Proceedings of International Joint Conference on Neural Networks, Orlando, Florida, USA, August 12-17, 2007

Enhancing Boosting by Feature Non-Replacement for Microarray Data Analysis

Geoffrey R. Guile and Wenjia Wang

Abstract—**We have investigated strategies for enhancing ensemble learning algorithms for DNA microarray data analysis. By using modified versions of AdaBoost, LogitBoost and BagBoosting we have shown that feature non-replacement provides an effective enhancement to the performance of all three algorithms, and overall, BagBoosting with feature non-replacement had the lowest error rates when used on six commonly-used cancer datasets.**

I. Introduction

DNA microarrays, which provide gene expression values, can be used for prediction of disease outcome, classification of cancer types and identification of relevant genes. However, because of high dimensionality and complexity (typically several thousand genes and less than a hundred samples,) machine learning techniques are needed for proper analyses. Ensemble methods such as bagging and boosting which produce a committee of classification models can be more accurate than those that produce a single model and have been successfully applied to DNA microarray data.

The principle behind ensemble learning is that although a classification algorithm may only be able to produce a model with slightly better accuracy than random guessing, if several such models are produced and combined into an ensemble, their combined accuracy will be greater than any single classifier, providing they are sufficiently diverse from each other to avoid making similar errors. Clearly, if all the models are identical there will be no advantage in having more than one, and boosting algorithms are therefore designed with the aim of producing a high level of diversity. In order to produce an ensemble, boosting algorithms iteratively employ another algorithm known as the *base learner* to generate a series of models. Any algorithm normally used for classification or prediction can be employed as the base learner, providing it allows weighting of samples. Initially all samples have equal weights, then after the first iteration the accuracy of the model produced is measured and the samples weights are adjusted so that the weights of misclassified samples are increased while those of correctly classified samples are reduced. At the next iteration the base learner will concentrate on the misclassified samples. A series of models is then produced with the sample weights being adjusted each time. These models are then combined into an ensemble voting committee. Normally the voting weight of each model is set according to its accuracy on the data. Different boosting algorithms vary in the way that the sample weights are adjusted and the way the votes of the committee members are set. These differences may result in different overall performance on a given dataset.

In this study we have investigated methods for enhancing boosting techniques when applied to the classification of six commonly used cancer microarray datasets.

II. Related Work

Boosting was applied to DNA microarray data for sample classification by Ben-Dor *et al.* [1] who compared the performance of several high-dimensional classification techniques, including AdaBoost [2] with which they used decision stumps as the base learner. Dudoit *et al.* [3] compared the performance of several classification methods including bagging and boosting. For their boosting algorithm they used AdaBoost with the Classification and Regression Trees (CART) algorithm [4] as the base learner. These reports suggested that boosting did not perform well on microarray data. However, several reports of the effective use of boosting with DNA microarray data have now been published. Dettling and Bühlmann [5] applied the LogitBoost algorithm [6] to microarray data and reported slightly better results than with AdaBoost. AdaBoost employs an exponential criterion for adjusting sample weights, while LogitBoost uses the binomial log likelihood, which increases linearly rather than exponentially for strong negative margins. According to Dettling and Bühlmann [5] this makes LogitBoost more robust when the data contain noise, or samples are mislabeled.

Long and Vega [7] investigated the effect of not replacing features once they had been used. They applied this strategy to AdaBoost and the Arc-x4-RW algorithm [4]. They found that the non-replacement versions of the algorithms gave lower error rates than those with replacement.

Ben-Dor *et al.* [1], Dettling and Bühlmann [5], and Long and Vega [7] all used decision stumps as the base learner. The decision stump is the simplest form of decision tree, using only one feature, and is therefore easy to implement and requires minimal computing power. However, decision stumps suffer from the limitation that they are unable to take

Manuscript received January 29, 2007.

G. R. Guile is a with the School of Computing Sciences, University of East Anglia, Norwich, Norfolk, NR4 7TJ, UK. (e-mail: g.guile@uea.ac.uk).

