Boosting method via the sparse learner approach for high-dimensional gene expression data

DOCTOR OF PHILOSOPHY

Mari Pritchard

Department of Statistical Science

The Graduate University for Advanced Studies

School of 2007

Contents

Abstract

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. We also review another problem in high-dimensional data. Sparser solutions are desirable from the view point of simple classification modeling and ease of interpretation however there is no unique sparse solution in any single classification problem. The possible combination of gene sets out of millions of gene expression data is huge. We show the existence of multiple optimum gene sets and consider the possible solutions.

1 Introduction

In the last couple of decades, bioinformatics has showed impressive progress. During the Human Genome Project (Vender et al. (2001)), the speed and cost of reading genome sequence was accelerated that made the end of the project faster than the originally planned. After the project, more species genome were read using the improved technology. More than 1,000 species genomes have finished being read and we can access this genome data on the web site (www.genomesonline.org). Genome information can be utilized in a variety of ways. One technology which uses genome information is microarray technology which measures gene expression levels from collected RNA. Microarray becomes a common technology for its great of interest to observe millions of gene expression behavior.

Microarray has various usages, one of them is in clinical research. Researchers collect gene expression data from patients so that they can examine it with their clinical information. Golub et al. (1999) categorized leukemia samples into subclasses using microarray and their clinical outcome information. Bittner et al. (2000) used gene expression data to subgroup melanomas. van't Veer et al. (2002) published a study to classify metastasis from primary breast cancer patients then their selected genes were used in the first FDA approved microarray diagnosis kit.

The progress of microarray technology has generated a new challenge in statistics and machine learning. Microarray can measure millions of gene expression data in one experiment however still the number of subjects is in the order of hundreds. This extremely unbalanced ratio of dimension p to sample size n makes the application of classic statistical methods challenging without specific modification.

Boosting is introduced by Freund and Schapire (1997) as one of the most powerful methods for machine learning along with Support Vector Machine which is reported by Vapnik (1995). The underlying idea is that many“ weak ”classifiers are 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 (1997). AdaBoost learning process proceeds sequentially while minimizing exponential loss each step. Observations are weighted each step so that AdaBoost can learn from the data points which are misclassified in the previous step.

Dudoit and Fridlyand (2002) compared AdaBoost to other classifiers using the gene expression data analysis and they concluded that AdaBoost did not perform at a comparable level. One major weakness of AdaBoost is sensitivity to outliers. Because AdaBoost algorithm puts heavy weight on data points which are misclassified in a previous step, outliers tend to have a large weight. As consequence, AdaBoost learns from the outliers therefore the final classifier often cannot show good performance. To prevent this, different loss functions are considered.

The number of iterations is also considered to cause overfitting. Cross validation is used to decide the number of iterations, this is called early stopping. Beside loss function and number of iterations, complexity of the set of weak learners is also an important factor which can cause overfitting. Friedman et al. (2000) pointed out that in the context of Boosting all weak learners are not equivalent, and there is no universal best choice for all situations. Because of the difficulty in finding an universal rule, how to choose the set of weak learners has not been paid enough attention. The set of weak learners is usually decided by the given data set. When the number of features is increased, the number of weak learners is also increased. The complexity of initial set of weak learners become larger.

In this thesis, we propose Sparse Learner Boosting. The key idea is to reduce the initial set of weak learner candidates. 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 weak learners candidates while keeping informative weak learners. The motivation for Sparse Learner Boosting is preventing overfitting of high-dimensional data. Early stopping is considered to prevent overfitting and Rosset et al. (2004) explained the relation of early stopping to regularization in boosting. When the size of dimension is large, the learning process proceeds very quickly in a greedy way. This results in complex model which does not have good performance against unknown data. In this case even early stopping does not prevent this behavior. We will show this using synthesized data and real data.

We also review another important problem in this thesis. As a result of technological progress, the possible number of gene expression measurements in one microarray has been rapidly growing, however many genes are not differentially expressed across subjects to any significant degree. These genes are irrelevant for classification purpose, therefore dimension reduction is applied to find differentially expressed genes. Ranking genes by the degree of different expression among class labels is often performed. Some top ranked genes are considered to be useful for classification. As microarray usage for clinical research has become common, researchers in different laboratories have followed a variety of procedures, some similar some different. In each case, these researchers expected to see similar genes being selected then used for classification, however overlap was small due to the fact that there is no unique gene set for any given classification problem. Some reasons are considered such as different experiment condition or different patient clinical status. Fan et al. (2006) compared five classification models which used different genes. They showed that four out of five classification methods showed similar prediction performance even though almost no common genes existed across those five methods. We considered multiple optimum gene sets existence in a single data set. We used gene sets

which are not top ranked then compared the classification performance using real data. The result showed the existence of multiple optimum gene sets.

The rest of this thesis is organized as follows. In Section 2, we review Boosting methods, related concepts and variants of Boosting methods. In Section 3 we address the details of Sparse Learner Boosting. In Section 4, we discuss the existence of multiple optimum gene sets. Section 5 is conclusions and remarks.

2 Review of Boosting methods

In this chapter we overview the details of Boosting methods and concepts related to Boosting methods then look at the modification of the original Boosting methods. The main idea of Boosting methods is learning from data sequentially using weak learners which are slightly better than random guesses. Two of the key points of Boosting methods are how to choose the weak classifier to be used for the final classifier and how to combine the selected classifiers. Boosting methods were studied from different view points because of their high performance. We review these different view points. Several weaknesses of Boosting methods and modifications to overcome these flaws have been pointed out. We address some of these weakness and modification in this section.

First we explain the notation. Then we present a brief history of Boosting methods. In section 2.3, we review Bagging which is similar to Boosting methods. In section 2.4, loss functions are mentioned. Then we present the details of AdaBoost and related concepts. Lastly the modifications of the Boosting methods are shown.

2.1 Notation

Before going into detail, we set a notation for the classification procedure. We denote input variables by the symbol X. Output is denoted by Y. We use uppercase letters such as X, Y when referring to the generic aspect of a variable. Let (X, Y) be a pair of random variables taking values in $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is a feature space in p-dimension and $\mathcal{Y} = \{-1, +1\}$ is a label set. Observed values are written in lowercase letter. For instance ith value of X is $\boldsymbol{x}_i$. For a given training data set, $\mathcal{L} = \{(\boldsymbol{x}_i, y_i) : i = (1, \ldots, n)\}$, consisting of n independent, identically distributed pairs having the same distribution as (X, Y). Let $F(\boldsymbol{x})$ be a discriminant function associates with a classification rule $H(\boldsymbol{x})$ which is a function of $\boldsymbol{x}$ into y.

