IPSJ Transactions on Bioinformatics Vol. 3 54–61 (June 2010)

Original Paper

Sparse Learner Boosting for Gene Expression Data

Mari Pritchard[†1]

Gene expression analysis is commonly used to analyze millions of gene expression data points. Challenging in this process has been the development of appropriate statistical methods for high-dimensional data. We propose Sparse Learner Boosting for gene expression data analysis. Boosting is performed to minimize the loss function, although this process can cause overfitting when a large number of variables are present. Ordinary boosting utilizes all of the potential weak learners in a given data set and constructs a decision rule. The fundamental idea of Sparse Learner Boosting is to reduce the complexity of the decision rule by using fewer weak learners than is usually required. This reduction prevents overfitting and improves performance during classification. Numerical studies support this modification for high-dimensional data, such as that obtained from gene expression analysis. We show that the proposed modification improves the performance of ordinary boosting methods.

1. Introduction

Microarray technology is commonly used for the analysis of gene expression data in many laboratories. Although this technology enables researchers to collect millions of gene expression data points quickly, analyzing many subjects simultaneously remains difficult; normally, the maximum number of samples is in the order of 100. This extremely unbalanced ratio of dimension p to sample size n makes the application of classic statistical methods challenging without specific modifications. Hastie and Tibshirani [1] referred to this as the $p \gg n$ problem, as one of the most challenging issues in bioinformatics.

We focus on the problem of sample classification using biomarkers from gene expression data. Technological advances have enabled researchers to use gene expression data for clinical purposes. This type of application was first reported by Golub, et al. [2], who categorized leukemia samples into subclasses. Later, Bittner, et al. [3] used microarrays to subgroup melanomas. van't Veer, et al. [4] published a study using microarrays to classify metastasis from primary breast cancer patients.

Boosting, which is introduced by Freund and Schapire [5] as one of the most powerful methods for machine learning along with Support Vector Machine which was reported by Vapnik [6]. The underlying idea is that many "weak" classifiers can be combined to build a "strong" classifier at the end of a series of learning steps. There are a number of boosting algorithms proposed, the most popular of which is AdaBoost, which is also proposed by Freund and Schapire [5]. Dudoit, et al. [7] compared AdaBoost to other classifiers using gene expression data analysis and they concluded that AdaBoost did not perform at a comparable level. For high-dimensional data, however, algorithms need to be modified to accommodate this situation.

In this paper, we propose Sparse Learner Boosting, which can be used with high-dimensional data. The key idea is to reduce the number of weak learner candidates. If you consider a set of decision stumps, the set is fully constructed by weak learners which are based on given features. We face a situation in which the number of features is extremely large, consequently there is a superfluous number of weak learners complicating the classification model. We propose truncating the candidate of weak learners for learning while keeping informative weak learners.

The number of learning steps can cause overfitting during boosting as well. Zhang and Yu [8] concluded that stopping the process early prevents overfitting, however early stopping does not give any reasonable performance if the size of the feature dimension is large or two class data is strongly overlapped. In these cases, the boosting method needs more modification. Sparse Boosting was first proposed by Bühlmann and Yu [9] as a variant boosting algorithm for high-dimensional data. Sparse Boosting yields sparser solutions than L_2 Boosting. We will discuss this point further later.

In this paper, we begin with an overview of AdaBoost and η-Boost before proposing Sparse Learner Boosting. Next we perform a simulation study to examine the performance of Sparse Learner Boosting. We conclude with an analysis of real data and a discussion of future work.

†1 The Graduate University for Advanced Studies

2. Sparse Learner Boosting

We first prepared a standard framework for the classification procedure. We let $\boldsymbol{x}$ be a feature vector of p dimensions and let y be a class label with values of ± 1. The given training data was $\{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, N\}$. The function of $\boldsymbol{x}$ into y, $f(\boldsymbol{x})$ is referred to as a classifier.