W. Wang is with the School of Computing Sciences, University of East Anglia, Norwich, Norfolk, NR4 7TJ, UK. (e-mail: wenjia.wang@uea.ac.uk).

interaction between features into account.

The BagBoosting algorithm [8] is based on LogitBoost and combines boosting with bagging. It was found to give lower error rates than standard LogitBoost [8]. At each boosting iteration several sets of bagged samples are produced by random sampling with replacement. The base learner produces a separate decision stump model for each set of bagged samples and all the models from that iteration are then added to the ensemble. The sample weights are taken into consideration when producing the decision stumps, but not when performing the sample bagging.

III. Methods

A. *Feature preselection*

Because of the high dimensionality of microarray data, most studies employ some form of feature preselection. Many preselection methods have been applied to microarray data, including Student's t-test [9] and the sign test (TNoM score) [1]. For this study we used the Wilcoxon-Mann-Whitney U-test which was first applied to microarray data by Park *et al.* [10] and also used by Dettling and Bühlmann [5] and Dettling [8]. This is a non-parametric test which allows genes to be ranked according to how well they discriminate between sample classes. For microarray data it has advantages over the t-test in that is less sensitive to outliers and does not assume a normal distribution (see [5]).

B. *Algorithms learning with feature non-replacement*

For the boosting algorithms we used AdaBoost, LogitBoost and BagBoosting, with and without feature replacement. The non-replacement strategy as applied to AdaBoost and LogitBoost works by removing a feature from the training dataset once it has been used in a model, thus any individual feature can be used only once in the ensemble. Fig 1. shows the boosting process.

The reason for employing the non-replacement strategy is that it forces the base learner to use unused features to potentially increase the amount of information taken into consideration when constructing the ensemble and increase the diversity of the models. Microarray data are usually of high dimensionality, characterized by a large number of attributes but a small number of samples. The attributes, representing the expression levels of different genes, contain a large amount of complementary information. However, if during boosting a feature correctly classifies all the training samples, (which is highly likely due to the high dimensionality of the data,) the weights will not change for the next iteration and the same feature will continue to be selected. By removing the features as they are used at each iteration, not only is the algorithm concentrating on the more difficult samples in the normal way, but now it is forced to use information different from that previously used. Therefore, the model produced will be based on information that is probably complementary to that used when making the previous models. Thus, more and diverse knowledge is learned (by different models) in producing the ensemble, which has a greater degree of diversity between the models.

For BagBoosting, which produces more than one model at each iteration, the number of available features is rapidly reduced, and if all features used at each iteration are removed, then a large initial number of features is required. Therefore, a modified non-replacement strategy was used where one of the features used at each iteration was randomly selected to not be replaced. (See discussion.)

Decision stumps were used as the base learner, therefore each model used only one feature. It should be noted that Long and Vega [7] used a modified voting method in their non-replacement version of AdaBoost where a stump that correctly classified all samples was not allowed to vote with infinite weight; in contrast to this we used the standard AdaBoost voting method, whether or not features were replaced, in order to ensure that any differences were entirely due to the non-replacement strategy.