2.2 Brief history of Boosting

The main idea of Boosting is to combine simple "weak learners" to build a strong learner which has better classification performance than a single learning, this is called ensemble learning. Several researchers reported significant improvements in performance using ensemble learning (Dietterich (2000), Breiman (1996), Breiman (1998), Dietterich and Bakiri (1995)).

Schapire (1990) was the first to provide a polynomial time Boosting algorithm. Drucker et al. (1993) applied the Boosting idea to an OCR task using neural networks as weak learners. Breiman (1996, 1998) proposed a similar method named bagging and arcing. Schapire and Singer (1999) showed theoretical support for their algorithms in the form of upper bound on generalization error. This theory was developed in the computational learning community based on the concept of PAC learning. Kearns and Valiant (1989) proved that weak learners, each of which perform slightly better than a guess, can be combined to form a powerful ensemble learning.

AdaBoost (Adaptive Boosting) which is proposed by Freund and Schapire (1997) is the most popular Boosting method. Boosting became common because of high performance. Some researchers stated Boosting did not overfit but some researchers reported Boosting overfit eventually. Boosting is learning from given training data in a greedy way. The algorithm is trying to minimize training error. Therefore if data is very noisy, Boosting learns the noise excessively then causes overfitting. We overview several Boosting methods and their main concepts.

Next we give a brief overview of bagging which has a similar concept to Boosting methods to show why Boosting methods work well.

2.3 Bagging

Breiman (1996, 1998) found that gains in accuracy could be obtained by aggregating predictors built from perturbed version of the training data. Set the training data $\mathcal{L} = \{(\boldsymbol{x}_i, y_i) : i = (1, \ldots, n)\}$. A bootstrap sample is generated from given training data with replacement. Bootstrap sample is generated as $\mathcal{L}^1, \ldots, \mathcal{L}^B$ and a classifier C^b is built from each bootstrap sample $\mathcal{L}^b$. Then the final classifier is built from $C^1, \ldots, C^B$ whose output is the class predicted most often by its sub-classifiers. The Bagging algorithm is as follows:

1. For any $b = 1, \ldots, B$

$$\mathcal{L}^b = \text{bootstrap sample from } \mathcal{L} \tag{1}$$

$$C^b(\boldsymbol{x}) = \varphi(\boldsymbol{x}; \mathcal{L}^b), \tag{2}$$

where φ is a classifier which returns a predicted label.

2. Final output is

$$C^*(\boldsymbol{x}) = \arg\max_{y=\pm 1} \sum_{b=1}^{B} I(C^b(\boldsymbol{x}) = y), \tag{3}$$

where $I(\cdot)$ denotes the indicator function, equaling 1 if the condition in parentheses is true, and otherwise 0.

Bagging updates data sequentially by resampling and then decides the predicted labels by majority votes. On the other Boosting methods use loss function during sequential learning. In the next subsection, we survey a variety of loss functions.

2.4 Loss function

Loss function is a function to measure a penalty incurred by the built model if making a misclassification. There are a variety of loss functions. The common loss functions are denoted by

$$L(y, F(\boldsymbol{x})) = \begin{cases} (y - F(\boldsymbol{x}))^2 & \text{Squared loss} \\ \exp(-yF(\boldsymbol{x})) & \text{Exponential loss} \\ \log(1 + \exp(-yF(\boldsymbol{x})) & \text{Loglikelihood.} \end{cases} \tag{4}$$

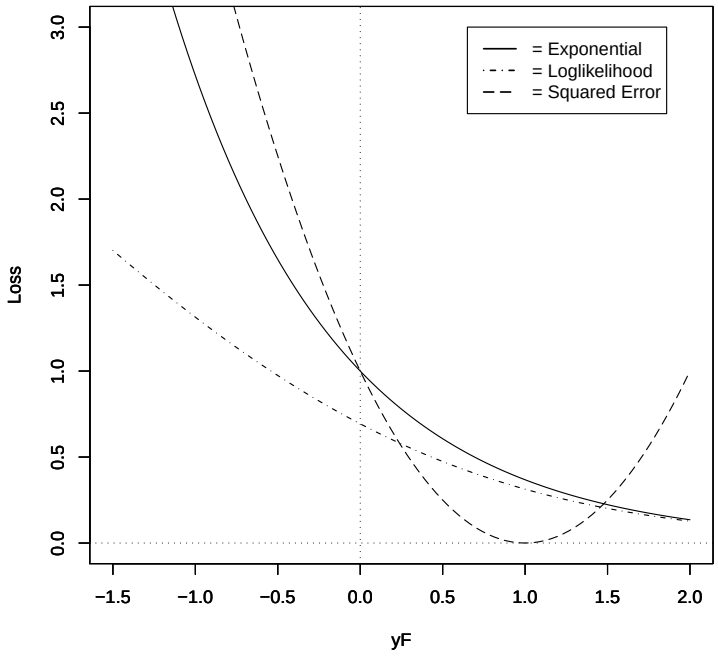

Figure 1: Loss functions. The x-axis is yf and y-axis is loss. Exponential loss, log-likelihood loss and squared loss are drawn.

Figure 1 shows these loss functions as a function of yF. The difference between these loss functions is in the degree of negative penalty. Squared loss is the most

sensitive for the misclassification so it gives a large penalty. The log-likelihood returns a smaller penalty to the misclassified data points than exponential loss. From the view point of optimization, the convex loss function is usually used because of its uniqueness of optimal point. AdaBoost uses the exponential loss function which gives a preferable nature to AdaBoost. We look at the details of AdaBoost algorithm in the next subsection.

2.5 AdaBoost

AdaBoost (Adaptive Boosting) introduced by Freund and Schapire (1997) is the most common Boosting method. AdaBoost builds“ strong ”classifier by linear combination of “ weak ” learners. Bagging learns from resampling data, on the other hand AdaBoost learns from reweighted data. The weight is decided by previous classification result. The data points which are misclassified have a large weight so the AdaBoost algorithm focuses on learning from the misclassified data points. The training data is sequentially reweighted meanwhile the coefficient is calculated at each step.