2.1 AdaBoost

AdaBoost algorithm is used to construct a "strong" classifier as a linear combination of "weak" learners. The training data is sequentially reweighted and the final classifier is based on a weighted vote of the weak classifiers. We used a set of classifiers defined as follows:

$$\mathcal{F} = \{f_j(\boldsymbol{x}) : j \in 1, \ldots, p\}. \tag{1}$$

Then, the final classifier was constructed using the weak classifiers,

$$F(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t f_t(\boldsymbol{x}). \tag{2}$$

The coefficients α_t and classifiers f_t are defined in the following discussion.

AdaBoost algorithm is characterized by the minimization of the exponential loss function, which is defined as

$$L_{\exp}(F) = \sum_{i=1}^{N} \exp(-y_i F(\boldsymbol{x}_i)). \tag{3}$$

Consider an update from F to $F + \alpha f$, the exponential loss function can be written as,

$$L_{\exp}(F + \alpha f) = \sum_{i=1}^{N} \exp(-y_i(F(\boldsymbol{x}_i) + \alpha f(\boldsymbol{x}_i))) \tag{4}$$

$$= L_{\exp}(F)\{\varepsilon(f)e^{\alpha} + (1 - \varepsilon(f))e^{-\alpha}\}, \tag{5}$$

where $\varepsilon(f)$ is weighted error rate and defined as

$$\varepsilon(f) = \frac{\sum_{i=1}^{N} I(y_i \neq f(\boldsymbol{x}_i)) \exp(-y_i F(\boldsymbol{x}_i))}{L_{\exp}(F)}. \tag{6}$$

In addition, α is calculated by

$$\arg\min_{\alpha \in \mathbb{R}} L_{\exp}(F + \alpha f) = \frac{1}{2} \log \frac{1 - \varepsilon(f)}{\varepsilon(f)}. \tag{7}$$

The optimal value of f is determined by minimizing the weighted error $\varepsilon(f)$. The details of the algorithm are as follows:

1. Set $w_1(i) = 1/N$ and $F_0 = 0$

2. For any $t = 1, \ldots, T$

a. Find

$$f_t = \arg\min_{f \in \mathcal{F}} \ \varepsilon(f) \tag{8}$$

where

$$\varepsilon_t(f_t) = \frac{\sum_{i=1}^{N} w_t(i) I(y_i \neq f_t(\boldsymbol{x}_i))}{\sum_{i'=1}^{N} w_t(i')}. \tag{9}$$

b. Calculate

$$\alpha_t = \frac{1}{2} \log \frac{1 - \varepsilon_t(f_t)}{\varepsilon_t(f_t)}. \tag{10}$$

c. Update

$$w_{t+1}(i) = w_t(i) \exp(-\alpha_t y_i f_t(\boldsymbol{x}_i)). \tag{11}$$

3. The final classifier is given by $\mathrm{sgn}\Big(\sum_{t=1}^{T} \alpha_t f_t(\boldsymbol{x})\Big)$.

2.2 η-Boost

η-Boost is proposed by Takenouchi and Eguchi [10] as a robust boosting algorithm. AdaBoost reportedly can be easily influenced by outliers, which breaks down the performance. η-Boost is defined by the loss function using the following a mixture of the exponential loss and naive error loss functions as

$$L_{\eta}(F) = \sum_{i=1}^{N} [(1 - \eta) \exp(-y_i F(\boldsymbol{x}_i)) - \eta y_i F(\boldsymbol{x}_i)]. \tag{12}$$

where $0 \leq \eta \leq 1$. η-Boost algorithm is written as follows.

1. Set $w_1^*(i) = 1/N$ and $F_0 = 0$

2. For any $t = 1, \ldots, T$

a. Find

$$f_t = \arg\min_{f \in \mathcal{F}} \ \varepsilon^*(f) \tag{13}$$

where

$$\varepsilon_t^*(f_t) = \sum_{i=1}^{N} w_t^*(i) I(y_i \neq f_t(\boldsymbol{x}_i)). \tag{14}$$

b. Calculate

$$\alpha_t^* = \log \frac{\sqrt{1-\varepsilon_t(f_t)+(\eta K_t)^2}+\eta K_t}{\sqrt{\varepsilon_t(f_t)}} \tag{15}$$

