A Low Variance Error Boosting Algorithm

Ching-Wei Wang and Andrew Hunter

University of Lincoln, Lincoln LN6 7TS, United Kingdom

Abstract. This paper introduces a robust variant of AdaBoost, *cw-AdaBoost*, that uses weight perturbation to reduce variance error, and is particularly effective when dealing with data sets, such as microarray data, which have large numbers of features and small number of instances. The algorithm is compared with AdaBoost, Arcing and MultiBoost, using twelve gene expression datasets, using 10-fold cross validation. The new algorithm consistently achieves higher classification accuracy over all these datasets. In contrast to other AdaBoost variants, the algorithm is not susceptible to problems when a zero-error base classifier is encountered.

1 Introduction

This paper introduces a modified version of AdaBoost, *cw-AdaBoost*, that uses weight-perturbation to improve performance. In contrast to other AdaBoost variants, the algorithm does not stop or reset weights if a zero-error classifier is produced, which allows it to reduce variance further than alternative algorithms, and makes it more robust than standard boosting algorithms when combined with "unstable" classifiers such as neural networks or decision trees. The algorithm is evaluated on a number of challenging data-sets against several alternative variants, and is shown to have superior performance. The performance is analyzed by considering the bias/variance decomposition of the classification error rate.

A large number of studies have shown the effectiveness of ensemble learning algorithms in improving classifier performance. Breiman [9] introduced the Bagging algorithm, which forms an ensemble by aggregating multiple classifiers, each of which is trained using a bootstrapped training set (randomly sampled with replacement from the training set). This approach is very effective in reducing the variance of the ensemble classifier, and is particularly useful if using "unstable" base classifiers such as decision trees or neural networks [8], that can produce convoluted decision regions which vary heavily according to the selection of the training set. In contrast, Freund's [15] Adaboost ensemble algorithm uses weighted training samples, and the weights are deterministically updated to emphasize misclassified instances from the training set. This allows even a relatively simple base classifier algorithm to adjust for complex decision surfaces, allowing both the bias and the variance to be reduced.

However, Boosting does have some known limitations, including that the deterministic sampling does not necessarily optimize the rate of variance reduction, and issues that occur when zero-error or high error base classifiers are created. The algorithm introduced in this paper addresses these limitations.

1.1 Related Work

The Boosting algorithm is extremely powerful, and consequently has received a great deal of attention from the machine learning community, not least in addressing some of the known limitations. Several authors have attempted to integrate the stochastic element of bagging into a boosting framework. Friedman [17] proposed a stochastic gradient Boosting, which randomly draws sub-samples of the training data (without replacement) at each iteration to train individual base classifiers. The intention of this method is to use the bootstrap sampling approach of Bagging to improve the variance reduction of Boosting. However, it leads to a problem that using smaller sub-samples in training base models causes the variance of the individual base classifiers to increase. Webb [30] proposed a MultiBoost algorithm by combining a modified boosting algorithm with wagging. MultiBoost wraps Boosting inside Bagging, utilizing the continuous Poisson distribution to generate a number of randomly weighted (sampled) data from the original training dataset and then constructs bags of individual ensembles, each of which learns by Boosting from the weighted samples (which in effect provide a randomized weighting start-point for the Boosting algorithm). A detailed analysis is given in section 2.3.

AdaBoost and its variants typically impose a stopping condition on the base classifier error rate. If this exceeds 0.5, they stop as the underlying theory only guarantees decreasing ensemble error performance for base-classifiers with better than random performance. This stopping criteria may be encountered due to the distortions introduced by the boosted weighting of some instances. However, they also stop if the error rate hits zero – this is surprisingly common in problems with low numbers of instances and large numbers of variables, where it is in fact still useful to form ensembles to counteract the high variance inherent in such a data set.

Early stopping of AdaBoost is a form of shrinkage, leading to low generalization and higher variance error. However, early stopping or low generation problem occur in original AdaBoost Algorithm and its successors. Variants of AdaBoost that halt under these conditions include MadaBoost by Domingo and Watanabc [14], $LPBoost$, $TotalBoost_v$, $TotalBoost_v^g$, Brownboost [16], BagBoosting [12], Logitboost [18], $AdaBoost_v^*$ and $AdaBoost_v^g$ by Warmuth et al [29]. Warmuth *et al* explicitly highlight early stopping as an important issue for future research.

In the original AdaBoostM1 paper [15], Freund and Schapire pointed out the main disadvantage of AdaBoostM1 that is unable to handle weak hypotheses with error greater than 0.5. It halts induction when error is greater than 0.5. To prevent early stopping, a variant of AdaBoostM1 is proposed by Bauer and Kohavi [7] to overcome this weakness. If the error is greater than 0.5, this variant of AdaBoostM1 throws away the base classifier and bootstraps a new sample set from the original input training set with identical weight 1 for every instance. It then re-builds a base classifier using the new sample. Although this allows ensemble building to continue, and may aid with variance reduction, it also discards the boosted weights which are largely responsible for the bias reduction

of AdaBoost. The model has another weakness – if one of the base classifiers achieves zero error rate, boosting stops, and furthermore the error free base classifier gets infinite voting power and becomes the only voter, turning the ensemble to a single classifier model.

Webb [30] addressed this latter issue by assigning the voting power of the error free classifier a specific value, $\log(10^{10})$, and restarting the boosting process using a new bootstrap sample from the original training set, echoing Bauer and Kohavi's approach to high errors. Webb then combines the modified algorithm with wagging and introduces another boosting algorithm, MultiBoost. However, these modified algorithms still suffer from low generalization; detailed analysis is given in section 2.3.

The importance of diversity in the pool of base classifiers has been discussed in a number of papers [2, 3, 11, 22, 23], showing that ensembles that enforce diversity fare better than ones that do not. The motivation of this work is to investigate a technique to overcome the low generalization problem of boosting algorithms. We apply the proposed technique to three boosting algorithms: the original AdaBoostM1, MultiBoost (Boosting without stopping conditions) and Arcing (Another type of Boosting with stopping conditions), and recommend the variant with highest performance.

1.2 Motivation

This research was motivated by the investigation of ensemble learning in the classification of gene expression data, which typically is high dimensional with a relatively low number of instances. We have observed that popular existing ensemble methods, including Bagging [9], Boosting (AdaBoostM1) [15] and Arcing (ArcX4) [8] and MultiBoost [30], encounter specific problems in processing such data sets which are not necessarily encountered in data sets with lower dimensionality and more samples. In our experiments in the classification of twelve gene expression data, we found that one or two error free models often dominate the ensemble. A detailed analysis is presented in section 2.

The main contribution of this research is to introduce a weight perturbation technique for boosting algorithms that increases the diversity of the base models, so reducing variance, without damaging the bias performance, and without allowing early stopping. The algorithm continues to perform, and to improve performance, when error free classifiers are encountered. The algorithm is also able to work with "unstable" (i.e. complex) classifiers such as decision trees, which inherently have relatively low bias, and are less likely to generate high error rate models than simpler base classifiers dealing with boosted samples.