2.5.1 Algorithm

We use a set of weak learners defined by

$$\mathcal{F} = \{f_j(\boldsymbol{x}) : j \in \{1, \ldots, p\}\}. \tag{5}$$

Each weak learner is generated by each feature. The detail of $f_j(\boldsymbol{x})$ is defined later. After T times iterations, the final classifier is constructed which is linear combination

of coefficient α and weak learners defined by

$$F(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t f_t(\boldsymbol{x}). \tag{6}$$

The coefficient α_t and classifiers f_t are defined by the following discussion. AdaBoost algorithm is characterized by the minimization of the exponential loss function, which is defined by

$$L_{\exp}(F) = \sum_{i=1}^{n} \exp(-y_i F(\boldsymbol{x}_i)). \tag{7}$$

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

$$\begin{aligned} L_{\exp}(F + \alpha f) &= \sum_{i=1}^{n} \exp(-y_i(F(\boldsymbol{x}_i) + \alpha f(\boldsymbol{x}_i))) & (8) \\ &= \varepsilon(f)e^{\alpha} + (1 - \varepsilon(f))e^{-\alpha}, & (9) \end{aligned}$$

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

$$\varepsilon(f) = \sum_{i=1}^{n} I(y_i \neq f(\boldsymbol{x}_i)) \exp(-y_i F(\boldsymbol{x}_i)). \tag{10}$$

The coefficient α is calculated by

$$\arg\min_{\alpha \in \mathbb{R}} L_{\exp}(F + \alpha f) = \frac{1}{2} \log \frac{1 - \varepsilon(f)}{\varepsilon(f)}. \tag{11}$$

The derivation of Equation (11) is written in the Appendix. The optimal value of f on step t is determined by minimizing the weighted error $\varepsilon(f)$. The weight on training example i on step t is denoted by $w_t(i)$. Initially, all weights are set equally, then the weights of incorrectly classified examples are increased on each step so that

the weak learners are forced to focus on the difficult examples in the training set. The details of algorithm is as follows:

1. Set $w_1(i) = 1/n$ and $F_o = 0$

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

a. Find

$$f_t = \arg\min_{f \in \mathcal{F}} \ \varepsilon_t(f) \tag{12}$$

where

$$\varepsilon_t(f) = \sum_{i=1}^{n} w_t(i) I(y_i \neq f(\boldsymbol{x}_i)). \tag{13}$$

b. Calculate

$$\alpha_t = \frac{1}{2} \log \frac{1 - \varepsilon_t(f_t)}{\varepsilon_t(f_t)}. \tag{14}$$

c. Update

$$w_{t+1}(i) = \frac{\exp(-\alpha_t y_i f_t(\boldsymbol{x}_i))}{Z_t}, \tag{15}$$

where $Z_t = \sum_{i=1}^{n} \exp(-\alpha_t y_i f_t(\boldsymbol{x}_i))$.

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

Like Bagging, the AdaBoost algorithm generates a set of classifier and votes them. Beyond this, the main difference between two algorithms is AdaBoost generates the classifier sequentially. On the other hand Bagging can generate them in parallel. On top of the difference, AdaBoost changes the weights of the training samples based

on the previous training result. AdaBoost algorithm increases the weights of examples misclassified by f_t and decreases the weights of correctly classified examples. Furthermore AdaBoost algorithm updates the weight of previously selected f_t to the worst weight. It can be shown that

$$\begin{aligned}
\varepsilon_{t+1}(f_t) &= \sum_{i=1}^{n} I(y_i \neq f_t(\boldsymbol{x}_i)) w_{t+1}(i) && (16)\\
&= \frac{\sum_{i=1}^{n} I(y_i \neq f_t(\boldsymbol{x}_i)) e^{(-\alpha_t y_i f_t(\boldsymbol{x}_i))} w_t(i)}{\sum_{i=1}^{n} I(y_i \neq f_t(\boldsymbol{x}_i)) e^{(\alpha_t y_i f_t(\boldsymbol{x}_i))} w_t(i) + \sum_{i=1}^{n} I(y_i = f_t(\boldsymbol{x}_i)) e^{(-\alpha_t y_i f_t(\boldsymbol{x}_i))} w_t(i)} \\
&= \frac{1}{2}. && (17)
\end{aligned}$$

We illustrate a typical example in Figure 2. The data set $\{(\boldsymbol{x}_i, y_i) : i = i, \ldots, n\}$ was generated as follows. The data $\boldsymbol{x} = (x_1, x_2)$ was sampled from the uniform distribution $U(-1.2, 1.2)$. The class labels y are defined by

$$y = \begin{cases} +1 & \text{if} \quad \sin(3x_1) \leq x_2 \\ -1 & \text{otherwise.} \end{cases} \tag{18}$$

The left panel in Figure 2 shows the Bayes rule decision boundary. The right panel shows AdaBoost boundary after 100 iterations. We can observe that the AdaBoost boundary is close to the Bayes boundary.

2.5.2 AdaBoost as additive logistic regression model

AdaBoost can be interpreted as a stagewise estimation procedure for fitting an additive logistic regression model. We write the details of logistic regression model in Appendix B. Consider the expectation of the exponential loss function,