where

$$K_t = \frac{(1-2\varepsilon_1(f_t))}{2\sqrt{\varepsilon_t(f_t)}} \left(\frac{(1-\eta)Z_t}{N}\right)^{-1} \tag{16}$$

with

$$Z_{t+1} = \sum_{i=1}^{N} \exp(-y_i F_t(\boldsymbol{x}_i)). \tag{17}$$

c. Update

$$w_{t+1}^*(i) = \frac{(1-\eta)\exp(-y_i F_t(\boldsymbol{x}_i))+\eta}{Z_{t+1}^*} \tag{18}$$

where

$$F_t(\boldsymbol{x}) = \sum_{m=1}^{t} \alpha_m^* f_m(\boldsymbol{x}) \tag{19}$$

$$Z_{t+1}^* = \sum_{i=1}^{N} (1-\eta)\exp(-y_i F_t(\boldsymbol{x}_i)) + \eta. \tag{20}$$

3. The discriminant function is $\operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t^* f_t(\boldsymbol{x})\right)$.

The derivations for Eqs. (14) and (15) are written in the Appendix.

2.3 Sparse Learner Boosting

We present here the details of Sparse Learner Boosting. Before we describe further details, we define weak learners. We use decision stumps as weak learner candidates, which are defined as

$$\mathcal{F} = \{f_j(\boldsymbol{x}, a, b) = a \cdot \operatorname{sgn}(x_j - b) : j \in \{1, \cdots, p\}, b \in \mathbb{R}\}, \tag{21}$$

where a is given a value 1 or -1 and $b \in \mathbb{R}$ is a threshold value. The threshold b is calculated using the following procedure. First, x_{j1} through x_{jp} of the jth feature vector are sorted as follows:

$$x_{(1)j} \leq x_{(2)j} \leq \cdots \leq x_{(n)j}. \tag{22}$$

Based on $x_{(i)j}$, we defined $(n-1)$ thresholds to satisfy below,

$$x_{(1)j} < b_{(1)j} < x_{(2)j} < \cdots < b_{(n-1)j} < x_{(n)j}. \tag{23}$$

For simplicity we specify $b_{(i)j} = (x_{(i)j} + x_{(i+1)j})/2$. The set of $b_{(i)j}$ is defined as

$$\mathcal{B} = \left\{b_{(1)j}, \ldots, b_{(n-1)j}\right\}. \tag{24}$$

The idea of Sparse Learner Boosting is to reduce the number of weak learner candidates. If two class distributions are strongly overlapped, the Boosting algorithm learns from the overlapped samples in a greedy way. The discriminant function becomes complex because of overfitting. Sparse Learner Boosting controls the complexity by reducing the number of weak learner candidates using false positive rate $\overline{\mathrm{FP}}(f)$ and false negative rate $\overline{\mathrm{FN}}(f)$ which are defined as

$$\overline{\mathrm{FP}}(f) = \frac{1}{n_1} \sum_{i=1}^{n} I(f(\boldsymbol{x}_i) \neq y_i, y_i = -1) \tag{25}$$

$$\overline{\mathrm{FN}}(f) = \frac{1}{n_2} \sum_{i=1}^{n} I(f(\boldsymbol{x}_i) \neq y_i, y_i = +1) \tag{26}$$

where n_1 is the number of subjects having class label -1 and n_2 having class label $+1$. Let $\mathcal{F}_\xi$ be a reduced set of weak learners which is defined as

$$\mathcal{F}_\xi = \{f : \min(\overline{\mathrm{FP}}(f), \overline{\mathrm{FN}}(f)) \leq \xi, f \in \mathcal{F}, \xi \in \mathbb{R}\}. \tag{27}$$

Equation (27) is the criterion to trim weak learners, in which ξ controls the sparseness. We refer to this criterion as trim overlapping learners criterion (TOL criterion). **Figure 1** shows an example of weak learners for both ordinary Boosting and Sparse Learner Boosting. The left panel shows the weak learner candidates for ordinary Boosting and the right panel is for Sparse Learner Boosting. It can be seen that the weak learners at the overlapped area are trimmed.

