{\left(\frac{n-1}{t-1}\right)}, H^{t-1}\right)\right] - cov\left(h_t, H^{t-1}\right); \\ & \text{if } h_{t-1} \in H^{\left(\frac{n-1}{t-1}\right)}: \\ & p\,(h_t)\left[p\left(H^{\left(\frac{n-1}{t-1}\right)}\right) - p\left(H^{\left(\frac{n-1}{t-1}\right)}, H^{t-1}\right)\right] + cov\,(h_t, h_{t-1}) - \\ & cov\left(h_t, h_{t-1} H^{t-1}\right). \end{aligned}$$

Due to the positive property of $p\left(H^{\left(\frac{n-1}{t-1}\right)}\right) - p\left(H^{\left(\frac{n-1}{t-1}\right)}, H^{t-1}\right)$, $p\,(h_t)$ should be kept as high as possible. Meanwhile, it is also negatively related to $cov\left(h_t, H^{t-1}\right)$ and $cov\left(h_t, h_{t-1} H^{t-1}\right)$. Reducing the correlation between h_t and H^{t-1} can boost the role of h_t in the current ensemble and improve the ensemble accuracy. This theorem suggests a trade-off between individual accuracies and the correlation among them.

■

Recall the training strategy of AdaBoost.NC. Intuitively, the training examples that receive all correct/wrong votes from the first $(t-1)$ classifiers must have higher $cov\left(h_t, H^{t-1}\right)$ than the ones that get partially correct votes. AdaBoost.NC emphasizes this group of data by using the penalty term without losing the accuracy of each classifier, because the examples get the attention back immediately if they are misclassified by the previous classifier rather than the whole ensemble. Each classifier plays a better role in AdaBoost.NC ensemble.

B. Empirical Evidence for AdaBoost.NC

To further show the effect of AdaBoost.NC on the error correlation and classification accuracy during the ensemble training, we apply AdaBoost.NC on a two-dimensional artificial data set, which comes from two Gaussian distributions with equal covariance and a small overlapping area close to the separating line. Each class contains 200 training examples. Using the same generation method and settings, we have a separate testing file, in which each class has 50 points.

In the experiments, we build 51 C4.5 decision trees composing one ensemble model. Penalty strength λ is set to 0, 2, and 9 respectively. When $\lambda = 0$, AdaBoost.NC is reduced to the standard AdaBoost. Every model is repeated 30 times. At each step of building a new classifier, we output the average $cov\left(h_t, H^{t-1}\right)$ value and error rate on the testing file. In Fig. 1, x-axis presents the training epoch increasing from 2 to 51; y-axis indicates the correlation and test error rates respectively.

We can see from Fig. 1 that correlation values at $\lambda = 9$ are generally smaller than values at $\lambda = 0$, especially during the first few training iterations. Correspondingly, it has the best generalization performance. This observation agrees with Theorem 2, and shows the effectiveness of AdaBoost.NC.

In [20], the authors mentioned an error rearrangement procedure when discussing how negatively correlated errors can be beneficial to an ensemble's performance. Basically, their analysis claimed that, if a classifier has to make an error, then make the error on a different example. Sorting errors in the negatively dependent way can minimize the number of ensemble errors. AdaBoost.NC can be viewed