$$J(F) = E(e^{-yF(\boldsymbol{x})}), \tag{19}$$

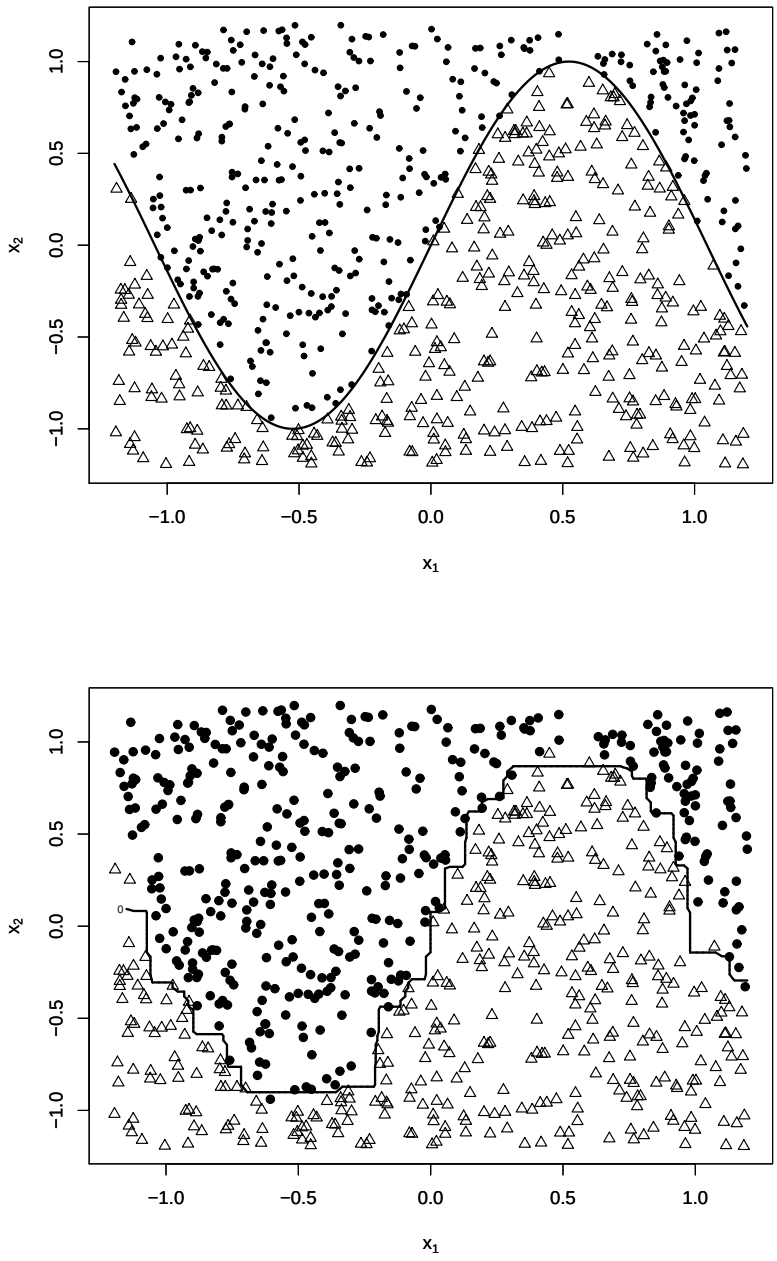

Figure 2: Typical example: Circles and triangles denote data points with class labels $+1$ and -1. In the top panel, solid line shows the Bayes rule boundary. In the bottom panel, the solid line shows the AdaBoost boundary after 100 iterations.

where E represents expectation. This can be rewritten as

$$E(e^{-yF(\boldsymbol{x})}|\boldsymbol{x}) = P(y=1|\boldsymbol{x})e^{-F(\boldsymbol{x})} + P(y=-1|\boldsymbol{x})e^{F(\boldsymbol{x})}. \tag{20}$$

Minimizing the exponential loss function is equivalent to

$$\frac{\partial E(e^{-yF(\boldsymbol{x})}|\boldsymbol{x})}{\partial F(\boldsymbol{x})} = -P(y=1|\boldsymbol{x})e^{-F(\boldsymbol{x})} + P(y=-1|\boldsymbol{x})e^{F(\boldsymbol{x})}. \tag{21}$$

Set the derivative to zero,

$$\begin{aligned}
-P(y=1|\boldsymbol{x})e^{-F(\boldsymbol{x})} + P(y=-1|\boldsymbol{x})e^{F(\boldsymbol{x})} &= 0 \\
-P(y=1|\boldsymbol{x})e^{-F(\boldsymbol{x})} + \{1 - P(y=1|\boldsymbol{x})\}e^{F(\boldsymbol{x})} &= 0 \\
-P(y=1|\boldsymbol{x})(e^{-F(\boldsymbol{x})} + e^{F(\boldsymbol{x})}) &= -e^{F(\boldsymbol{x})} \\
P(y=1|\boldsymbol{x}) &= \frac{e^{F(\boldsymbol{x})}}{e^{-F(\boldsymbol{x})} + e^{F(\boldsymbol{x})}}.
\end{aligned} \tag{22}$$

In the same way,

$$P(y=-1|\boldsymbol{x}) = \frac{e^{-F(\boldsymbol{x})}}{e^{-F(\boldsymbol{x})} + e^{F(\boldsymbol{x})}}. \tag{23}$$

This is also

$$F(\boldsymbol{x}) = \frac{1}{2}\log\frac{P(y=+1|\boldsymbol{x})}{P(y=-1|\boldsymbol{x})}. \tag{24}$$

In Equation (24), the left hand term $F(\boldsymbol{x}) = \sum_{t=1}^{T}\alpha_t f_t(\boldsymbol{x})$, therefore AdaBoost can be considered as an additive logistic regression model.

2.5.3 Training error and generalization error

Freund and Schapire (1997) showed the upper bound of training error. The training error $\mathrm{Err}_{\mathrm{tr}}(F(\boldsymbol{x}))$ is

$$\mathrm{Err}_{\mathrm{tr}}(F(\boldsymbol{x})) = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq F(\boldsymbol{x}_i)), \tag{25}$$

which is bounded by

$$\mathrm{Err}_{\mathrm{tr}}(F) \quad \leq \quad 2^T \prod_{t=1}^{T} \sqrt{\varepsilon_t(f_t)(1-\varepsilon_t(f_t))}. \tag{26}$$

Equation (26) is proved as follows. First we show that

$$w_{T+1}(i) = \frac{1}{n} \cdot \frac{\exp(-y_i F(\boldsymbol{x}))}{\prod_{t=1}^{T} Z_t}. \tag{27}$$

It holds that

$$\begin{aligned} w_{t+1}(i) &= w_1(i) \cdot \frac{\exp(-\alpha_1 y_i f_1(\boldsymbol{x}_i))}{Z_1} \cdots \frac{\exp(-\alpha_T y_i f_T(\boldsymbol{x}_i))}{Z_T} && (28)\\ &= \frac{1}{n} \cdot \frac{\exp(-y_i \sum_{t=1}^{T} \alpha_t f_t(\boldsymbol{x}_i))}{\prod_{t=1}^{T} Z_t} && (29)\\ &= \frac{1}{n} \cdot \frac{\exp(-y_i F(\boldsymbol{x}))}{\prod_{t=1}^{T} Z_t}. && (30) \end{aligned}$$

The expected exponential loss function can be formed by

$$\begin{aligned} L(F) &= \frac{1}{n}\sum_{i=1}^{n} \exp(-y_i F(\boldsymbol{x}_i)) && (31)\\ &= \sum_{i=1}^{n} w_{T+1}(i) \prod_{t=1}^{T} Z_t && (32)\\ &= \prod_{t=1}^{T} Z_t. && (33) \end{aligned}$$