We integrated Sparse Learner Boosting into AdaBoost and η-Boost. Sparse Learner Boosting using AdaBoost and η-Boost is referred to as Sparse AdaBoost and Sparse η-Boost, respectively. The procedure for calculating α is the same as in AdaBoost and η-Boost. The discriminant function,

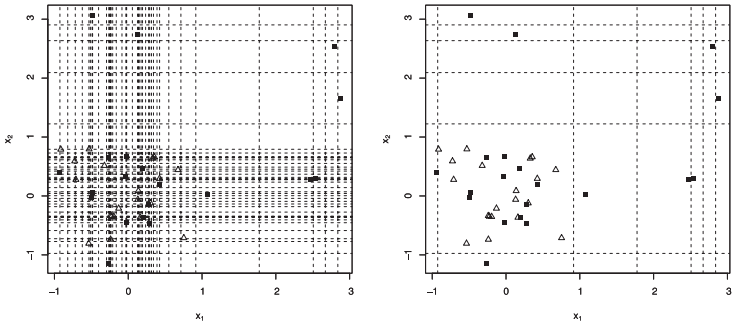


Fig. 1 Typical example for weak learners. The dashed lines denote decision stumps. The left panel showed the weak learners for ordinary boosting. The right panel is for Sparse Learner Boosting where $\xi = 0$.

$$F(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t f_t(\boldsymbol{x}) \tag{28}$$

gives the classifier $g(\boldsymbol{x}) = \mathrm{sgn}(F(\boldsymbol{x}) - c^*)$, where c^* is the optimum threshold which is difined as follows,

$$c^* = \arg\min_{c} \sum_{i=1}^{n} I(\mathrm{sgn}(F(\boldsymbol{x}_i) - c) \neq y_i). \tag{29}$$

We next compare the complexity of ordinary Boosting with Sparse Learner Boosting. We use Vapnik-Chervonenkis dimension (VC dimension) which is introduced by Vapnik and Chervonenkis [11]. VC dimension is defined as the cardinality of the largest set of points that the weak learners can shatter for any labeling of $\{x_1, \ldots, x_n\}$. The VC dimension is denoted as VC and the set of decision stumps is referred to as $\mathcal{F}_{ds}$. The VC shatter coefficient $S_n(\mathcal{F}_{ds})$ is defined as

$$S_n(\mathcal{F}_{ds}) = \max_{\{x_1,\ldots,x_n\}\in\mathbb{R}^p} |\{x_1, \ldots, x_n\} \cap \{x | f(\boldsymbol{x}, a, b) = 1 | f(\boldsymbol{x}, a, b) \in \mathcal{F}_{ds}\}|. \tag{30}$$

The VC dimension of $\mathcal{F}_{ds}$ is defined as

$$\mathrm{VC}(\mathcal{F}_{ds}) = \max_{n'} \{n' | S_{n'}(\mathcal{F}_{ds}) = 2^{n'}\}. \tag{31}$$

For each feature vector x_j $(j = 1, \ldots, p)$, $\mathcal{F}_{ds}$ yields a maximum of $2n$ different subsets of $\{x_1, \ldots, x_n\}$ resulting in the following:

$$2pn \geq 2^n \tag{32}$$

From the above we have

$$\mathrm{VC}(\mathcal{F}_{ds}) < \lfloor 2(1 + \log_2 p) \rfloor, \tag{33}$$

where $\lfloor \cdot \rfloor$ is the floor function. See Kawakita and Eguchi [12] for detailed discussion.

We then calculate the VC dimension of Sparse Learner Boosting. The reduced weak learner candidates by TOL criterion is a subset of ordinary Boosting weak learner candidates. The number of weak learner candidates is defined by the number of samples and variables. The number of weak learner candidates is $p(n-1)$. The reduced weak learner candidates can be written $\delta p(n-1)$, $0 < \delta \leq 1$. Equation (32) for Sparse Learner Boosting is

$$2p\delta n \geq 2^n \tag{34}$$

The VC dimensions of the reduced weak learners $\mathrm{VC}(\mathcal{F}_\xi)$ is

$$\mathrm{VC}(\mathcal{F}_\xi) < \lfloor 2(1 + \log_2 \delta p) \rfloor. \tag{35}$$

Equation (35) expresses that weak learners trimed by TOL criterion are equivalent to the δp variable selection. We also observe that the upper bound of $\mathrm{VC}(\mathcal{F}_\xi)$ is less than that of $\mathrm{VC}(\mathcal{F}_{ds})$. This means that Sparse Learner Boosting is less complex model than ordinary boosting methods.