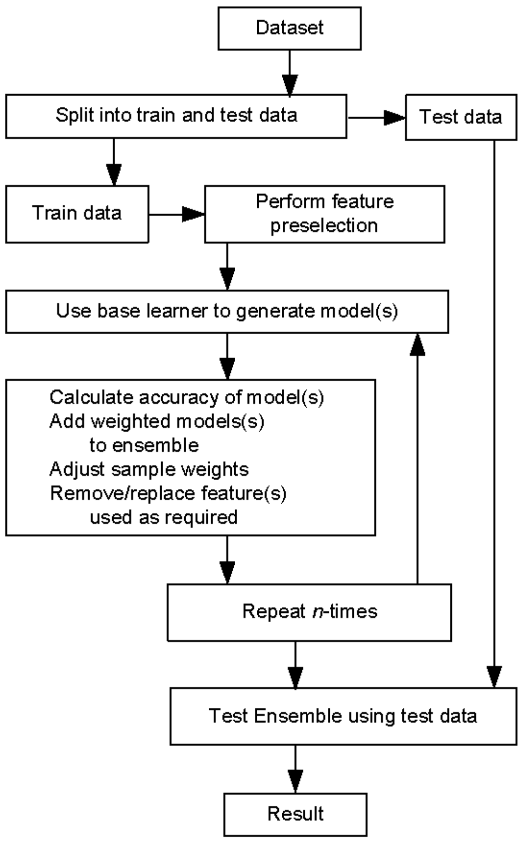
Dataset
Split into train and test data
Test data
Train data
Perform feature preselection
Use base learner to generate model(s)
Calculate accuracy of model(s)
Add weighted models(s) to ensemble
Adjust sample weights
Remove/replace feature(s) used as required
Repeat *n*-times
Test Ensemble using test data
Result

Fig. 1. Flowchart of boosting process.

C. *Number of boosting iterations*

With successive iterations of boosting the accuracy of an ensemble improves, at some stage a peak accuracy is reached which may be followed by a decrease in accuracy. Eventually a stable accuracy will be reached. Ideally boosting should be stopped once the peak accuracy has been reached. However, deciding where to stop boosting is something that is not easy to determine using just the training data. Dettling and Bühlmann [5] employed leave-one-out cross validation on the training data to estimate the best number of iterations, but found that this usually gave worse results than using a fixed number of 100 iterations. Long and Vega [7] used the same number of iterations as features preselected: either 10 or 100. Our initial investigations (results not included in this paper) showed that the majority of ensembles reached maximum accuracy within 10 iterations of boosting. Therefore, for all the experiments performed for this study the number of iterations was set at 25, to ensure that a stable level of accuracy had been reached. Both the best and final accuracies are reported.

D. Experiments

A flowchart of the experiments is shown in Fig. 2.

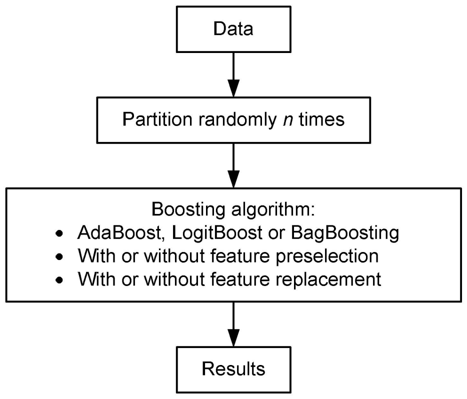
Data
Partition randomly *n* times
Boosting algorithm:
• AdaBoost, LogitBoost or BagBoosting
• With or without feature preselection
• With or without feature replacement
Results

Fig. 2. Flowchart of microarray data classification.

A dataset was randomly partitioned into a training set comprising two thirds of the data and a test set comprising one third. Each experiment was repeated 50 times with different random partitions to test the consistency of the models. The partitioning program used a random seed to ensure that the same set of partitions could be generated repeatedly, in order that experiments could be reproduced. Feature preselection was always performed only on the training dataset of each split, in order to ensure that the results obtained on the test dataset best reflected those likely to be obtained with new, out of sample data.

For datasets with more than two sample classes, we expanded the sample attribute into binary attributes for each class.

AdaBoost and BagBoosting were applied with and without feature replacement to each set of data. In order to allow fair comparison with the work of others, and to investigate the effects of preselecting different numbers of genes, experiments were performed with 50, 100, 200 and 500 genes preselected, and without preselection. Feature selection was only performed on the training dataset of each partition. For BagBoosting, 25 iterations of bagging were performed at each boosting stage. The lowest and final test set error rates obtained with each partition were averaged and recorded. Differences of the mean error rates with and without feature replacement were calculated. For the datasets with more than two sample classes, results were generated for classifying each class against all the other classes, then the mean results for all classes were calculated.