Z_t can be rewritten as follows:

$$\begin{aligned} Z_t &= \sum_{i=1}^{n} w_t(i)\exp(-\alpha_t y_i f_t(\boldsymbol{x}_i)) && (34)\\ &= \sum_{i=1}^{n} I(y_i \neq f(\boldsymbol{x}_i))w_t(i)e^{-\alpha} + \sum_{i=1}^{n} I(y_i = f(\boldsymbol{x}_i))w_t(i)e^{\alpha} && (35)\\ &= e^{(-\alpha)}(1-\varepsilon_t(f_t(\boldsymbol{x}))) + e^{(\alpha)}\varepsilon_t(f_t(\boldsymbol{x})) && (36)\\ &= 2\sqrt{\varepsilon_t(f_t(\boldsymbol{x}))(1-\varepsilon_t(f_t(\boldsymbol{x})))}. && (37) \end{aligned}$$

Therefore

$$L(F) = \frac{1}{n}\sum_{i=1}^{n}\exp(-y_i F(\boldsymbol{x}_i)) = 2^T \prod_{t=1}^{T}\sqrt{\varepsilon_t(f_t(\boldsymbol{x}))(1-\varepsilon_t(f_t(\boldsymbol{x})))}. \quad (38)$$

Since

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq F(\boldsymbol{x}_i)) \leq \frac{1}{n}\sum_{i=1}^{n}\exp(-y_i F(\boldsymbol{x}_i)), \quad (39)$$

the upper bound of the training error is

$$\mathrm{Err}_{\mathrm{tr}}(F(\boldsymbol{x})) \leq 2^T \prod_{t=1}^{T}\sqrt{\varepsilon_t(f_t(\boldsymbol{x}))(1-\varepsilon_t(f_t(\boldsymbol{x})))}. \quad (40)$$

Then this completes the proof.

When $\varepsilon_t(f_t) \leq 1/2 - \gamma$ ($\gamma > 0$), Equation (40) is

$$\begin{aligned} 2^T \prod_{t=1}^{T}\sqrt{\varepsilon_t(f_t)(1-\varepsilon_t(f_t))} &= \prod_{t=1}^{T}\sqrt{1-4\gamma_t^2} && (41)\\ &\leq \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right). && (42) \end{aligned}$$

Then the training error of the combined classifiers decreases exponentially. Freund and Schapire (1997) study the error outside the training data, which is generalization error. Equation (40) shows that the training error is small; however the interest of classification is in how well the final classifier works against unknown data. The generalization error is the probability of misclassifying a new example, while the test error is the fraction of mistakes on a sample test set, thus generalization error is expected test error. Freund and Schapire (1997) showed how to bound the generalization error of the final classifier in terms of its training error as follows:

$$\hat{P}[F(\boldsymbol{x}) \neq y] + \tilde{O}\left(\sqrt{\frac{Td}{n}}\right), \tag{43}$$

where $\hat{P}[\cdot]$ denotes empirical probability on the training sample, n is the size of the sample, d is VC dimension of the base classifier space and T is the number of iterations. VC dimension is a measure of model complexity. We will mention the details of VC dimension later. From the bound of training error suggests that Boosting will overfit if run forever. However some authors observed empirically that Boosting does not cause overfitting, and furthermore, the test error is driven down long after the training error reaches zero.

In response to these empirical findings, Schapire et al. (1998) and Bartlett (1998) gave an alternative analysis in terms of the margins of the training data. Schapire et al. (1998) defined the margin for Boosting methods as follows:

$$M_{Boost}(\boldsymbol{x}, y) = \frac{yF(\boldsymbol{x})}{||\boldsymbol{\alpha}||_1} = \frac{y\sum_{t=1}^{T}\alpha_t f_t(\boldsymbol{x})}{||\boldsymbol{\alpha}||_1}, \tag{44}$$

where $||\boldsymbol{\alpha}||_1$ is l_1 norm for $\boldsymbol{\alpha}$ vector which is $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_T)$. Since the denominator is normalized by the sum of the coefficients α_t, the range of the margins is $[-1, +1]$. The margin is positive if and only if $F(\boldsymbol{x})$ correctly classifies the example thus the

margin can be read with confidence. They observed that more iterations give a closer value to +1 using separable examples. Schapire et al. (1998) also proved that larger margins on the training set translates into a superior upper bound on the generalization error which is at most

$$\hat{P}\left[M_{Boost}(\boldsymbol{x}, y) \leq \theta\right] + \tilde{O}\left(\sqrt{\frac{d}{n\theta^2}}\right), \tag{45}$$

for any $\theta > 0$ with high probability. Schapire et al. (1998) showed the empirical result in his paper. We synthesized a toy example to observe the progress of the margin in accordance with the number of iterations. The plot of the toy example is in Figure 3. Black circles and white circles denote data points with class labels +1 and −1 respectively. The data points with class label +1 follow the normal distribution $N((0,0), 1.5\boldsymbol{I})$. The data points with class labels −1 follow the normal distribution $N((4,4), 1.5\boldsymbol{I})$.

Figure 4 shows the progress of training error and test error. Figure 5 presents the cumulative distribution of margins after 500 and 1000 iterations. Training error reached zero at a very early stage, however the margin becomes larger at 1000 iterations than at 500 iterations.

However the bound is rather weak furthermore when a data sample contains noise, the margin does not approach 1. Grove and Schuurmans (1998) showed an example in which generalization error was not decreased even when the margin goes to 1. The margin theory however is used to explain the connection between AdaBoost and Support Vector Machine. We will review the relation to Support Vector Machine in the next subsection.

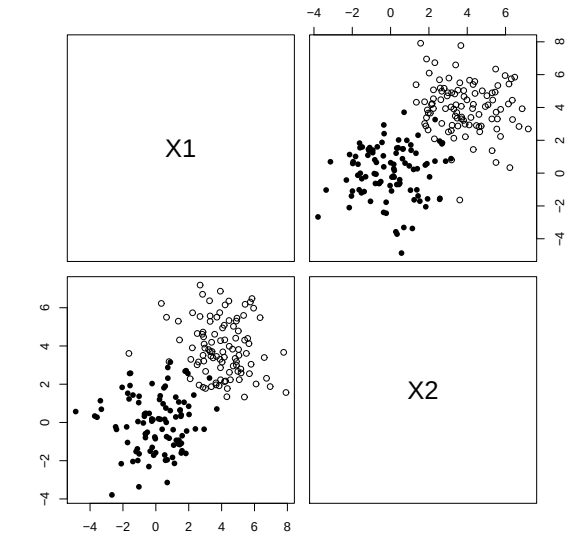

Figure 3: Toy example. Black circles and white circles denote data points with class labels $+1$ and -1 respectively.