3. Simulation

In this section, we describe a number of simulation studies using synthetic and real data. First, we compare the boundaries of AdaBoost and Sparse AdaBoost. Then we show several synthesized and real data analysis results to compare the performance of the proposed method with conventional boosting methods.

3.1 Case 1. Synthetic Data with Two-dimensional Feature Vectors

For comparing decision boundaries by ordinary boosting and Sparse Learner Boosting, two variable data sets were generated. The data was defined as $\{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, 100\}$ and $\boldsymbol{x}_i = (x_{1i}, x_{2i})$. The feature vectors with class label $+1$ follow the normal distribution $N((1,1), \boldsymbol{I})$. The feature vectors with class label -1 follow the mixture distribution of $N((0,0), \boldsymbol{I})$ and $N((4,4), \boldsymbol{I})$. The prior probability of each distribution is 0.9 and 0.1 respectively. In **Fig. 2**,

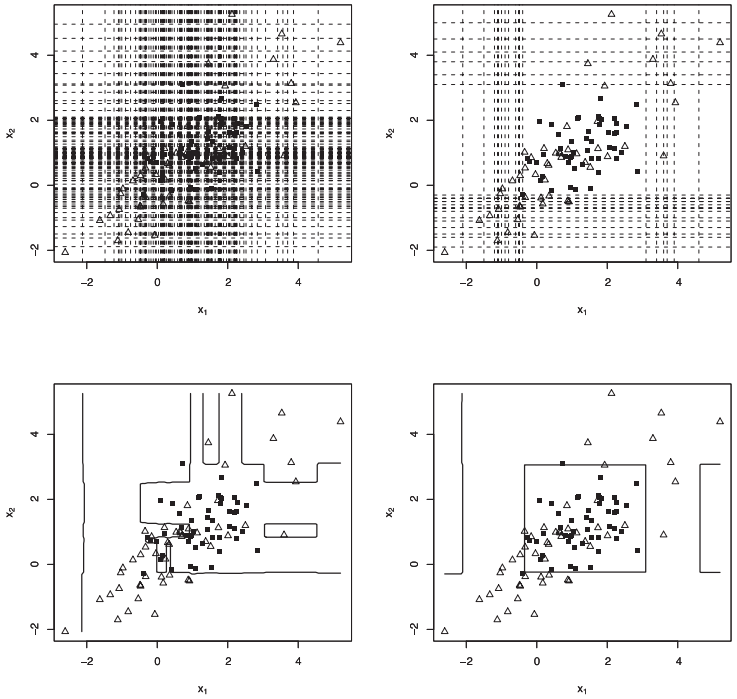


Fig. 2 The data generated by case 1. Squares and triangles denote data points with class labels +1 and −1 respectively. The upper left panel is weak learners for AdaBoost. The upper right panel shows weak learners for Sparse Learner AdaBoost. The lower left panel shows the boundary which was built by AdaBoost. The lower right panel is the boundary for Sparse Learner AdaBoost.

we show the boundaries for AdaBoost and Sparse AdaBoost. We also show weak learners for each method. The squares and triangles denote feature vectors with labels +1 and −1, respectively. The upper two panels are weak learners for AdaBoost and Sparse Learner Boosting. The lower two panels are the boundaries after the learning steps. The number of iterations was decided by cross validation.

We confirmed that Sparse AdaBoost results in a simple boundary in comparison to AdaBoost. The panels of weak learners suggest that AdaBoost could construct a complex boundary with too many weak learners.

Table 1 Test error rates for multivariate data. The number of variables, $p = (10, 100, 300, 500)$ were used. The number of iteration was decided by three fold cross validation. The parameter η for η-boost was fixed to 0.1.