E. Datasets

Six microarray datasets were used as test cases in this study: the Brain dataset [11], the Colon dataset [9], the Leukemia dataset [12], the Lymphoma dataset [13], the Prostate dataset [14] and the Small Round Blue Cell Tumor (SRBCT) dataset [15]. The Lymphoma and Leukemia datasets were preprocessed as in Dudoit *et al.* [3].[1] A summary of the database demography is given in Table 1.

TABLE 1
DEMOGRAPHY OF THE SIX DATASETS

Dataset	Number of		
	Features (genes)	Samples	Classes
Leukemia	3571	72	2
Colon	2000	62	2
Prostate	6033	102	2
Lymphoma	4026	62	3
SRBCT	2308	63	4
Brain	5597	42	5

F. Code

The code for randomizing the data was written in Java version 1.5.0. Experiments were performed using the R Statistical Package version 2.3.1. The code for AdaBoost, LogitBoost and BagBoosting was modified from that of the R package "boost" of [8], with the additional option incorporated to allow the user to decide whether features were to be replaced once they had been used.

IV. Results

Summaries of the results obtained are given in Tables 2–7. Full tables of results are given in the Appendix.

Discussion and Evaluation

A. Results compared with previous studies

For comparison purposes, we repeated as closely as possible the relevant previous work. The results we obtained for AdaBoost with and without feature replacement (Table 2) closely match those obtained by Long and Vega [7], while our results for LogitBoost (Table 3) and BagBoosting (Table 5) closely match those obtained by Dettling [8]. Slight variations occur between experimenters due to differences in data partitioning. The comparisons of our results with those of previous workers have verified our experimental procedure and implementation, and thus will validate the results we obtained when applying the non-replacement

[1] These datasets are all available preprocessed as used by Dettling [8] from the website http://stat.ethz.ch/~dettling/bagboost.html.

strategy to LogitBoost and BagBoosting.

When we applied the non-replacement strategy to LogitBoost (Table 3) and BagBoosting (Table 5) we found that it gave the lowest error rates for all datasets. Thus the non-replacement strategy always gave the lowest error rate for each combination of dataset and boosting algorithm.

TABLE 2
Best result obtained with each dataset using AdaBoost.

Dataset	Number of features pre-selected	Mean error With replacement best	With replacement final	Non-replacement best	Non-replacement final
Leukemia	100	0.0425	0.0608	**0.0117**	0.0275
Colon	100	0.1505	0.2095	**0.1467**	0.1914
Prostate	500	0.0576	0.0835	**0.0500**	0.0788
Lymphoma	500	0.0086	0.0257	**0.0060**	0.0159
SRBCT	50	0.0158	0.0253	**0.0042**	0.0164
Brain	200	0.0533	0.0967	**0.0448**	0.0893

(Lowest error rate for each dataset in **bold**.)

TABLE 3
Best result obtained with each dataset using LogitBoost.

Dataset	Number of features pre-selected	Mean error With replacement best	With replacement final	Non-replacement best	Non-replacement final
Leukemia	100	0.0475	0.0583	**0.0183**	0.0317
Colon	All	0.1610	0.1991	**0.1543**	0.1943
Prostate	500	0.0559	0.0918	**0.0535**	0.0800
Lymphoma	200	0.0143	0.0216	**0.0057**	0.0114
SRBCT	50	0.0198	0.0307	**0.0052**	0.0169
Brain	100	0.0737	0.1186	**0.0660**	0.1057

(Lowest error rate for each dataset in **bold**.)