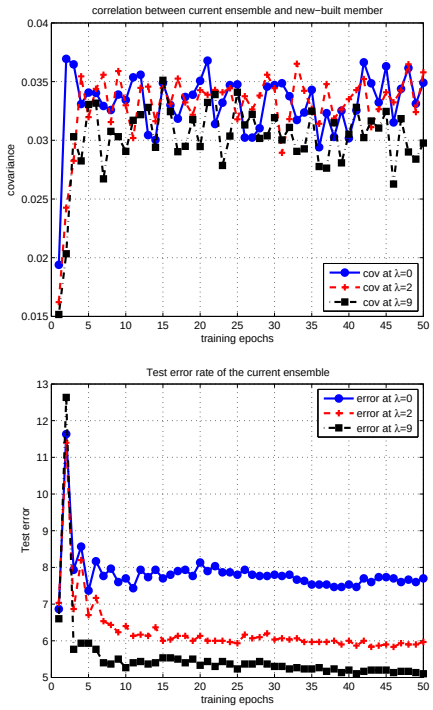


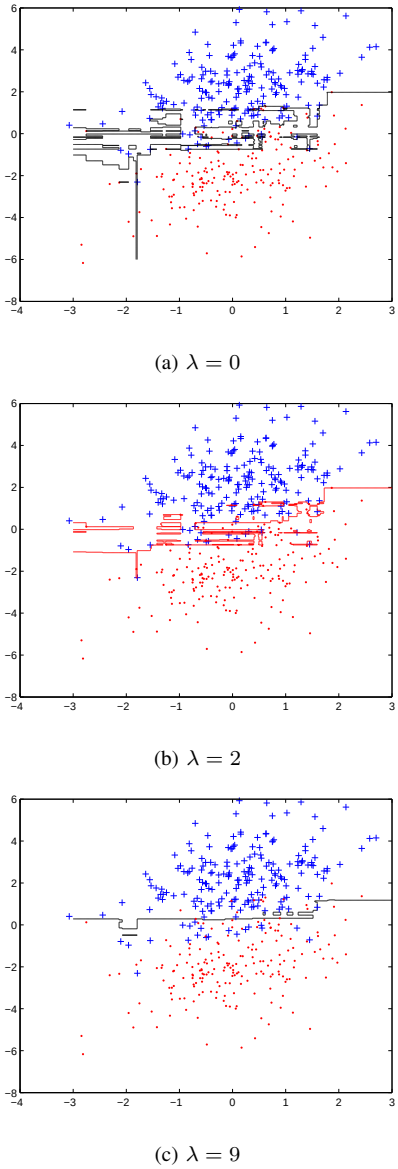
Figure 1: Correlation and test error during ensemble training

as an automatic error rearrangement procedure. The least ambiguous and correctly classified training examples are boosted, which gives the chance for the individual classifier to make different errors instead of the same ones. From the above discussions, we can see that AdaBoost.NC is a learning algorithm that encourages both of the low error rate and negatively correlated errors.

Next, we examine the classification boundaries produced by AdaBoost.NC and see how this improvement happens graphically. We continue the above experiments on the same artificial data. Fig. 2 compares the classification boundaries with the same three λ values.

As λ sets higher, AdaBoost.NC produces a smoother boundary and less overfits the overlapping points. On this artificial data, the most difficult points are distributed in the overlapping area. When $\lambda = 0$, the boundary includes a lot of small squares and a spiky region, because the algorithm keeps boosting this part of examples. Although they are more likely to be chosen for training, they are still hard to be separated. When $\lambda = 2$, the situation is relieved to some extent. Fewer squares are produced. When $\lambda = 9$, more "easier" examples are selected to help train every classifier, which results in a better boundary further.

As we expect, AdaBoost.NC is a simple and effective ensemble algorithm, which not only emphasizes misclassified examples, but also makes use of easy examples to reduce the chance of making errors on the same examples. Therefore, both accuracy and diversity are guaranteed.



(a) $\lambda = 0$

(b) $\lambda = 2$

(c) $\lambda = 9$

Figure 2: Classification boundaries of AdaBoost.NC at $\lambda = \{0, 2, 9\}$

IV. The Effectiveness of AdaBoost.NC

We have explained the learning mechanism of AdaBoost.NC and understood why it can bring performance improvement from both theoretical and experimental standpoints. In this section, we will investigate the impact of two pre-defined parameters involved in AdaBoost.NC on its effectiveness, which are penalty coefficient λ and ensemble size T.

We conduct our experiments on ten real-world benchmarks with two classes from UCI data repository [5]. Data

information is summarized in Table II. In the experiments, we adjust λ in $[0, 12]$, and T in $[5, 201]$. When λ equals to 0, AdaBoost.NC is reduced to the standard AdaBoost. We randomly sample 80% of the data as the training set, if no testing file is provided. The rest are for the test. Each algorithm is run 30 times at every parameter setting, and averages are computed as the points in the following figures.