		Variables			
		10	100	300	500
		Test Error	Test Error	Test Error	Test Error
AdaBoost	Sparse	0.136	0.246	0.268	0.266
	non-Sparse	0.311	0.378	0.349	0.347
η-Boost	Sparse	0.136	0.243	0.232	0.288
	non-Sparse	0.309	0.422	0.424	0.435

3.2 Case 2. Synthetic Multivariate Data

Next, we investigate the performance of Sparse Learner Boosting with conventional Boosting methods. We set several multivariate data sets in a setting of $p \gg n$. All feature vectors with class label $y = +1$ follow the normal distribution $N(\boldsymbol{\mu}_{+1}, 0.5\boldsymbol{I}_p)$. The mean vector $\boldsymbol{\mu}_{+1} = (0.1, \ldots, 0.1)^T$. All feature vectors with class label $y = -1$ were generated from the normal mixture distribution which has a probability density $p(\boldsymbol{x}) = (1-\pi)g(\boldsymbol{x}) + \pi h(\boldsymbol{x})$. The probability distribution of $g(\boldsymbol{x})$ is the normal distribution $N(\boldsymbol{\mu}_{-1}, 0.5\boldsymbol{I}_p)$ with mean vector $\boldsymbol{\mu}_{-1} = (0, \ldots, 0)^T$. The probability distribution of $h(\boldsymbol{x})$ is $N(\boldsymbol{\mu}_{\text{out}}, 0.5\boldsymbol{I}_p)$ with mean vector $\boldsymbol{\mu}_{\text{out}} = (3, \ldots, 3)^T$. The prior probability π is 0.1. We fixed the number of observations $n = 100$ and changed the number of variables as $p = (10, 100, 300, 500)$. Weak learners for Sparse Learner Boosting were determined by Eq. (27). We used $\xi = 0$. The number of iterations was decided by 3-fold cross validation. 3-fold cross validation was repeated 50 times and the number of iterations which showed the lowest cross validation error was used to predict class labels of the test data. We generated 10 data sets for both training data and test data, so that the average test error rate were calculated. To simplify comparisons, we set $\eta = 0.1$ for η-Boost.

Table 1 shows the result of the test error rate. Sparse Learning Boosting showed better performance across all cases. On the other hand, even when the number of iterations is decided by cross validation, non-Sparse AdaBoost and non-Sparse η-Boost could not show good predictive power.

4. Analysis of Real Data

We used the gene expression data reported by van't Veer, et al. [4]. This study included 97 primary breast cancer patients, of which 78 were used for training data to build the discriminant function. 19 were used as test data to evaluate the classifier. Distant metastasis was observed within 5 years in 34 patients out of the 78 patients for training data, whereas the remaining 44 patients were disease-free after at least 5 years. The dimension of feature vector was 24,481, or the number of genes.

Observation values were normalized by taking the logarithm of the ratio of individual RNA to pooled RNA during preprocessing. Filtering criteria (van't Veer, et al. [4]) were applied to reduce the dimensionality. Using two-fold change in the expression level with P-value < 0.01 in five patients or more of the 78 patients yieled approximately 5,000 genes. See Roberts (2000) for further details.

The correlation coefficient between the expression of each probe and disease outcome was calculated; 200 probes had less than 0.3 absolute value of correlation coefficients. Those 200 probes were considered to be significantly associated with the disease outcome. These 200 probes were determined by training data, test data was not used. The original data is available at http://www.rii.com/publications/default.htm.

The class label y_i associates with the period in which cancer recurrence occurred; the group of patients diagnosed with a recurrence within five years was assigned -1, whereas the group that did not show a recurrence within five years was represented by $+1$.

We applied Sparse Learner Boosting and non-Sparse Learner Boosting using 200 variables. The result was shown in **Table 2**. The number of iterations and ξ was determined by 3-fold cross validation. 3-fold cross validation was repeated 50 times. The lowest test error was shown in the table. We used $\eta = 0.1$ in this data as well. Sparse AdaBoost showed 0.158 test error. Neither non-Sparse AdaBoost nor non-Sparse η-Boost could achieve the low error rate given by Sparse Learner Boosting.