The algorithm maintains instance weights, like boosting, but it addition to updating these to emphasize misclassified instances (thus reducing bias), it uses an efficient resampling technique to perturb the weights – in effect, bootstrapping from the weighted training set. This perturbation reduces variance, and allows the algorithm to continue successfully even if a zero error base classifier is encountered.

We have experimented with modified versions of Boosting, Arcing and MultiBoost, generating three modified algorithms (cw-AdaBoost, cw-Arcing and cw-MultiBoost). In evaluation, these algorithms were compared with Bagging, Boosting, Arcing and MultiBoost, in the classification of 12 gene expression datasets [4–6, 10, 13, 19, 20, 24, 27, 31, 32] utilizing the 10-fold cross validation technique. The experimental results show that the modified algorithms achieve significantly better performance than the original approaches. The cw-AdaBoost algorithm consistently achieves higher accuracy over 12 gene expression datasets than the existing algorithms. We have previously briefly introduced this work in [28].

The outline of this paper is as follows. In section 2, four benchmark algorithms (Bagging, Boosting, Arcing and MultiBoost) are described. Section 3 describes the new algorithms and the proposed modification technique, and section 4 presents the experimental results. We conclude in section 5. A detailed presentation of experimental results is given in the appendix.

2 Analyses of Benchmark Ensemble Learning Algorithms

We have benchmarked the algorithm against the original AdaBoostM1, MultiBoost (a variant of Boosting that also tries to integrate the advantages of bagging), and Arcing (Another type of boosting without stopping conditions). We have experimented with variants of each of these algorithms using the new resampling approach, and have also benchmarked against Bagging. The study shows that the proposed modification improves all of these boosting variants, but that the simple cw-AdaBoost (AdaBoostM1 integrated with the new sampling algorithm) is most effective.

2.1 Bagging

Bagging [9] forms an ensemble by bootstrapping from the data set to build individual base classifiers. These are combined using an un-weighted voting mechanism, and the classification output is the most often predicted class label. It is characteristic of Bagging that base models are constructed independently. In other words, knowledge is not accumulated between iterations: previously learned experience does not affect the learning process afterwards.

1 **Bagging Algorithm:**
Given a training set $S : (x_1, y_1), \ldots, (x_M, y_M)$ with labels $y_j \in Y = \{1, \ldots N\}$, a base learner I and the number of base models to build T, produce the Bagging classifier $C^*(x)$ by the following steps.
1. for i = 1 to T
   1.1. $\acute{S}$ = bootstrap sample from S (i.i.d. sample with replacement)
   1.2. build a base model $C_i = \mathrm{I}(S_i)$
2. $C^*(x) = \arg\max_{y \in Y}(\sum_{t:C_t(x)=y} 1)$

2 **Weakness Analysis:**
Bagging cannot reduce bias below that of the base classifiers.

2.2 AdaBoost (AdaBoostM1)

The breakthrough feature of boosting is the sequential development of base classifiers. The algorithm assigns weights to instances; in particular, the weights of misclassified instances are increased with each iteration, so that increased attention is paid to correcting mistakes made on previous iterations. The major difference between Bagging and Boosting is that individual base models in Bagging are built independent to each other whereas base models in Boosting are adaptively built. Freund and Schapire [15] proposed several extensions of Boosting called adaptive Boosting, including AdaBoost, AdaBoostM1, AdaBoostM2 and AdaBoost.R. In this research, we adopt AdaBoostM1 as the benchmark Boosting method.

1 **AdaBoostM1 Algorithm:**
Given a training set $S : (x_1, y_1), \ldots, (x_M, y_M)$ with labels $y_j \in Y = \{1, \ldots N\}$, a base learner I and the number of base models to build T, produce the Boosting classifier $C^*(x)$ by the following steps.
1. Create a new set S_1 with instance weight $w_k = 1$ where $k = 1 \ldots M$
2. for i = 1 to T
   2.1. build a base model $C_i = \mathrm{I}(S_i)$
   2.2. $E_i = \frac{1}{M}(\sum_{x_k \in S_i : C_i(x_k) \neq y_k} w_k)$
   2.3. if $(E_i > 0.5) \vee (E_i = 0)$, deduct 1 from i and abort loop.
   2.4. $B_i = \frac{E_i}{1-E_i}$
   2.5. for each $x_k \in S_i$, if $C_i(x_k) \neq y_k$, then multiply B_i to w_k
   2.6. Normalize weights
3. $C^*(x) = \arg\max_{y \in Y}(\sum_{t : C_t(x) = y} \log \frac{1}{B_t})$

2 **Weakness Analysis:**
AdaBoostM1 terminates when a base classifier with error greater than 0.5, or equal to 0, is obtained. That is, the Boosting algorithm stops learning when its performance on the training data is worse than by guessing, or it achieves perfect performance. In the most extreme case, if the error is zero on the first iteration then the algorithm constructs a single base classifier; this happens surprisingly frequently in gene expression data analysis, where the high input dimensionality and low number of instances often make it possible to achieve perfect performance on the *training* set. In such situations, the Boosting algorithm is unable to construct an effective ensemble, and its performance is drastically reduced; it has problems of low generalization and high variance. In addition, AdaBoostM1 does not have any stochastic element, and so although it achieves some variance reduction by virtue of the diverse ensembles generated, this effect is sometimes more limited than it might be.

2.3 Modified AdaBoostM1 and MultiBoost

MultiBoost [30] wraps Boosting inside Bagging and generates each bagged ensemble by Boosting, in order to combine the advantages of Boosting in bias

reduction and Bagging in variance reduction. The adopted boosting algorithm is an AdaBoostM1 variant [7], which removes the stopping condition when the error rate is greater than 0.5. Step 2.3 of the original AdaBoostM1 algorithm is modified to:

Modified AdaBoostM1 by Bauer and Kohavi [7]

2.3.1 If $E_i > 0.5$, set S_i to a bootstrap sample from original S with weight 1 for every instance and go back to step 2.1 to restart building a classifier (this step is limited to 25 times after which it exits the loop)
2.3.2 If $E_i = 0$, deduct 1 from i and abort loop

However, there is still early stopping issue in the modified AdaBoostM1 algorithm when an error free base classifier is obtained. Therefore, Webb further modifies the boosting algorithm to remove the stop conditions when $E_i = 0$. When $E_i = 0$, he assigns the voting power of the base classifier to $log(10^{10})$, resets instance weights to random weights using the continuous Poisson distribution, and re-starts the training procedure.