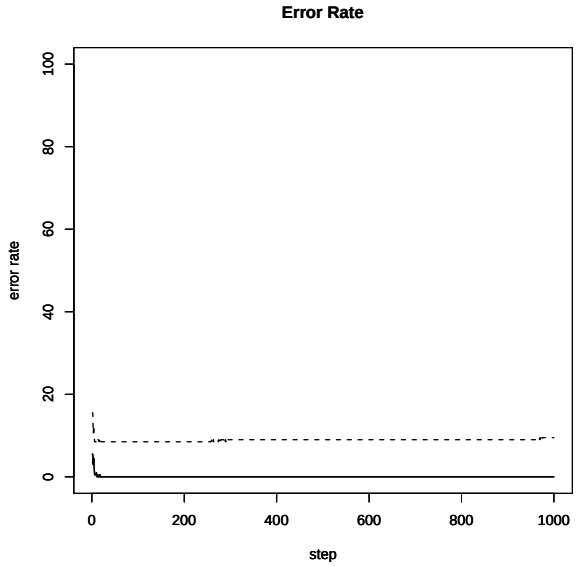

Figure 4: Progress of training error and test error. The x-axis is the number of iteration and the y-axis is error. The solid line and dashed line show the progress of training error and test error respectively.

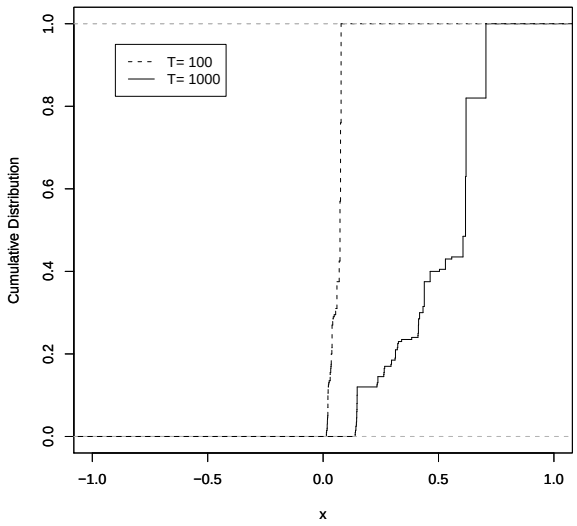

Figure 5: The cumulative distribution of margins of the training data after 100 and 1000 times iterations, indicated by dashed line and solid line respectively.

2.5.4 Relation to Support Vector Machine

Support Vector Machine (SVM) is a well known machine learning method as is Boosting method. SVM is introduced by Vapnik (1998). The details of SVM is written in Appendix. Shapire addressed the relationship between AdaBoost and SVM using the margin. The basic concept of SVM is called 'maximum minimum margin'. A data point is viewed as a p-dimensional vector. SVM aims to find a hyperplane which can separate such points. Shapire defined the SVM maxium minimum margin which is

$$m_{\mathrm{SVM}} = \min_i \frac{y_i F(\boldsymbol{x}_i)}{||\boldsymbol{\beta}||_2}, \tag{46}$$

where $||\boldsymbol{\beta}||_2$ is the l_2 or Euclidean norm of $\boldsymbol{\beta}$ vector which is $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$. This is similar to the Boosting margin Equation (44). Viewed this way, the connection

between SVM and Boosting method becomes clear. Both methods aim to find a linear combination in a high dimensional space which has a large margin. When described in this manner, SVM and AdaBoost seems to be similar however there are several important differences which Shapire pointed out.

The first one is that different norms can result in very different margins. SVM uses Euclidean norm and Boosting uses l_1 norm as the denominator. This difference may not be very significant when one considers low dimensional spaces however in the case of high dimension size, the difference between the norms can result in very large differences in the margin values. The second difference is computational requirements. The computation involved in maximizing the margin is mathematical programming. The difference between the two methods in this regard is that SVM corresponds to quadratic programming, while Boosting method corresponds only to linear programming. The third difference is that different approaches are used to search efficiently in high dimensional space. SVM finds the maximum margin in a high dimensional spaces using kernel trick which is a method to solve a non-linear problem by mapping the original observations into a higher-dimensional space. This makes a linear classification in the higher-dimension equivalent to non-linear classification in the original space. The boosting approach is instead to employ greedy search which adds a weak classifier one by one with re-weighted original data. As kernel trick and greedy search are very different approaches, the resulting learning algorithms can be very different.

Figure 6 upper panel shows a simple example of separable data in two dimensions, with its margin-maximizing separating hyper-plane. The lower panel shows a Boosting margin maximizing separating hyper-plane for the same simple example as the SVM margin.

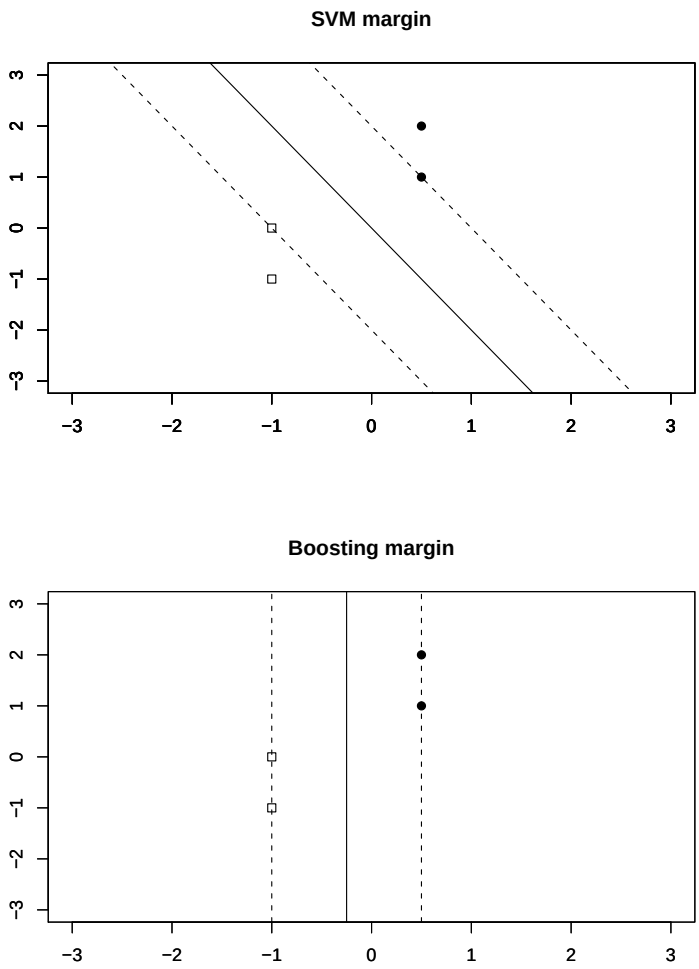

Figure 6: A simple example of two data points in each class. The upper panel shows the SVM margin and the lower panel shows the Boosting margin. The solid line is the boundary.