Table 2 Test error rates for real data. The variables were sorted by correlation to the class label. 200 variates were used.

		Variables
		200
AdaBoost	Sparse	0.158
	non-Sparse	0.316
η-Boost	Sparse	0.211
	non-Sparse	0.316

5. Conclusions and Future Work

Simple modification of the current boosting methods was performed, resulting in this Sparse Learner Boosting. We used Sparse Learner Boosting to analyze synthesized data and real gene expression data, which both confirmed that Sparse Learner Boosting improves classification performance.

We showed that Sparse Learner Boosting gives a drastic reduction in VC dimension compared to non-Sparse Learner Boosting. From the view point of feature reduction, reducing the size of p by ranking methods or regularization is common. On the other hand, we propose an efficient method to reduce weak learners, called Sparse Learner Boosting. We decided the number of iterations by cross validation in our simulation however we saw that classification performance was affected by sparseness, in which early stopping is not good enough to prevent overfitting. We note that the TOL criterion is useful to truncate the weak learners candidates when the two class data is heavily overlapped. If there is no overlap or a very small overlap, trimming weak learners candidates does not show performance improvement.

Sparse Boosting proposed by Bühlmann and Yu [9] is designed to find sparser solutions for high-dimensional data. These authors modified L_2 Boosting. L_2 Boosting minimizes the residual sum of squares, so that Sparse Boosting considers all previous boosting iterations in addition to current residuals. Our proposed method reduces the number of weak learner candidates and prevents overfitting which is applicable for a wide class of boosting methods including L_2 Boosting.

In conclusion, we propose a new boosting method, Sparse Learner Boosting, and confirm its ability to analyze high-dimensional data.

References

1) Hastie, T. and Tibshirani, R.: Efficient quadratic regularization for expression

arrays, *Biostatistics*, Vol.5, pp.329–340 (2004).
2) Golub, R., Slonim, K.,Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, R, Caligiuri, M., Bloomfield, D. and Lander, S.: Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring, *Science*, Vol.286, pp.531–537 (1999).
3) Bittner, M., Meltzer, P., Chen, Y., Jiang, Y., Seftor, E., Hendrix, M., Radmacher, M., Simon, R., Yakhini, Z., Ben-Dor, A., Sampas, N., Dougherty, E., Wang, E., Marincola, F., Gooden, C., Lueders, L., Glatfelter, A., Pollock, P., Carpten, J., Gillanders, E., Leja, D., Dietrich, K., Beaudry, C., Berens, M., Alberts, D., Sondak, V., Hayward, N. and Trent, J.: Molecular classification of cutaneous malignant melanoma by gene expression profiling, *Nature*, Vol.406, pp.536–540 (2000).
4) van't Veer, L., Dai, H., van de Vijver, M., He, Y., Hart, A., Mao, M., Peterse, H., van der Kooy, K., Marton, M., Witteveen, A., Schreiber, G., Kerkhoven, R., Roberts, C., Linsley, P., Bernards, R. and Friend, S.: Gene expression profiling predicts clinical outcome of breast cancer, *Nature*, Vol.415, pp.530–553 (2002).
5) Freund, Y. and Schapire, R.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting, *J. Comput. Syst. Sci.*, Vol.55, pp.119–139 (1997).
6) Vapnik, V.: *The Nature of Statistical Learning Theory*, Springer (1995).
7) Dudoit, S. and Fridlyand, J.: Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data, *J. Am. Stat. Assoc.*, Vol.97, pp.77–87 (2002).
8) Zhang, T. and Yu, B.: Boosting with early stopping: Convergence and consistency, *Annals of Statistics*, Vol.33, pp.1538–1579 (2005).
9) Bühlmann, P. and Yu, B.: Sparse Boosting, *The Journal of Machine Learning Research*, Vol.7, pp.1001–1024 (2006).
10) Takenouchi, T. and Eguchi, S.: Robustifying AdaBoost by Adding the Naive Error Rate, *Neural Computation*, Vol.16, pp.767–787 (2004).
11) Vapnik, V. and Chervonenkis, A.: On the uniform convergence of relative frequencies of events to their probabilities, *Theory of probability and its Applications*, Vol.16, pp.264–280 (1971).
12) Kawakita, M. and Eguchi, S.: Boosting method for local learning in statistical pattern recognition, *Neural Computation*, Vol.20, pp.2792–2838 (2008).
13) Roberts, C., Nelson, B., Marton, M., Stoughton, R., Meyer, M., Bennett, H., He, Y., Dai, H., Walker, W., Hughes, T., Tyers, M., Boone, C. and Friend, S.: Signaling and Circuitry of Multiple MAPK Pathways Revealed by a Matrix of Global Gene Expression Profiles, *Science*, Vol.287, pp.873–880 (2000).