1 **MultiBoost Algorithm:**
Given a training set $S : (x_1, y_1), \ldots, (x_M, y_M)$ with labels $y_j \in Y = \{1, \ldots N\}$, a base learner I, the number of base models to build T, and a vector of integers V_j specifying the iteration at which each subcommittee $j > 1$ should terminate, produce the MultiBoost classifier $C^*(x)$ by the following steps.
1. Create a new set S_1 with instance weight $w_k = 1$ where $k = 1 \ldots M$
2. set j = 1
3. for i = 1 to T
3.1. if $V_j = i$,
3.1.1. reset S_i to random weights drawn from continuous Poisson distribution.
3.1.2. normalize weights
3.1.3. increment j by 1
3.2. build a base model $C_i = \mathrm{I}(S_i)$
3.3. $E_i = \frac{1}{M}(\sum_{x_k \in S_i : C_i(x_k) \neq y_k} w_k)$
3.4. if $E_i > 0.5$,
3.4.1. set S_i to random weights drawn from the continuous Poisson distribution
3.4.2. normalize weights
3.4.3. increment j by 1
3.4.4. go to step 3.2
3.5. if $E_i = 0$,
3.5.1. set $B_i = 10^{-10}$
3.5.2. set S_i to random weights drawn from the continuous Poisson distribution
3.5.3. normalize weights

3.5.4. increment j by 1
3.6. Otherwise,
3.6.1. $B_i = \frac{E_i}{1-E_i}$
3.6.2. for each $x_k \in S_i$,
3.6.2.1. if $C_i(x_k) \neq y_k$, divide w_k by $2E_i$
3.6.2.2. otherwise, divide w_k by $2(1 - E_i)$
3.6.2.3. if $w_k < 10^{-8}$, set w_k to 10^{-8}
4. $C^*(x) = \arg\max_{y \in Y}(\sum_{t:C_t(x)=y} \log \frac{1}{B_t})$

2 **Weakness Analysis:**
There are two problems with this design: first, the algorithm discards previously learned knowledge (in the form of Boosting weights) and restarts the training procedure from scratch even when it has obtained a zero error on the input training data; second, the algorithm sets B_i to 10^{-10} when an error free base model is obtained. The latter seriously affects the performance of the ensemble model, damaging its generalization performance, as such base classifiers dominate the ensemble. This drawback is apparent in our experimental results, showing that the algorithm performs very poorly in some of the datasets, such as "colon tumor" and "prostate outcome."

Table 1. MultiBoost Performance with different B_i

iteration	10	20	30	40
$Prostate_{Outcome}$				
MultiBoost($B_i = 10^{-10}$)	57.14	76.19	71.43	76.19
MultiBoost($B_i = 10^{-8}$)	52.38	76.19	71.43	76.19
cw-AdaBoost	100.00	100.00	95.24	95.24
$Breast_{Cancer}$				
MultiBoost($B_i = 10^{-10}$)	83.51	86.60	87.63	91.75
MultiBoost($B_i = 10^{-8}$)	85.57	86.60	89.69	91.75
cw-AdaBoost	90.72	95.88	95.88	95.88
$Colon_{Tumor}$				
MultiBoost($B_i = 10^{-10}$)	80.65	79.03	79.03	79.03
MultiBoost($B_i = 10^{-8}$)	80.65	79.03	79.03	79.03
cw-AdaBoost	93.55	93.55	93.55	91.94

Furthermore, resetting the weights on each iteration of Bagging discards the knowledge on weight setting gained during Boosting. We can expect each run of Boosting to converge back towards approximately the same weights, but the procedure is time-consuming. The experimental results are consistent with our theory and show that MultiBoost improves more slowly than our new algorithms. Fig 1 shows the results on one gene expression dataset, i.e. Breast Cancer, and

illustrates the faster convergence of cw-AdaBoost and cw-Arcing. In addition, we evaluate the performance of MultiBoost, which sets B_i to a bigger value 10^{-8} when $E_i = 0$ to assign smaller decision power to error free classifiers. The aim is to investigate if the performance of MultiBoost can be improved. However, there is no clear improvement by changing B_i value when $E_i = 0$. The results are displayed in Table 1.

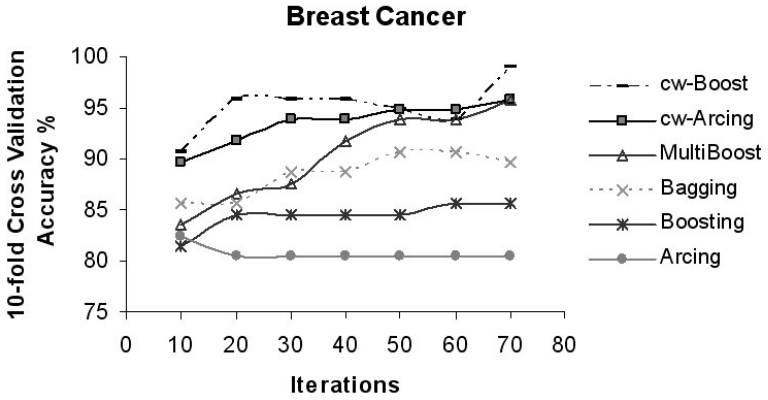

Fig. 1. Fast convergence of cw-AdaBoost and cw-Arcing

2.4 Arcing

There are two types of Arcing, i.e. Arc-fs using weighted voting and arcX4 using un-weighted voting. In this paper, we adopt arcX4 because the arcX4 algorithm is suggested to have a slight edge in test set error results [8] on smaller datasets, and the experimental datasets in this research tend to have fewer instances. The framework of Arcing is similar to the one employed in Boosting. They both proceed in sequentially self-adjusting steps. However, there are three major differences between Arcing and Boosting: (1) Arcing does not employ a stop condition; (2) Arcing adopts an un-weighted voting system; (3) Arcing adapts its behavior based on the accumulation $\{E_k\}$ of its faults in history and examines all previous base classifiers' faults when constructing a new base classifier, whereas Boosting considers only the error of the previous iteration's base classifier.

1 **Arcing Algorithm:**
Given a training set $S : (x_1, y_1), \ldots, (x_M, y_M)$ with labels $y_j \in Y = \{1, \ldots N\}$, a base learner I and the number of base models to build T, produce the Arcing classifier $C^*(x)$ by the following steps.
1. Create a new set S_1 with instance weight $w_k = 1$ where $k = 1 \ldots M$

2. Create a vector $\{E_k\}$ where $E_k = 0$ and $k = 1 \ldots M$
3. for i = 1 to T
   3.1. build a base model $C_i = \mathrm{I}(S_i)$
   3.2. for each $x_k \in S_i$, if $C_i(x_k) \neq y_k$, add 1 to E_k and set $w_k = 1 + {E_k}^4$
   3.3. Normalize weights
4. $C^*(x) = \arg\max_{y \in Y}(\sum_{t:C_t(x)=y} 1)$

2 **Weakness Analysis:**
A drawback of Arcing is its deteriorating performance and the decreasing diversity of base models as more base models are built. Once a base classifier exactly fits the training dataset, there will be no change in the accumulated misclassification values $\{E_k\}$, and hence all instances' weights remain the same in building the next base model, due to the re-weighting function ($w_k = 1 + {E_k}^4$). Thus, Arcing will continuously produce identical base models once an error-free classifier is built. In the worst case, Arcing may generate a basket of identical base classifiers. In other cases, after an error free base classifier is trained, Arcing will continuously produce identical base models until the maximum number of base models are built. Consequently, the diversity of base models of Arcing gradually decreases after that point. Our experimental results show that the accuracy of the entire Arcing model deteriorates once this happens.

3 Proposed Modification: cw-resampling

Boosting halts induction when the optimization problem becomes infeasible [14] [29]. Bauer and Kohavi [7] and Freund and Schapire [15] addressed the early stopping issue, but left open the question of iteration bounds for future research. Although Webb [30] and Breiman [8] introduced the modified boosting algorithms, MultiBoost and Arcing, to address stopping conditions issues, these modified boosting algorithms still suffer from generalization issues, as discussed above.