Rosset et al. (2004) added further interpretation to the Boosting margin relation to the SVM margin. Equation (44) defined the Boosting margin using α_t coordinates which are calculated during Boosting iterations. Rosset et al. (2004) represented the Boosting margin using β and $f(\boldsymbol{x}) = \boldsymbol{x}$. Boosting discriminant function can be rewritten

$$F(\boldsymbol{x}) = \sum_{j(t)=j}^{p} \beta_{j(t)} f_j(\boldsymbol{x}), \tag{47}$$

where $\beta_{j(t)} = \sum_{j(t)=j} \alpha_t$. Using this notation, minimum of the Boosting margin can be written

$$m_{\text{Boost}} = \min_i \frac{y_i F(\boldsymbol{x}_i)}{||\boldsymbol{\beta}||_1}, \tag{48}$$

where $||\boldsymbol{\beta}||_1$ is the l_1 norm of $\boldsymbol{\beta}$ vector which has p dimension. From this representation, Rosset et al. (2004) showed the relation between m_{SVM} and m_{Boost} as follows:

$$\frac{yF(\boldsymbol{x})}{||\boldsymbol{\beta}||_1} = \frac{yF(\boldsymbol{x})}{||\boldsymbol{\beta}||_2} \cdot \frac{||\boldsymbol{\beta}||_2}{||\boldsymbol{\beta}||_1}. \tag{49}$$

From this representation, we can observe that the Boosting margin will tend to be large if the ratio $||\boldsymbol{\beta}||_2/||\boldsymbol{\beta}||_1$ is large. To see this, consider fixing the l_1 norm then comparing the l_2 norm of two candidates, one with many small components and another with a few large components and many zero components. For example $\boldsymbol{\beta}_{\text{non-sparse}} = (1, 2, 1, 1, 2, 2, 1)$ and another $\boldsymbol{\beta}_{\text{sparse}} = (3, 3, 3, 0, 0, 0, 1)$. Both l_1 norm are the same, 10. Then calculate l_2 norm for $\boldsymbol{\beta}_{\text{non-sparse}}$ and $\boldsymbol{\beta}_{\text{sparse}}$, which are 16 and 28 respectively. The sparse $\boldsymbol{\beta}$ vector has a larger l_2 norm, hence a larger Boosting margin.

2.5.5 Number of iteration and step size to avoid overfitting

As we described in the previous section, AdaBoost decreases training error exponentially but it has a tendency to cause overfitting. Several algorithms are proposed to handle this situation. Grove and Schuurmans (1998) and Jiang (2004) showed that running Boosting forever causes overfitting. Zhang and Yu (2005) proposed that stopping the learning process early prevents overfitting. This is called early stopping. Zhang and Yu (2005) studied the numerical convergence, consistency, and statistical rates of convergence of boosting with early stopping. Using the numerical convergence they concluded that the early stopping strategy is shown to be consistent based on iid samples. Besides the number of iterations, step size can cause overfitting as well. Friedman et al. (2000) proposed to fix the step size which is denoted α in Equation (14). The fixed step size Boosting is called ε-boosting. Friedman et al. (2000) mentioned that fixing the step size to a very small value makes the learning speed slower thus preventing overfitting. This eliminates a favorable statistical property however their empirical results showed a large performance improvement in regression case. They noted that the change was less significant in zero-one loss.

2.5.6 Relation to Lasso

Rosset et al. (2004) introduced another Boosting methods interpretation. Lasso is tracking a path of approximate solutions to loss function with l_1 constrained which is Lasso. Given a convex non-negative loss functions $L(\cdot, \cdot)$ such as exponential loss or logit loss, consider the 1-dimensional path of optimal solutions to l_1 constrained optimization problems over the training data,

$$\hat{\beta}(c) = \underset{||\beta||_1 \leq c}{\arg\min} \sum_{i=1}^{n} L(y_i, \boldsymbol{\beta} f(\boldsymbol{x}_i)). \tag{50}$$

As c varies, $\hat{\beta}(c)$ traces a 1-dimensional "optimal curve" through $\mathbb{R}^p$. If an optimal solution for the non-constrained problem exists and has finite l_1 norm c_0, then $\hat{\beta}(c) = \hat{\beta}(c_0) = \hat{\beta}, \forall c > c_0$. In the case of separable binary class data, there is no finite-norm optimal solution for either exponential loss or logit loss. Then the constrained solutions will always have $||\hat{\beta}(c)||_1 = c$.

A different way of building a solution which has l_1 norm c, is to run the ε-boosting algorithm for c/ε iterations. This will give an $\alpha^{c/\varepsilon}$ vector which has l_1 norm exactly c. For the norm of the geometric representation $\beta^{c/\varepsilon}$ to also be equal to c. Rosset et al. (2004) showed the similarity using real data in their paper.

From the next section, we review the modification of Boosting using different loss functions.

2.6 Boosting variants by different loss function

In the previous section, we addressed the details of AdaBoost. AdaBoost uses the exponential loss function as objective function. As we reviewed in an early section, there are a variety of loss functions. In this section, we present variants of Boosting methods which use different loss functions. We review η-Boost, LogitBoost and SparseL_2Boost.