Table II: Summary of classification data sets

Name	Train	Test	Attributes	Classes
promoter	84	22	57	2
sonar	166	42	60	2
ionosphere	280	71	34	2
house-votes-84	348	87	15	2
crx	489	201	15	2
breast-w	559	140	9	2
pima	614	154	8	2
german	800	200	20	2
hypothyroid	2530	633	25	2
insurance	5822	4000	85	2

When λ is varying, we fix $T = 9$. C4.5 decision trees and MLP networks are chosen as base learners. Fig.3 presents the results ordered by the size of the problem. In each plot, the first point on the curve indicates the number of misclassified examples produced by AdaBoost. The following points are the mean numbers of errors made by AdaBoost.NC with respect to λ. If any of those points is significantly lower than the first, then a performance improvement over AdaBoost happens.

In Fig. 3, we have the following observations: 1) AdaBoost.NC with a proper λ can generally improve classification performance for both tree and MLP cases, especially on the last six data sets with larger sizes. However, the optimal λ is domain-dependent. For example, the best λ is around 4 for data set "crx"; the error keeps decreasing even when λ reaches 12 for "insurance". 2) The optimal λ is also related to the base learner. For the example of "german", when C4.5 is applied, the lowest error is achieved at $\lambda = 2$, but the best $\lambda = 9$ in the case of MLP. 3) MLP is less sensitive to the change of λ. As λ gets large, tree ensembles tend to degrade the performance, such as "promoter", "house-votes-84", and "breast-w". Generally speaking, AdaBoot.NC with the optimal λ leads to higher classification accuracies, but the best λ depends on problem domains and base learners. The effectiveness of AdaBoost.NC tends to be weakened by large λ, when decision trees are used as the base learner. An unclear issue here is when a large λ can cause performance degradation. The reasons why the sensitivity of AdaBoost.NC to the parameter setting differs among problem domains and base learners will be explored in our future work.

Now, let us see the performance tendencies by varying the ensemble size T within $[5, 201]$. Each plot in Fig. 4 depicts two curves produced by AdaBoost and AdaBoost.NC with C4.5 as the base learner. The λ used in AdaBoost.NC is the optimal value obtained from Fig. 3.

In Fig. 4, we have the following observations: 1) the effectiveness of both AdaBoost and AdaBoost.NC is not influenced by increasing T. There is no obvious overfitting phenomenon. 2) The performance of both AdaBoost and AdaBoost.NC becomes stable when T gets 41-51. Significant error reduction rarely occurs after that. 3) AdaBoost.NC is more effective than AdaBoost when T is small, particularly for $T \leq 21$, such as "sonar", "ionosphere", "breast-w", and "german". 4) As T becomes very large, AdaBoost.NC and AdaBoost reach a similar performance level in some cases, such as "crx", "breast-w", and "german". In other cases, however, AdaBoost.NC consistently outperforms AdaBoost, which is not affected by T, such as "pima", "hypothyroid", and "insurance". It is interesting to find out the reasons. In summary, the performance of AdaBoost.NC is not influenced by the ensemble size. It is more effective when T is comparatively small.