The proposed modification focuses on the optimization of boosting to reduce variance, and on preventing the algorithms from failing when error free classifiers occur. We therefore specify three key requirements of the proposed modification.

1 Key Requirements of the Proposed Modification

First, instead of stopping, the algorithm should be able to continue optimization and build more classifiers when an error free model is obtained. Second, the algorithm should be able to utilize the knowledge accumulated during sequential learning. In other words, when $E_i = 0$, the ensemble does not reset weights or restart from a bootstrapped set, which throws away the knowledge learned. Third, the decision of an ideal ensemble model should depend on a number of non-identical mature decision makers with low error rate rather than a few decision makers.

2 Design

The proposed model uses a weight perturbation approach to effectively resample around the weightings produced by boosting. This allows the algorithm to continue adding base classifiers even if a zero base classifier is discovered, and indeed to perform bias removal (by intermittent boosting steps) in this circumstance. Furthermore, for robustness, the decision power of base classifiers is based on their error rate rather than a fixed value. This allows boosting models to benefit from variance reduction and alleviates the overfitting problem. An illustration of the proposed design in comparison with the existing boosting algorithms is presented in Fig 2.

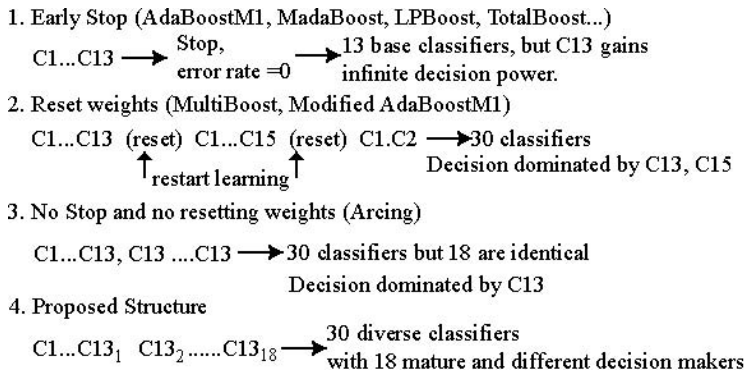

Fig. 2. Illustration of the proposed design and low generalization issues of existing boosting methods: If $C13$ is an error free classifier, boosting methods in the first group will produce only 13 classifiers no matter how large the number of base models originally specified, and as $C13$ gains infinite decision power, the decision of these ensembles is dominated by one base classifier; boosting methods in the second group assign considerably high decision power to the two error free models, $C13, C15$, and thus the decision is dominated by these two base classifiers; boosting methods in the third group continuously produce identical classifiers $C13$, and the decision of such ensemble models is dominated by this one classifier; the proposed structure generate diverse classifiers and an effective ensemble with 30 different and 18 mature decision makers.

2 Implementation

First, before training each base classifier we alter the instance weights using the random resampling approach for weighted instances described below; second, provided the base classifier performance is not zero we update the weights using a boosting approach. Thus, even if the base classifier has error zero, the algorithm continues to produce diverse classifiers. The weight perturbation algorithm injects some randomness into the learning behavior, without wholly

discarding the knowledge built up in previous iterations of boosting. It effectively bootstraps a new training set by sampling from the weighted training set generated on the previous iteration (i.e. it uses the instance weights to influence the selection frequency in bootstrapping). It thus keeps the sequential adaptive learning strengths of boosting, while injecting randomness to generate diverse classifiers and improve performance where boosting would fail.

Importantly, the standard "fairly sampling" technique as used by AdaBoost [7] and Bagging is not suitable here. The standard sampling technique resets every instance's weight to 1 and then samples with replacement. This throws away all knowledge learned, and re-starts learning from the beginning, which loses the virtue of boosting algorithms. An extreme example is MultiBoost, which resets weights both periodically and whenever $\epsilon_i > 0.5 \vee \epsilon_i = 0.5$. Under situations without error free base models obtained, MultiBoost has slower convergence and poorer performance than the new variant, with less than 40 base classifiers, as shown in Fig 1.

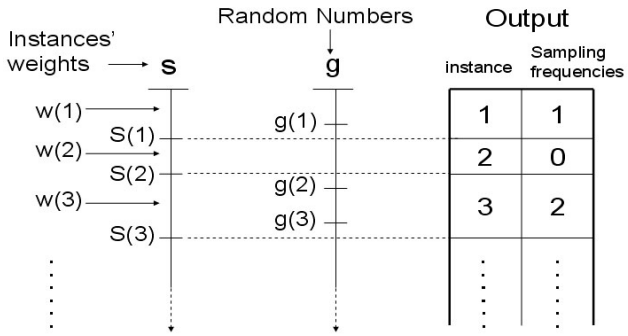

Fig. 3. Re-sampling Scheme

The weight perturbing algorithm is computationally efficient, with time complexity $O(n)$. It uses an array of cumulative weight bins, s, and an array of cumulative random numbers, g, normalized to the same final sum. The new weights are assigned according to the number of random values associated with the corresponding bin; see fig 3. This is equivalent to producing the new weights by weighted resampling with replacement but has optimal time complexity. The new algorithms (cw-AdaBoost, cw-Arcing and cw-MultiBoost) and the weight perturbing algorithm (cw-Resample) are presented below.

3.1 cw-AdaBoost Algorithm

Given a training set $S : (x_1, y_1), \ldots, (x_M, y_M)$ with labels $y_j \in Y = \{1, \ldots N\}$, a base learner I, the number of base models to build T, and an integer R (maximum

number of times to perturb data; in experiments we use R=10), produce the cw-AdaBoost classifier $C^*(x)$ by the following steps.
1. Create a new set S_1 with instance weight $w_k = 1$ where $k = 1 \dots M$
2. for i = 1 to T
   2.1. set r = 0
   2.2. perturb S_i using cw-Resample
   2.3. build a base model $C_i = \mathrm{I}(S_i)$
   2.4. increment r by 1
   2.5. $E_i = \frac{1}{M}(\sum_{x_k \in S_i : C_i(x_k) \neq y_k} w_k)$
   2.6. if $(E_i = 0) \wedge (r \leq R)$, go to step 2.2.
   2.7. if $(E_i > 0.5) \vee (E_i = 0)$, deduct 1 from i and abort loop.
   2.8. $B_i = \frac{E_i}{1 - E_i}$
   2.9. for each $x_k \in S_i$, if $C_i(x_k) \neq y_k$, then multiply B_i to w_k
   2.10. Normalize weights
3. $C^*(x) = \arg\max_{y \in Y}(\sum_{t : C_t(x) = y} \log \frac{1}{B_t})$