Appendix

A.1 Derivation of the η-Boost Algorithm

We here show derivations of Eqs. (14) and (15) from the η-Boost algorithm. The η-Boost algorithm was derived by minimizing the loss function (12). Loss function (12) is rewritten as follows,

$$L_\eta(F+\alpha f) = \sum_{i=1}^{N}[(1-\eta)\exp\{-y_i(F(\boldsymbol{x}_i)+\alpha f)\} - \eta y_i(F(\boldsymbol{x}_i+\alpha f))]. \quad (36)$$

We define f_t to minimize the gradient of the loss function $L_\eta(F+\alpha f_t)$ at $\alpha = 0$

$$\frac{\partial}{\partial\alpha}L_\eta(F+\alpha f)|_{\alpha=0} = \sum_{i=1}^{N}[-y_i f(\boldsymbol{x}_i)\{(1-\eta)\exp(-y_i F(\boldsymbol{x}_i)) + \eta\}]. \quad (37)$$

We rewrite Eq. (37) using the indicated function as follows:

$$\begin{aligned}\frac{\partial}{\partial\alpha}L_\eta(F+\alpha_t f_t)|_{\alpha=0} &= \sum_{i=1}^{N}[-I\{y_i = f(\boldsymbol{x}_i)\}w_t + I\{y_i \neq f(\boldsymbol{x}_i)\}w_t] \\ &= 2\sum_{i=1}^{N}[w_t I\{y_i \neq f(\boldsymbol{x}_i)\}] - \sum_{i=1}^{N} w_t, \quad (38)\end{aligned}$$

where $w_t = (1-\eta)\exp(-y_i F(\boldsymbol{x}_i)) + \eta$. From Eq. (38), we find a value of f_t to minimize the weighted error rate. This is the derivative of Eq. (14) Next α_t is calculated by minimizing η-Loss as follows:

$$\alpha_t = \arg\min_{\alpha} \frac{\partial}{\partial\alpha}L_\eta(F+\alpha f), \quad (39)$$

which implies that α_t is a solution of the equation

$$\frac{\partial}{\partial\alpha}L_\eta(F+\alpha f) = 0. \quad (40)$$

The equation is written as follows:

$$(1-\eta)e^{-\alpha} - A + (1-\eta)e^{(+\alpha)}B - \eta C = 0$$

which is solved by

$$\alpha = \log\left\{\frac{\eta C}{2(1-\eta)B} + \sqrt{\frac{A}{B} + \left(\frac{\eta C}{(1-\eta)B}\right)}\right\}$$

where

$$A = \sum_{y_i f(\boldsymbol{x}_i)=1} e^{-y_i F(\boldsymbol{x}_i)}$$

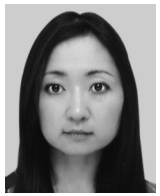
$$B = \sum_{y_i f(\boldsymbol{x}_i)=-1} e^{-y_i F(\boldsymbol{x}_i)}$$

$$C = 2\left(\sum_{y_i f(\boldsymbol{x}_i)=-1} +1\right) - N.$$

This is the derivation of Eqs. (14) and (15).

(Received November 24, 2009)
(Accepted April 5, 2010)
(Released June 17, 2010)

(Communicated by *Shigeyuki Oba*)



Mari Pritchard received her master degree from Kurume University in 2007. She is a currently Ph.D. candidate at the Graduate University of Advanced Studies. She worked for Rosetta Biosoftware as Asian Pacific Technical Account Manager from 2002 through 2009.