V. Conclusions

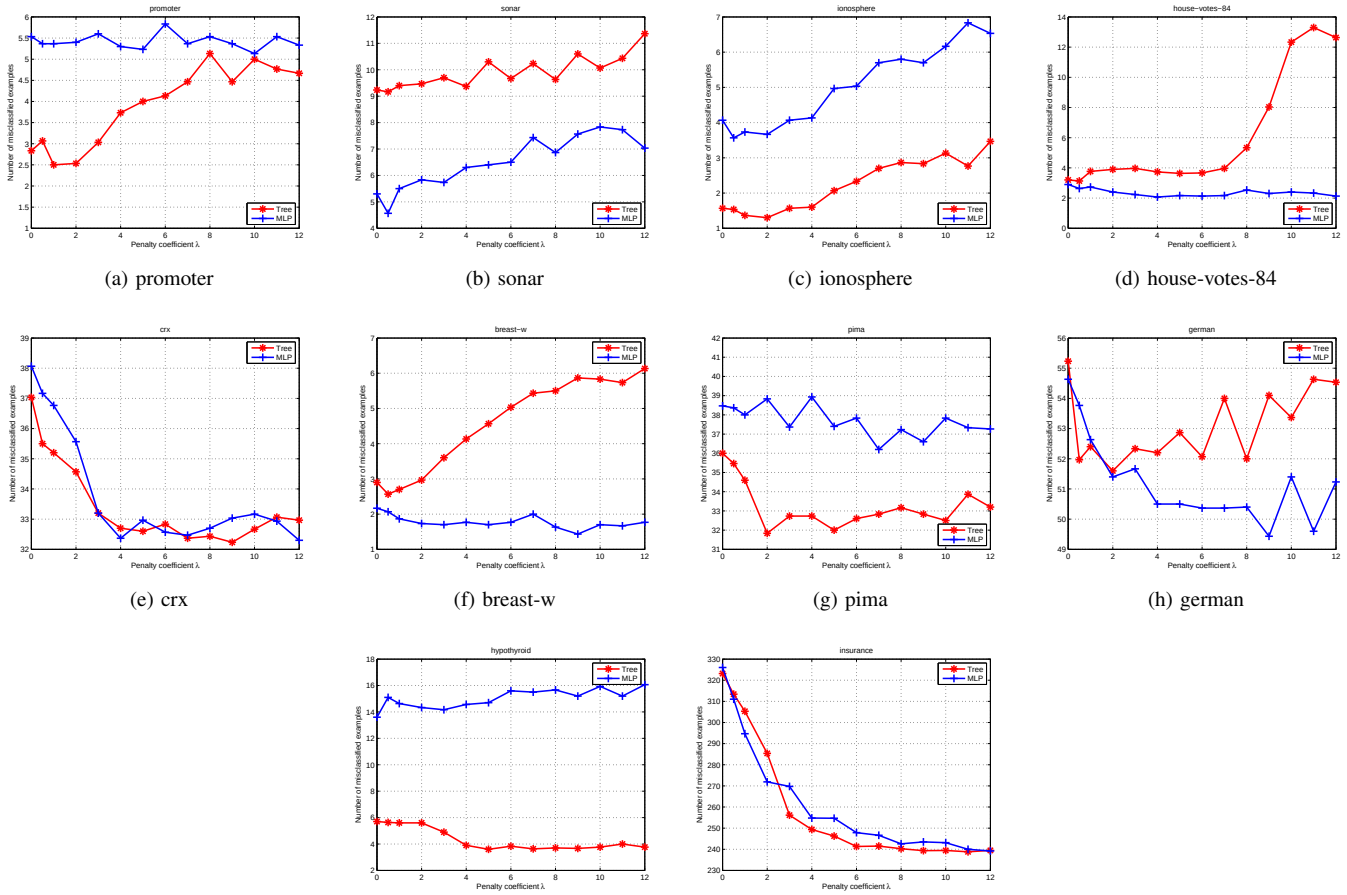
As a further study of negative correlation learning for classification ensembles, this paper gives deeper insight into why AdaBoost.NC algorithm can produce less correlated ensembles and improve classification performance. Both theoretical and empirical evidences are provided, by considering the standard AdaBoost as the baseline model.

Some unspecified points about AdaBoost.NC algorithm are clarified first, including the reasons for using AdaBoost training framework and taking into account the magnitude of amb term as a penalty instead of amb itself. We derive the upper bound on AdaBoost.NC's training error and the expression of computing the weight of each classifier.

Then, we give a theorem to show that decorrelating the current classifier and the current ensemble in a sequential training context with a majority voting combination rule can be beneficial to the classification accuracy. AdaBoost.NC is shown to be capable of reducing this correlation and the test error.