3.2 cw-Arcing Algorithm

Given a training set $S : (x_1, y_1), \dots, (x_M, y_M)$ with labels $y_j \in Y = \{1, \dots N\}$, a base learner I, the number of base models to build T, and an integer R (maximum number of times to perturb data; in experiments we use R=10), produce the cw-Arcing classifier $C^*(x)$ by the following steps.
1. Create a new set S_1 with instance weight $w_k = 1$ where $k = 1 \dots M$
2. Create a vector $\{E_k\}$ where $E_k = 0$ and $k = 1 \dots M$
3. for i = 1 to T
   3.1. set r = 0
   3.2. perturb S_i using cw-Resample
   3.3. build a base model $C_i = \mathrm{I}(S_i)$
   3.4. increment r by 1
   3.5. $E_i = \frac{1}{M}(\sum_{x_k \in S_i : C_i(x_k) \neq y_k} w_k)$
   3.6. if $(E_i = 0) \wedge (r \leq R)$, go to step 3.2.
   3.7. for each $x_k \in S_i$, if $C_i(x_k) \neq y_k$,
      3.7.1 add 1 to E_k
      3.7.2 $w_k = 1 + E_k^4$
   3.8. Normalize weights
4. $C^*(x) = \arg\max_{y \in Y}(\sum_{t : C_t(x) = y} 1)$

3.3 cw-MultiBoost

Given a training set $S : (x_1, y_1), \dots, (x_M, y_M)$ with labels $y_j \in Y = \{1, \dots N\}$, a base learner I, the number of base models to build T, a vector of integers V_j specifying the iteration at which each subcommittee $j > 1$ should terminate, and an integer R (maximum number of times to perturb data; in experiments we use R=10), produce the MultiBoost classifier $C^*(x)$ by the following steps.
1. Create a new set S_1 with instance weight $w_k = 1$ where $k = 1 \dots M$

2. set j = 1
3. for i = 1 to T
   3.1. set r = 0
   3.2. perturb S_i using cw-Resample
   3.3. build a base model $C_i = \mathrm{I}(S_i)$
   3.4. increment r by 1
   3.5. $E_i = \frac{1}{M}(\sum_{x_k \in S_i : C_i(x_k) \neq y_k} w_k)$
   3.6. if $(E_i = 0) \wedge (r \leq R)$, go to step 3.2
   3.7. if $(E_i > 0.5) \vee (E_i = 0)$, deduct 1 from i and abort loop
   3.8. $B_i = \frac{E_i}{1-E_i}$
   3.9. if $V_j = i$,
      3.9.1. reset S_i to random weights drawn from continuous Poisson distribution.
      3.9.2. normalize weights
      3.9.3. increment j by 1
   3.10. otherwise,
      3.10.1. for each $x_k \in S_i$, if $C_i(x_k) \neq y_k$, multiply B_i to w_k
      3.10.2. normalize weights
4. $C^*(x) = \arg\max_{y \in Y}(\sum_{t:C_t(x)=y} \log \frac{1}{B_t})$

3.4 cw-Resample Algorithm

Given a dataset S: a sequence of instances with weights:$(i_1, w_1), \ldots, (i_M, w_M)$, we produce new dataset with the following steps.
1. Generate M random number: $r_1, \ldots, r_M$
2. $R = \sum_{j=1 \ldots M} r_j$
3. $W = \sum_{j=1 \ldots M} w_j$
4. set $a = 1$ and $b = 1$
5. let $g(a) = (\sum_{j=1 \ldots a} \frac{r_j}{R}) \times M$
6. let $s(b) = (\sum_{j=1 \ldots b} w_j$
7. if $(a > M) \wedge (b > M)$, then terminate.
8. if $g(a) < s(b)$,
   8.1. select instance i_b into the output dataset
   8.2. increment a by 1
   8.3. go to step 5
9. otherwise,
   9.1. increment b by 1
   9.2. go to step 5

4 Experiments

The experiments are conducted using 12 published gene expression datasets [4–6, 10, 13, 19, 20, 24, 27, 31, 32], which are obtained from [26]. Details of the data cleaning process are given in the original paper. In evaluation, Ambroise

and McLachlan [1] recommend using 10-fold rather than leave-one-out cross-validation for gene expression data analysis. In this research, 10-fold cross validation is utilized and C4.5 decision tree algorithm [25] is used as the base classifier. Furthermore, to investigate the influence of the number of base models used, we evaluate the classification accuracy of the ensembles with different numbers of base classifiers (from 10 to 70 classifiers in steps of 10). The experimental results show that the modified algorithms all perform better than the corresponding original algorithms. The cw-AdaBoost algorithm consistently performs best over all 12 gene expression datasets, and is our recommended variant. In order to compare the performances of the seven algorithms on 12 datasets

Table 2. 10-fold Cross Validation Accuracy% for single dataset(Breast Cancer)

base classifiers	10	20	30	40	50	60	70	P_m
Bagging	85.57	85.57	88.66	88.66	90.72	90.72	89.69	-0.71
Arcing	82.47	80.41	80.41	80.41	80.41	80.41	80.41	-8.52
Boosting	81.44	84.54	84.54	84.54	84.54	85.57	85.57	-4.83
MultiBoost	83.51	86.60	87.63	91.75	93.81	93.81	95.88	1.20
cw-Arcing	89.69	91.75	93.81	93.81	94.84	94.84	95.88	4.29
cw-AdaBoost	90.72	95.88	95.88	95.88	94.84	93.81	98.97	5.91
cw-MultiBoost	90.72	87.63	89.69	91.75	91.75	95.88	95.88	2.67
Average E_i	86.30	87.48	88.66	89.54	90.13	90.72	91.75	

with different number of iterations, we first generate the cross validated average accuracy E_i of the algorithms on a specific iteration number i, to represent the average performance of 7 algorithms with iteration i. Given $A_i(m)$ is 10-fold cross validation accuracy of the algorithm m with iteration i, $E_i = A_i(m)/7$. We then create a performance index P_m to compare the relative performance of the algorithm m. ($P_m = (\sum_{i=10,20,..,70}(A_i(m) - E_i))/7$.) Table 2 illustrates the performance index on a single dataset, and Table 3 displays the relative performance indices P_m on 12 gene expression datasets, showing that the new methods (particularly cw-AdaBoost, which has the best results for nine datasets, and the second best for two others) obtain consistently high performance index values.

Using the Wilcoxon signed rank test to compare the performance of cw-AdaBoost with cs-Arcing (the second best algorithm), we obtain the Wilcoxon statistic W=52 with N=12 samples, yielding the z-value $2.02 > 1.96$, and therefore conclude that the performance is significantly better at the 97.5% one-sided confidence level. Stronger results are obtained in comparing cw-AdaBoost with the algorithms, so that we conclude that cw-AdaBoost has superior performance. (The 10-fold cross validation results on the other 11 datasets are presented in the Appendix.)