TABLE 4
Comparison of non-replacement strategies used with BagBoosting

Dataset	Mean error With replacement best	With replacement final	Non-replacement best	Non-replacement final	Full non-replacement best	Full non-replacement final
Leukemia	0.0417	0.0475	**0.0208**	0.0308	0.0233	0.0300
Colon	0.1581	0.1772	**0.1495**	0.1800	**0.1495**	0.1714
Prostate	0.0523	0.0670	**0.0500**	0.0606	0.0512	0.0670
Lymphoma	0.0206	0.0206	**0.0133**	0.0152	0.0181	0.0206
SRBCT	0.0300	0.0317	**0.0209**	0.0255	0.0298	0.0307
Brain	0.1077	0.1220	**0.0355**	0.0613	0.1111	0.1237

(Non-replacement = single feature removed at each iteration, full non-replacement = all features used at each iteration removed. Lowest error rate for each dataset in **bold**.)

TABLE 5
Best result obtained with each dataset using BagBoosting.

Dataset	Number of features pre-selected	Mean error With replacement best	With replacement final	Non-replacement best	Non-replacement final
Leukemia	50	0.0558	0.0609	**0.0225**	0.0275
Colon	200	0.1543	0.1800	**0.1429**	0.1714
Prostate	All	0.0523	0.0670	**0.0500**	0.0606
Lymphoma	All	0.0206	0.0206	**0.0133**	0.0152
SRBCT	200	0.0381	0.0409	**0.0000**	0.0070
Brain	All	0.1077	0.1220	**0.0355**	0.0613

(Lowest error rate for each dataset in **bold**.)

TABLE 6
Best number of preselected features

Dataset	Best number of features with AdaBoost	LogitBoost	BagBoosting
Leukemia	100	100	50
Colon	200	All	200
Prostate	500	500	All
Lymphoma	500	200	All
SRBCT	50	50	200
Brain	200	100	All

(Best number of features = number of genes preselected that resulted in the lowest value for the mean of the final error rate.)

TABLE 7
Classifier giving the lowest prediction error.

Dataset	Best	Final
Leukemia	AdaBoost	AdaBoost/BagBoosting
Colon	BagBoosting	BagBoosting
Prostate	AdaBoost/BagBoosting	BagBoosting
Lymphoma	LogitBoost	LogitBoost
SRBCT	BagBoosting	BagBoosting
Brain	BagBoosting	BagBoosting

(Non-replacement strategy gave lowest error for all dataset/classifier combinations.)

B. *Classification strategy*

From the full results tables (Tables 8–13) it can be seen that in the vast majority of cases applying the non-replacement strategy gave a lower error rate than not applying it. Also, for every combination of dataset and boosting algorithm the best result was obtained when the non-replacement strategy was applied.

For the final accuracy (i.e. after 25 iterations of boosting) BagBoosting gave the lowest error rates on almost all datasets. For the best performance (i.e. the lowest error obtained during all the iterations) no algorithm had a better performance overall, although LogitBoost only outperformed AdaBoost and BagBoosting on one dataset. Because of the difficulties in deciding where to stop boosting (see Section III C) we consider that the final error level is the best measure for comparing the performance of the different algorithms. On this basis, BagBoosting with non-replacement gave the best performance overall.

For most of the experiments performed using BagBoosting the non-replacement strategy used was to remove one feature at each iteration, rather than all features used at that iteration. We found that more accurate results were obtained with BagBoosting when one feature was removed at each iteration: there was no advantage in removing all features used at each iteration, and for most datasets the accuracy was similar to when features were replaced. Even where removing all features gave better performance than replacement, the accuracy was no better than that obtained by removing one feature each iteration. This is understandable because with a large number of features being removed at each iteration, most of the relevant features are removed very quickly, so that at the later stages of boosting, when the algorithm is concentrating on the more difficult samples, there are few relevant features left. Therefore, the models produced in the later stages of boosting are of low accuracy and cannot improve the performance of the ensemble.