Finally, the effectiveness of AdaBoost.NC is examined on some real-world data sets by varying two parameters in the algorithm – penalty strength λ and ensemble size T. Generally speaking, the optimal λ depends on problem domains and base learners. Large λ may cause performance degradation, especially when decision tree is applied as the base learner. AdaBoost.NC's performance becomes stable as T gets larger. It is more effective when T is comparatively small.

There are still some remaining issues for future studies: 1) how do we decide the range of λ for a given classification problem. 2) What kind of problems is AdaBoost.NC more effective on? For example, it is interesting to explore its performance on class imbalance domains. 3) More investigation



(a) promoter (b) sonar (c) ionosphere (d) house-votes-84

(e) crx (f) breast-w (g) pima (h) german

(i) hypothyroid (j) insurance

Figure 3: Performance of AdaBoost.NC by varying λ. (X-axis: penalty coefficient $\lambda \in [0, 12]$; Y-axis: the number of misclassified examples.)

needs to be done for multi-class problems.

ACKNOWLEDGMENT

This work was partially supported by an Overseas Research Student Award (ORSAS) and a scholarship from the School of Computer Science, University of Birmingham, UK.

REFERENCES

[1] Y. Liu and X. Yao, "Ensemble learning via negative correlation," *Neural Networks*, vol. 12, no. 10, pp. 1399–1404, 1999.

[2] G. Brown, "Ensemble learning," *Encyclopedia of Machine Learning*, pp. 1–24, 2010.

[3] S. Wang, H. Chen, and X. Yao, "Negative correlation learning for classification ensembles," in *International Joint Conference on Neural Networks, WCCI*, 2010, pp. 2893–2900.

[4] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. of the 13th. Int. Conf. on Machine Learning*, 1996, pp. 148–156.

[5] A. Asuncion and D. Newman. (2007) UCI machine learning repository. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