A distinctive feature of gene expression data is that error free base model can be generated, causing the low generalization issue discussed above. The results

Table 3. Performance Index P_m on 12 Gene Expression Datasets

	AMLALL	Brain	Breast	CNS	Colon	$DLBCL_t$
Bagging	-1.39	-1.22	-0.71	-0.34	-5.73	-1.83
Arcing	-7.14	-2.37	-8.52	-9.63	-1.12	-7.77
Boosting	-4.36	-4.08	-4.83	1.56	6.02	1.14
MultiBoost	3.37	-0.37	1.20	-0.82	-7.11	1.51
cw-Arcing	3.37	2.49	4.29	3.23	8.10	1.69
cw-AdaBoost	3.17	4.49	5.91	5.13	6.26	2.99
cw-MultiBoost	2.98	1.06	2.67	4.42	-6.42	2.25
	Lung	$DLBCL_o$	$Prostate_o$	$MLL_{Leukemia}$	$Prostate_t$	Subtype
Bagging	-0.72	-4.27	-15.86	-2.88	-0.94	0.61
Arcing	-0.72	-9.20	-3.61	-1.87	-1.18	-6.56
Boosting	-0.17	-2.20	-6.33	-0.87	-0.05	1.57
MultiBoost	1.02	0.90	-5.65	-1.47	-1.99	0.78
cw-Arcing	0.94	5.09	12.04	0.32	2.45	0.17
cw-AdaBoost	1.49	5.33	18.84	4.28	1.72	1.75
cw-MultiBoost	1.17	4.34	12.72	2.49	-0.62	1.66

are consistent with our theories, which are: (1) arcing keeps producing identical base models after an error free base model is built, and therefore the diversity of base models of arcing deteriorates. Thus, the variance error increases afterwards; (2) the stopping condition of boosting terminates further constructions of base classifiers and prevents further reduction of the variance error; (3) the performance of the multiboost is adversely affected by assigning $\log 10^{10}$ to the decision power of an error free base model, leading to a small number of error free base classifiers domaining the decision output.

4.1 Variance and Bias

In this section we present an analysis of the bias and variance of the algorithms, using the breast cancer dataset, and Kohavi and Wolpert's [21] approach to variance and bias decomposition.

There are 97 instances with 835 attributes in the original breast cancer data set D. Utilizing Kohavi and Wolpert's approach [21], a sample of size 40 without replacement from the original data D is taken to produce a training set source. From the remainder, a test set of size 40 is sampled without replacement. There are 50 training samples to produce 50 trained ensemble models, which are then applied to the test set, and the bias and variance are calculated from the predictions on the test set.

Fig 4 tabulates the experimental results on the bias, variance and error of the ensemble models. Fig 5 illustrates the differences in the performance of the original algorithms and the modified method. In general, the results show that the modified algorithms perform well on both bias and variance reduction. The

Bias^2	10	20	30	40	50	60	80
Bagging	0.1381	0.129	0.1307	0.1281	0.132	0.1308	0.1272
Boosting	0.113	0.1102	0.1079	0.1118	0.1063	0.1076	0.1079
cw-Boost	0.1135	0.1074	0.1045	0.1027	0.1011	0.0981	0.0974
Arcing	0.1408	0.1244	0.1155	0.1089	0.1108	0.1121	0.1142
cw-Arcing	0.1269	0.1094	0.1063	0.1057	0.1064	0.1038	0.1053
MultiBoost	0.1292	0.1137	0.1098	0.1058	0.1067	0.1038	0.1017
cw-MultiBoost	0.1236	0.1117	0.1138	0.1049	0.1082	0.1024	0.1074

Variance	10	20	30	40	50	60	80
Bagging	0.1662	0.1549	0.1542	0.1523	0.1492	0.1483	0.1467
Boosting	0.1596	0.1495	0.1464	0.1453	0.1448	0.1453	0.1453
cw-Boost	0.1285	0.11	0.1015	0.1012	0.0997	0.0978	0.0937
Arcing	0.1636	0.1529	0.1519	0.1608	0.1717	0.178	0.1858
cw-Arcing	0.1191	0.1163	0.1155	0.112	0.112	0.1122	0.1117
MultiBoost	0.1695	0.1413	0.1359	0.1291	0.1283	0.1259	0.1239
cw-MultiBoost	0.1279	0.1213	0.1078	0.1042	0.1013	0.1011	0.0962

Error	10	20	30	40	50	60	80
Bagging	0.3077	0.287	0.2881	0.2835	0.2842	0.2821	0.2768
Boosting	0.2758	0.2628	0.2572	0.26	0.254	0.2558	0.2561
cw-Boost	0.2446	0.2196	0.2081	0.206	0.2028	0.1979	0.193
Arcing	0.3077	0.2804	0.2705	0.273	0.286	0.2937	0.3039
cw-Arcing	0.2484	0.2281	0.2242	0.22	0.2207	0.2182	0.2193
MultiBoost	0.3021	0.2579	0.2484	0.2375	0.2375	0.2323	0.2281
cw-MultiBoost	0.254	0.2354	0.2239	0.2112	0.2116	0.2056	0.2056

Fig. 4. $Bias^2$, $Variance$ and Error .

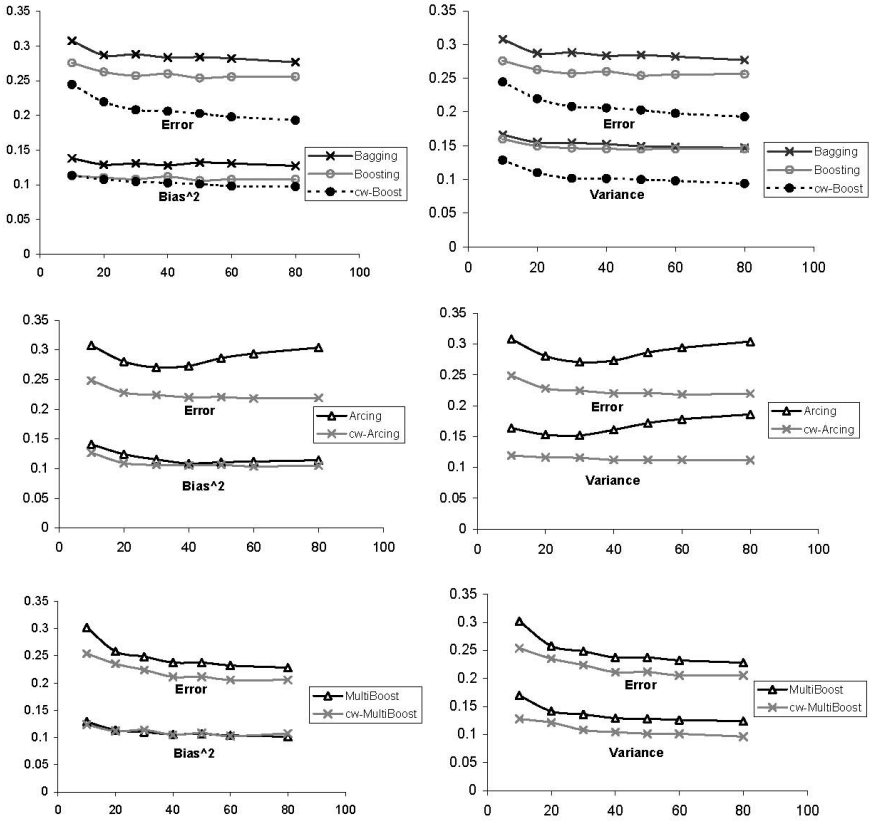

Fig. 5. Comparison on $Bias^2$, $Variance$ and Error between the original algorithms and the algorithms with the proposed modifications.

top row, which compares bagging, boosting and the new algorithm show that, for this data set, boosting and bagging have equal performance on variance, but as expected boosting has lower bias. The cw-AdaBoost algorithm matches this low bias, and is able to continue boosting, further reducing the bias. The cw-AdaBoost algorithm also has noticeably lower variance than either boosting or bagging, indicating that the weight perturbation approach is highly effective. The second row show that the standard arcing algorithm deteriorates after around 30 iterations, at which point it keeps producing identical base models after an error free base model is built, leading to growing variance error. Boosting similarly stops improving after about 50 iterations. MultiBoost restarts learning every 10 iterations, and so it benefits from variance reduction in comparison to the original Boosting algorithm, but converges slowly; the proposed modification cw-MultiBoost variant achieves faster variance reduction.