C. *Feature preselection*

The number of preselected features (genes) that gave the lowest error rate varied between datasets and with the algorithm used: there was no optimum value. The fact that for several dataset/classifier combinations, the lowest classification error rate was obtained when all the genes were available suggests that for these datasets the preselection method was not able to identify the most relevant genes. Of the many methods that have been used for feature preselection we employed the Wilcoxon-Mann-Whitney U-test. Lee *et al.* [16] compared three preselection techniques in combination with several classification methods including AdaBoost and LogitBoost. They noted that the preselection method employed can have a significant effect on the performance of the classification methods. They concluded that while the Wilcoxon-Mann-Whitney U-test gave the best results for classical classification techniques, the soft-thresholding method [17] was better when ensemble techniques were used. However, they always preselected 50 genes and did not examine the effects of varying the number of features preselected. In addition, their results are not directly comparable with ours because they used CART, not decision stumps as the base learner for their ensemble methods.

V. Conclusions

The feature non-replacement strategy gave lower error rates than when replacement was employed, for all three boosting algorithms and all six datasets. Thus, we conclude that feature non-replacement is an excellent method for enhancing the performance of boosting algorithms on DNA microarray data.

When the non-replacement strategy was employed, BagBoosting gave a better final accuracy than AdaBoost or LogitBoost for the majority of datasets. It would therefore appear to be the best boosting algorithm for DNA microarray analysis of the three used. In this study we applied the non-replacement strategy to BagBoosting by either not replacing one feature, or not replacing any features used at each iteration. Between these two extremes there may be a level of replacement that gives minimum error rates. Further investigation may be able to determine what is the optimum degree of replacement to use with BagBoosting.

We found that there was no optimum number of features to preselect. The optimum method of feature preselection for microarray data and the optimum number of features to preselect are areas where further work is needed.

The error rates varied considerably between datasets, and for some datasets all the classifiers had high error rates. While this may be due to high levels of noise in the data, it may also indicate that there were feature interactions that the base learner was unable to model. For this study, in common with [1], [5] and [8] we used decision stumps as the base learner, more complex models that can take feature interaction into account may give better results and their use should be explored.

While the non-replacement strategy appears to be particularly suited to microarray data, because of their large number of attributes in relation to the number of samples, it may also be beneficial for data where the number of attributes is lower then the number of samples, and this should be investigated.

Thus, areas where further research is needed are:

1. The optimum level of feature replacement to employ with BagBoosting.
2. The best methods for feature preselection to use when boosting techniques are applied to microarray data.
3. Whether more complex base learners can give better results than decision stumps.
4. How effective the non-replacement strategy would be for data where the number of attributes is lower than the number of samples

Appendix

The full results tables are given here for each dataset.

TABLE 8
Result obtained with Leukemia dataset.

Algorithm	Number of features pre-selected	Mean error			
		With replacement		Non-replacement	
		best	final	best	final
AdaBoost	50	0.0433	0.0600	0.0117	0.0342
	100	0.0425	0.0608	0.0117	0.0275
	200	0.0425	0.0583	0.0133	0.0292
	500	0.0442	0.0625	0.0142	0.0375
	All	0.0458	0.0642	0.0175	0.0342
LogitBoost	50	0.0483	0.0592	0.0183	0.0342
	100	0.0475	0.0583	0.0183	0.0317
	200	0.0483	0.0608	0.0208	0.0367
	500	0.0475	0.0625	0.0192	0.0333
	All	0.0483	0.0625	0.0175	0.0342
BagBoosting	50	0.0558	0.0609	0.0225	0.0275
	100	0.0592	0.0617	0.0242	0.0283
	200	0.0600	0.0625	0.0225	0.0292
	500	0.0608	0.0667	0.0225	0.0317
	All	0.0417	0.0475	0.0208	0.0308

TABLE 9
Result obtained with Colon dataset.