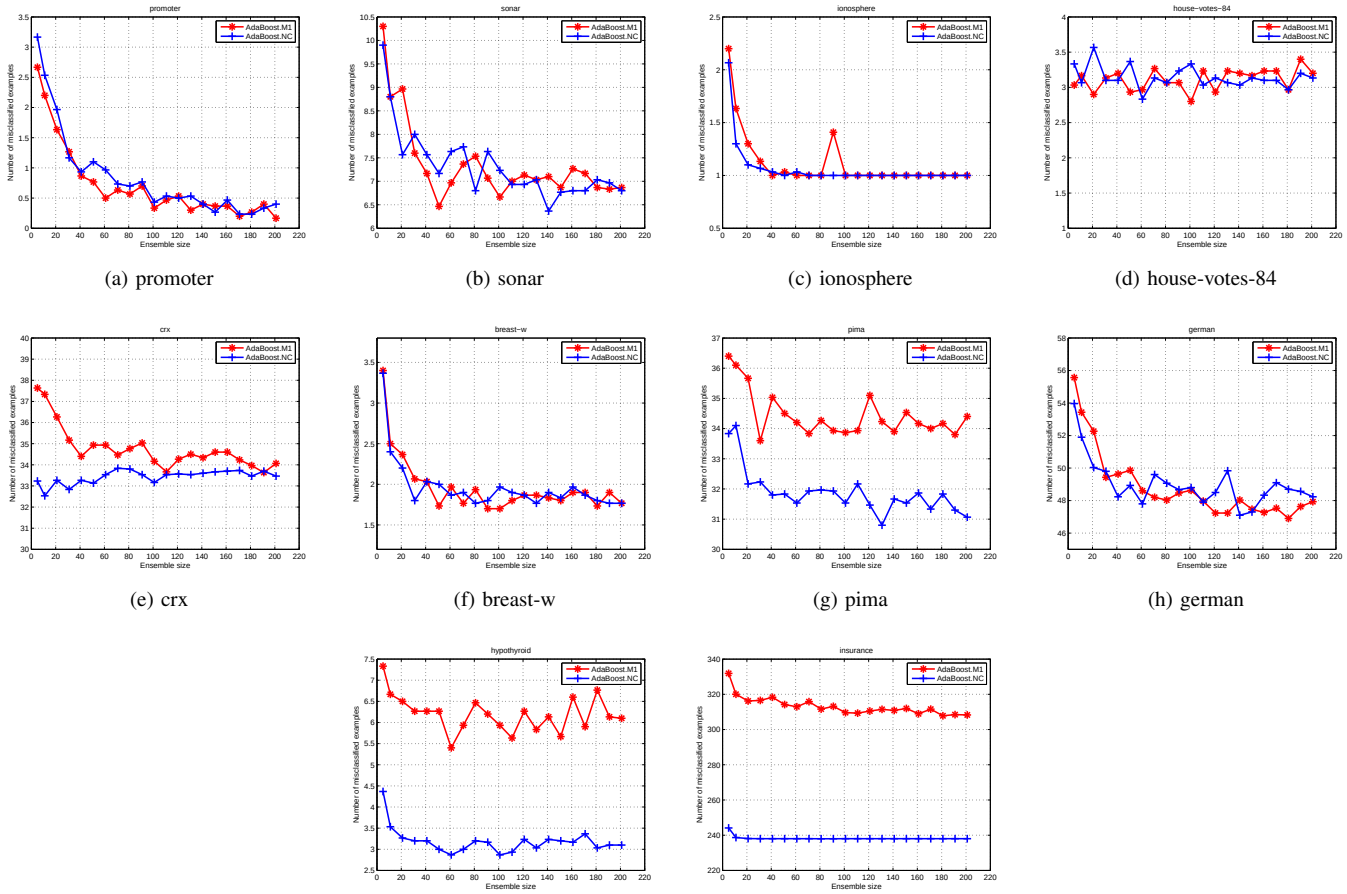
[6] H. Chen, "Diversity and regularization in neural network ensembles (chapter 3)," Ph.D. dissertation, School of Computer Science, University of Birmingham, 2008, Url: http://www.comp.leeds.ac.uk/scshc/.

[7] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Advances in Neural Information Processing Systems*, vol. 7, 1995, pp. 231–238.

[8] J. Quinlan, "Bagging, boosting, and c4.5," in *The 1996 13th National Conference on Artificial Intelligence, AAAI 96*, 1996, pp. 725–730.

[9] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.

[10] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting,"



(a) promoter (b) sonar (c) ionosphere (d) house-votes-84

(e) crx (f) breast-w (g) pima (h) german

(i) hypothyroid (j) insurance

Figure 4: Performance of AdaBoost and AdaBoost.NC by varying T. (X-axis: ensemble size $T \in [5, 201]$; Y-axis: the number of misclassified examples.)

Journal of Computer and System Sciences, vol. 55, pp. 119–139, 1997.

[11] C. A. Shipp and L. I. Kuncheva, "An investigation into how adaboost affects classier diversity," in *IPMU*, 2002, pp. 203–208.

[12] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *The Annals of Statistics*, vol. 26, no. 5, pp. 1651–1686, 1998.

[13] R. E. Schapire, "The boosting approach to machine learning: An overview," in *MSRI Workshop on Nonlinear Estimation and Classification*, 2002, pp. 1–23.

[14] L. Breiman, "Prediction games and arcing algorithms," *Neural Computation*, vol. 11, no. 7, 1999.

[15] A. Murua, "Upper bounds for error rates of linear combinations of classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 591–602, 2002.

[16] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.

[17] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognition*, vol. 40, no. 12, pp. 3358–3378, 2007.

[18] R. Kohavi and D. H. Wolpert, "Bias plus variance decomposition for zero-one loss functions," *Machine Learning: Proceedings of the Thirteenth International Conference*, pp. 275–283, 1996.

[19] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Wiley Press, 2004.

[20] K. M. Ali and M. J. Pazzani, "On the link between error correlation and error reduction in decision tree ensembles," 1995.