5 Conclusion

This paper has introduced modifications to three boosting methods to generate efficient boosting models for training high dimensional datasets with low numbers of instances. Training this type of dataset (particularly with unstable base classifiers like decision tree) tends to generate error free base models and causes malfunctions on conventional Boosting, Arcing, and MultiBoost, leading to low generalization. The modified algorithms, which use a weight perturbation method combined with sequential update, and discards the stopping condition, allows ensemble generation to continue, further lowering variance. We have introduced the cw-AdaBoost algorithm, which demonstrates superior performance on 12 gene expression data where it performs consistently well. We thus recommend it for wider use.

6 Acknowledgement

The authors thank the anonymous reviewers for their valuable comments.

References

1. Ambroise, C., , McLachlan, G. J.: Selection bias in gene extraction on the basis of microarray gene-expression data. Proceedings of the National Academy of Sciences of the United States of America **99(10)** (2002) 6562–6566.
2. Amit, Y., , Blanchard G.: Multiple Randomized Classifiers. Technical report, University of Chicago (2001)
3. Ali, K. M., , Pazzani, M. J.: Error Reduction through Learning Multiple Descriptions. International Journal of Machine Learning **24** (1996) 173–202
4. Alon, U., , Barkai, N., , Notterman, D. A., , Gish, K., , Ybarra, S., , Mack, D., , Levine, A. J.: Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. National Academy of Science, Cell Biology **96** (1999) 6745–6750

5. Armstrong, S. A., , Staunton, J. E. , , Silverman, L. B., , Pieters, R. et al.: MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia. Nature Genetics **30** (2002) 41–47
6. Ash, A. A., , Michael, B. E., , Davis, R. E. , , Ma, C., , Izidore, S. L., , Andreas, R. et al.: Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. Nature **403** (2000) 503–511
7. Bauer, E., , Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. International Journal of Machine Learning **36** (1999) 105–139
8. Breiman, L.: Bias, Variance, and Arcing Classifiers. Technical report 460, Statistics Department, UC Berkeley (1996)
9. Breiman, L.: Bagging predictors. International Journal of Machine Learning **24** (1996) 134–140
10. Catherine, L. N., , Mani, D. R. , , Rebecca, A. B., , Pablo, T. et al.: Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. Cancer Research **63** (2003) 1602–1607
11. Dasgupta, S., , Long, P. M.: Boosting with diverse base classifiers. Proceedings of the Conference on Computational Learning Theory (2003) 273–287
12. Dettling, M.: BagBoosting for tumor classification with gene expression data. Bioinformatics **20(18)** (2004) 3583–3593
13. Dinesh, S., , Phillip, G. F., , Kenneth, R., , Donald, G. J., , Judith, M. et al.: Gene expression correlates of clinical prostate cancer behavior. Cancer Cell **1** (2002) 203–209
14. Domingo, C. , , Watanabe, O.: MadaBoost: A modification of AdaBoost. Technical Reports on Mathematical and Computing Sciences TR-C138 (2000)
15. Freund, Y., , Schapire, R.: Experiments with a new boosting algorithm. Proceedings of the Thirteenth International Conference on Machine Learning, San Francisco (1996) 148–156
16. Freund, Y.: An Adaptive Version of the Boost by Majority Algorithm. International Journal of Machine Learning **43(3)** (2001) 293–318
17. Friedman, J.: Stochastic Gradient Boosting. Computational Statistics and Data Analysis **38** (2002) 367–368
18. Friedman, J. H., , Hastie, T. , , Tibshirani, R.: Additive logistic regression: A statistical view of boosting. The Annals of Statistics **28** (2000) 337–374
19. Gavin, J. G., , Roderick, V. J., , Li-Li, H., , Steven, R. G., , Joshua, E. B., , Sridhar, R., , William, G. R., , David, J. S., , Raphael, B.: Translation of Microarray Data into Clinically Relevant Cancer Diagnostic Tests Using Gene Expression Ratios in Lung Cancer and Mesothelioma. Cancer Research **62** (2002) 4963–4967
20. Golub, T. R., , Slonim, D. K., , Tamayo, P., , Huard, C., , Gaasenbeek, M. et al.: Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science **286** (1999) 531–537
21. Kohavi, R., , Wolpert, D.: Bias plus variance decomposition for zero-one loss functions. Proceedings of the Thirteenth International Machine Learning Conference.
22. Kuncheva, L. I.: Diversity in multiple classifier systems. Information Fusion **6** (2005) 3–4
23. Long, P.M., , Vega, V.B.: Boosting and Microarray Data. International Journal of Machine Learning **52** (2003) 31-44
24. Pomeroy, S. L., , Tamayo, P., , Gaasenbeek, M., , Sturla, L. M. et al.: Prediction of central nervous system embryonal tumour outcome based on gene expression. Nature **415** (2002) 436–442

25. Quinlan, J. R.: Bagging, boosting and c4.5. Proceedings of the thirteenth national conference on artificial intelligence (1996) 725–730.

26. Tan, A.C. , , Gilbert, D.: Ensemble machine learning on gene expression data for cancer classification. Applied Bioinformatics **2** (2003) S75–S83

27. Veer, L. J., , Dai, H., , van de Vijver, M. J., , He, Y. D., , Hart, A. A., , Mao, M., Peterse, H. L. et al.: Gene expression profiling predicts clinical outcome of breast cancer. Nature **415(6871)** (2002) 484–5

28. Wang, C. -W.: New Ensemble Machine Learning Method for Classification and Prediction on Gene Expression Data. Proceedings of the international conference of the IEEE Engineering in Medicine and Biology Society **2** (2006) 3478–3481

29. Warmuth, M. K., , Liao, J., , Ratsch, G: Totally corrective boosting algorithms that maximize the margin. In Proceedings of the 23rd international Conference on Machine Learning **148** (2006) 1001–1008

30. Webb, G. I.: MultiBoosting: A Technique for Combining Boosting and Wagging. International Journal of Machine Learning **40** (2000) 159–196

31. Yeoh, E. J., , Ross, M. E., , Shurtleff, S. A., , Williams, W. K., , Patel, D., , Mahfouz, R., , Behm, F. G. et al.: Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. Cancer Cell **1(2)** (2002) 133–143

32. Zembutsu, H., , Ohnishi, Y., , Tsunoda, T., , Furukawa, Y., , Katagiri, T., , Ueyama, Y. et al.: Genome-wide cDNA microarray screening to correlate gene expression profiles with sensitivity of 85 human cancer xenografts to anticancer drugs. Cancer Research **62(2)** (2002) 518–27