Algorithm	Number of features pre-selected	Mean error			
		With replacement		Non-replacement	
		best	final	best	final
AdaBoost	50	0.1638	0.2133	0.1533	0.1991
	100	0.1505	0.2095	0.1467	0.1914
	200	0.1486	0.2038	0.1448	0.1943
	500	0.1495	0.2133	0.1476	0.2086
	All	0.1514	0.2076	0.1543	0.1943
LogitBoost	50	0.1667	0.2219	0.1562	0.2000
	100	0.1572	0.2010	0.1505	0.1972
	200	0.1572	0.2010	0.1505	0.1972
	500	0.1610	0.2076	0.1562	0.1953
	All	0.1610	0.1991	0.1543	0.1943
BagBoosting	50	0.1543	0.2000	0.1591	0.1886
	100	0.1562	0.1857	0.1448	0.1734
	200	0.1543	0.1800	0.1429	0.1714
	500	0.1553	0.1800	0.1476	0.1753
	All	0.1581	0.1772	0.1495	0.1800

TABLE 10
Result obtained with Prostate dataset.

Algorithm	Number of features pre-selected	Mean error			
		With replacement		Non-replacement	
		best	final	best	final
AdaBoost	50	0.0741	0.1106	0.0670	0.0976
	100	0.0611	0.1006	0.0606	0.0935
	200	0.0594	0.0953	0.0565	0.0912
	500	0.0576	0.0835	0.0500	0.0788
	All	0.0547	0.0923	0.0529	0.0835
LogitBoost	50	0.0729	0.1200	0.0676	0.0982
	100	0.0612	0.0923	0.0617	0.0882
	200	0.0606	0.0947	0.0564	0.0859
	500	0.0559	0.0918	0.0535	0.0800
	All	0.0570	0.0870	0.0529	0.0835
BagBoosting	50	0.0594	0.0888	0.0570	0.0965
	100	0.0588	0.0782	0.0553	0.0817
	200	0.0547	0.0753	0.0529	0.0682
	500	0.0517	0.0723	0.0512	0.0647
	All	0.0523	0.0670	0.0500	0.0606

TABLE 11
Result obtained with Lymphoma dataset.

Algorithm	Number of features pre-selected	Mean error			
		With replacement		Non-replacement	
		best	final	best	final
AdaBoost	50	0.0124	0.0286	0.0083	0.0184
	100	0.0133	0.0286	0.0083	0.0178
	200	0.0105	0.0267	0.0070	0.0162
	500	0.0086	0.0257	0.0060	0.0159
	All	0.0232	0.0330	0.0089	0.0216
LogitBoost	50	0.0152	0.0273	0.0067	0.0124
	100	0.0159	0.0244	0.0060	0.0133
	200	0.0143	0.0216	0.0057	0.0114
	500	0.0146	0.0232	0.0063	0.0140
	All	0.0260	0.0305	0.0102	0.0197
BagBoosting	50	0.0165	0.0292	0.0200	0.0244
	100	0.0146	0.0251	0.0181	0.0206
	200	0.0130	0.0219	0.0181	0.0197
	500	0.0130	0.0254	0.0197	0.0216
	All	0.0206	0.0206	0.0133	0.0152

TABLE 12
Result obtained with SRBCT dataset.

Algorithm	Number of features pre-selected	Mean error			
		With replacement		Non-replacement	
		best	final	best	final
AdaBoost	50	0.0158	0.0253	0.0042	0.0164
	100	0.0166	0.0276	0.0048	0.0189
	200	0.0164	0.0312	0.0069	0.0246
	500	0.0169	0.0310	0.0070	0.0257
	All	0.0271	0.0417	0.0098	0.0307
LogitBoost	50	0.0198	0.0307	0.0052	0.0169
	100	0.0214	0.0307	0.0069	0.0186
	200	0.0221	0.0352	0.0086	0.0207
	500	0.0238	0.0390	0.0093	0.0224
	All	0.0290	0.0395	0.0138	0.0269
BagBoosting	50	0.0340	0.0365	0.0013	0.0092
	100	0.0387	0.0425	0.0013	0.0073
	200	0.0381	0.0409	0.0000	0.0070
	500	0.0413	0.0457	0.0016	0.0121
	All	0.0300	0.0317	0.0209	0.0255

TABLE 13
Result obtained with Brain dataset.