Appendix

iteration	10	20	30	40	50	60	70
$Prostate_{Outcome}$							
Bagging	66.67	61.90	61.90	61.90	61.90	61.90	61.90
Arcing	66.67	76.19	76.19	80.95	80.95	71.43	71.43
Boosting	76.19	71.43	71.43	71.43	71.43	71.43	71.43
MultiBoost	57.14	76.19	71.43	76.19	76.19	76.19	76.19
cw-Arcing	90.48	95.24	90.48	90.48	90.48	85.71	90.48
cw-AdaBoost	100.00	100.00	95.24	95.24	100.00	95.24	95.24
cw-MultiBoost	80.95	95.24	95.24	85.71	95.24	95.24	90.48
AMLALL							
Bagging	91.67	94.44	94.44	95.83	95.83	94.44	95.83
Arcing	88.89	88.89	88.89	88.89	88.89	88.89	88.89
Boosting	91.67	91.67	91.67	91.67	91.67	91.67	91.67
MultiBoost	95.83	100.00	100.00	100.00	100.00	100.00	100.00
cw-Arcing	98.61	100.00	98.61	100.00	100.00	100.00	98.61
cw-AdaBoost	100.00	98.61	98.61	98.61	98.61	100.00	100.00
cw-MultiBoost	97.22	100.00	98.61	100.00	98.61	100.00	98.61
Brain Tumor							
Bagging	86.00	88.00	86.00	84.00	86.00	82.00	82.00
Arcing	82.00	84.00	84.00	84.00	84.00	84.00	84.00
Boosting	80.00	84.00	82.00	82.00	82.00	82.00	82.00
MultiBoost	72.00	82.00	88.00	88.00	88.00	90.00	92.00
cw-Arcng	82.00	86.00	90.00	90.00	92.00	90.00	90.00
cw-AdaBoost	88.00	92.00	92.00	88.00	92.00	90.00	92.00
cw-MultiBoost	80.00	84.00	92.00	90.00	88.00	88.00	88.00
ColonTumor							
Bagging	82.26	82.26	79.03	79.03	80.65	80.65	80.65
Arcing	91.94	85.48	83.87	83.87	83.87	83.87	83.87
Boosting	90.32	90.32	91.94	93.54	93.54	93.54	93.54
MultiBoost	80.65	79.03	79.03	79.03	79.03	79.03	79.03
cw-Arcing	96.77	93.55	91.94	93.55	93.55	95.16	96.77
cw-AdaBoost	93.55	93.55	93.55	91.94	91.94	91.94	91.94
cw-MultiBoost	80.65	80.65	85.48	74.19	79.03	79.03	80.65

$DLBCL_{Tumor}$							
Bagging	93.51	92.21	92.21	92.21	92.21	92.21	92.21
Arcing	92.21	84.42	85.71	85.71	85.71	85.71	85.71
Boosting	96.10	96.10	96.10	94.81	94.81	94.81	94.81
MultiBoost	92.21	94.81	97.40	97.40	96.10	96.10	96.10
cw-Arcing	90.90	96.10	97.40	94.81	97.40	97.40	97.40
cw-AdaBoost	92.20	97.40	97.40	98.70	97.40	98.70	98.70
cw-MultiBoost	93.51	96.10	96.10	96.10	98.70	97.40	97.40
$DLBCL_{Outcome}$							
Bagging	79.31	87.93	89.66	94.83	91.38	93.10	94.83
Arcing	86.21	86.21	86.21	84.48	84.48	84.48	84.48
Boosting	89.66	89.66	93.81	93.10	93.10	93.10	93.10
MultiBoost	84.48	94.83	96.55	96.55	98.28	98.28	98.28
cw-Arcing	98.28	98.28	100.00	100.00	100.00	100.00	100.00
cw-AdaBoost	100.00	100.00	100.00	100.00	100.00	100.00	98.28
cw-MultiBoost	91.34	100.00	100.00	100.00	100.00	100.00	100.00
Lung Cancer							
Bagging	97.24	97.24	97.24	97.24	97.24	97.24	97.24
Arcing	97.24	97.24	97.24	97.24	97.24	97.24	97.24
Boosting	97.79	97.79	97.79	97.79	97.79	97.79	97.79
MultiBoost	97.79	98.90	98.90	98.90	99.45	99.45	99.45
cw-Arcing	98.90	98.90	99.45	99.45	98.90	98.34	98.34
cw-AdaBoost	99.45	99.45	99.45	99.45	99.45	99.45	99.45
cw-MultiBoost	97.79	98.90	99.45	99.45	99.45	99.45	99.45
$MLL_{Leukemia}$							
Bagging	90.23	90.23	90.23	91.67	91.67	91.67	91.67
Arcing	91.67	93.06	93.06	91.67	91.67	91.67	91.67
Boosting	93.06	93.06	93.06	93.06	93.06	93.06	93.06
MultiBoost	93.06	93.04	91.67	93.06	93.06	91.67	91.67
cw-Arcing	95.84	95.84	95.84	93.06	93.06	93.06	93.06
cw-AdaBoost	95.84	100.00	95.83	98.61	100.00	98.61	98.61
cw-MultiBoost	91.67	94.44	98.61	95.83	97.22	98.61	98.61
CNS							
Bagging	88.33	88.33	86.67	88.33	86.67	86.67	88.33
Arcing	78.33	78.33	78.33	78.33	78.33	78.33	78.33
Boosting	88.33	88.33	90.00	90.00	90.00	90.00	90.00
MultiBoost	85.00	86.67	85.00	86.67	86.67	90.00	90.00
cw-Arcing	86.67	91.67	93.33	91.67	91.67	91.67	91.67
cw-AdaBoost	91.67	91.67	95.00	93.33	93.33	93.33	93.33
cw-MultiBoost	93.33	88.33	91.67	93.33	93.33	93.33	93.33

$Prostate_{Tumor}$							
Bagging	94.92	95.48	95.48	95.48	94.92	94.92	94.92
Arcing	94.92	94.92	94.92	94.92	94.92	94.92	94.92
Boosting	96.05	96.05	96.05	96.05	96.05	96.05	96.05
MultiBoost	88.24	94.12	94.85	95.59	95.59	95.59	94.85
cw-Arcing	98.87	98.31	98.31	98.31	98.31	98.87	98.87
cw-AdaBoost	97.74	97.74	98.31	97.18	97.74	97.74	98.31
cw-MultiBoost	93.38	94.85	96.32	95.59	95.59	97.06	95.59
SubtypeALL							
Bagging	89.91	91.13	91.13	91.13	90.83	91.13	91.13
Arcing	88.07	89.91	85.63	81.65	80.73	80.12	80.12
Boosting	89.30	92.05	92.97	92.05	91.74	92.66	92.35
MultiBoost	86.85	88.99	91.44	92.35	92.66	92.66	92.66
cw-Arcing	89.30	90.21	89.91	90.83	91.13	91.13	90.83
cw-AdaBoost	90.21	92.05	92.66	92.05	92.05	92.35	92.97
cw-MultiBoost	88.07	92.35	92.05	92.05	92.35	93.58	93.27