Algorithm	Number of features pre-selected	Mean error			
		With replacement		Non-replacement	
		best	final	best	final
AdaBoost	50	0.0567	0.1045	0.0517	0.0933
	100	0.0540	0.1017	0.0483	0.0912
	200	0.0533	0.0967	0.0448	0.0893
	500	0.0538	0.0950	0.0479	0.0902
	All	0.0754	0.1249	0.0631	0.1151
LogitBoost	50	0.0732	0.1211	0.0671	0.1089
	100	0.0737	0.1186	0.0660	0.1057
	200	0.0731	0.1197	0.0671	0.1123
	500	0.0734	0.1191	0.0683	0.1072
	All	0.0874	0.1209	0.0809	0.1143
BagBoosting	50	0.0826	0.0943	0.0660	0.0869
	100	0.0810	0.0886	0.0650	0.0812
	200	0.0857	0.0921	0.0586	0.0737
	500	0.0786	0.0831	0.0626	0.0757
	All	0.1077	0.1220	0.0355	0.0613

REFERENCES

[1] A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, and Z. Yakhini, “Tissue Classification with Gene Expression Profiles,” *Journal of Computational Biology*, vol. 7, pp. 559–584, 2000.

[2] Y. Freund, R. Schapire, and N. Abe, “A Short Introduction to Boosting,” *Journal of the Japanese Society for Artificial Intelligence*, vol. 14, pp. 771–780, 1999.

[3] S. Dudoit, J. Fridlyand, and T. P. Speed, “Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data,” *Journal of the American Statistical Association*, vol. 97, pp. 77–87, 2002.

[4] L. Breiman, “Arcing classifiers,” *Annals of Statistics*, vol. 26, pp. 801–823, 1998.

[5] M. Dettling and P. Bühlmann, “Boosting for tumor classification with gene expression data,” *Bioinformatics*, vol. 19, pp. 1061–1069, 2003.

[6] N. Friedman, M. Linial, I. Nachman, and D. Pe’er, “Using Bayesian Networks to Analyze Expression Data,” *Journal of Computational Biology*, vol. 7, pp. 601–620, 2000.

[7] P. M. Long and V. B. Vega, “Boosting and Microarray Data,” *Machine Learning*, vol. 52, pp. 31–44, 2003.

[8] M. Dettling, “BagBoosting for tumor classification with gene expression data,” *Bioinformatics*, vol. 20, pp. 3583–93, 2004.

[9] U. Alon *et al.*, “Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays,” *Proceedings of the National Academy of Sciences USA*, vol. 96, pp. 6745–6750, 1999.

[10] P. J. Park, M. Pagano, and M. Bonetti, “A nonparametric scoring algorithm for identifying informative genes from microarray data,” *Pac. Symp. Biocomput.*, pp. 52–63, 2001.

[11] S. L. Pomeroy *et al.*, “Prediction of central nervous system embryonal tumour outcome based on gene expression,” *Nature*, pp. 436–441, 2002.

[12] T. R. Golub *et al.*, “Molecular classification of cancer: class discovery and class prediction by gene expression monitoring,” *Science*, vol. 286, pp. 531–7, 1999.

[13] A. A. Alizadeh *et al.*, “Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling,” *Nature*, pp. 503–511, 2000.

[14] D. Singh *et al.*, “Gene expression correlates of clinical prostate cancer behavior,” *Cancer Cell*, vol. 1, pp. 203–209, 2002.

[15] J. Khan *et al.*, “Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks,” *Nature Medicine*, vol. 7, pp. 673–9, 2001.

[16] J. W. Lee, J. B. Lee, M. Park, and S. H. Song, “An extensive comparison of recent classification tools applied to microarray data,” *Comp. Statistics and Data Analysis*, vol. 48, pp. 869–885, 2005.

[17] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu, “Diagnosis of multiple cancer types by shrunken centroids of gene expression,” *Proceedings of the National Academy of Sciences USA*, vol. 99, pp. 6567–6572, 2002.