2.6.1 η-Boost

η-Boost is proposed by Takenouchi and Eguchi (2004) 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

$$L_\eta(F) = \sum_{i=1}^{n} [(1-\eta)\exp(-y_i F(\boldsymbol{x}_i)) - \eta y_i F(\boldsymbol{x}_i)]. \tag{51}$$

where $0 \leq \eta \leq 1$. We can derive η-Boost by the sequential minimization of the loss function, Equation (51). See more details Eguchi and Copas (2001). 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

$$f_t = \arg\min_{f \in \mathcal{F}} \ \varepsilon^*(f) \tag{52}$$

where

$$\varepsilon_t^*(f_t) = \sum_{i=1}^{n} w_t^*(i) I(y_i \neq f_t(\boldsymbol{x}_i)). \tag{53}$$

b. Calculate

$$\alpha_t^* = \log \frac{\sqrt{1 - \varepsilon_t(f_t) + (\eta K_t)^2} + \eta K_t}{\sqrt{\varepsilon_t(f_t)}} \tag{54}$$

where

$$K_t = \frac{(1 - 2\varepsilon_1(f_t))}{2\sqrt{\varepsilon_t(f_t)}} \left(\frac{(1-\eta) Z_t}{N} \right)^{-1} \tag{55}$$

with

$$Z_{t+1} = \sum_{i=1}^{n} \exp(-y_i F_t(\boldsymbol{x}_i)). \tag{56}$$

c. Update

$$w_{t+1}^*(i) = \frac{(1-\eta)\exp(-y_i F_t(\boldsymbol{x}_i)) + \eta}{Z_{t+1}^*} \tag{57}$$

where

$$F_t(\boldsymbol{x}) = \sum_{m=1}^{t} \alpha_m^* f_m(\boldsymbol{x}) \tag{58}$$

$$Z_{t+1}^* = \sum_{i=1}^{n} (1-\eta)\exp(-y_i F_t(\boldsymbol{x}_i)) + \eta. \tag{59}$$

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

The derivations for Equations (53) and (54) are written in the Appendix. Fig (7) shows an example which data has 10 observations were contaminated especially near the boundary. This example addresses that AdaBoost is very sensitive to noise. From the complicated boundary in AdaBoost shows AdaBoost learns from the noise.

2.6.2 Properties of mislabel model

We look at further details of mislabel model. We show some properties of mislabel model which is introduced by Takenouchi et al. (2008). They expand mislabel model to multiclass model. Set $\mathcal{G} = \{1, \ldots, K\}$ is finite multiclass labels. Conditional probability of $G = g, X = \boldsymbol{x}$ say $p(g|\boldsymbol{x})$. Consider mislabel model $p_\zeta(g|\boldsymbol{x})$ defined by

$$p_\zeta(g|\boldsymbol{x}) = \frac{1}{M(\boldsymbol{x})}\left[\left\{1 - \sum_{k\neq g}\zeta_k(\boldsymbol{x})\right\} p(g|\boldsymbol{x}) + \sum_{k\neq g}\zeta_k(\boldsymbol{x}) p(k|\boldsymbol{x})\right], \tag{60}$$

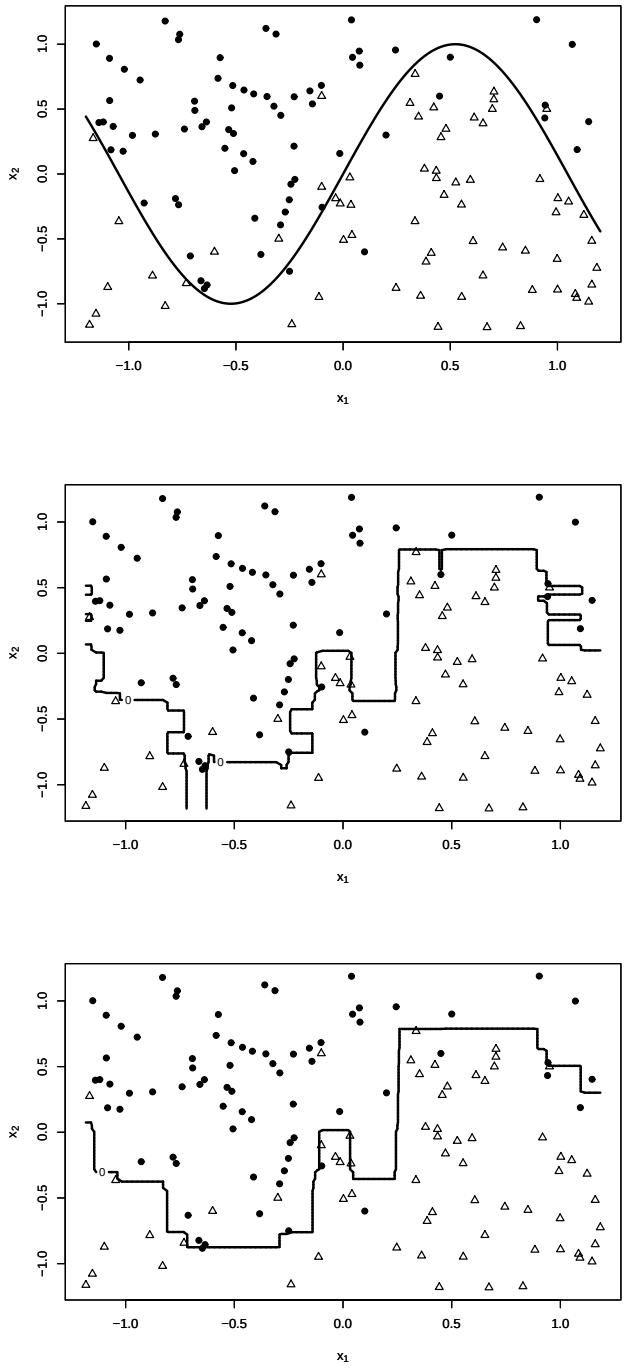

Figure 7: Typical example: Circles and triangles denote data points with class labels $+1$ and -1. In the top panel, solid line shows the Bayes rule boundary. In the center panel, the solid line is AdaBoost boundary. In the bottom panel, the solid line shows the η-Boost boundary after 100 iterations.

where $M(\boldsymbol{x})$ is a normalization constant and $\zeta(\boldsymbol{x})$ is an element which is subset of $\mathcal{D}$, both are defined by

$$M(\boldsymbol{x}) = \sum_{k=1}^{g} \left[\left\{ 1 - \sum_{k \neq g} \zeta_k(\boldsymbol{x}) \right\} p(g|\boldsymbol{x}) + \sum_{k \neq g} \zeta_k(\boldsymbol{x}) p(k|\boldsymbol{x}) \right], \tag{61}$$

$$\mathcal{D} = \{\zeta(\boldsymbol{x}) = (\zeta_1(\boldsymbol{x}), \ldots, \zeta_g(\boldsymbol{x})); 0 \leq \sum_{k \in \mathcal{G}} \zeta_k(\boldsymbol{x}), \zeta_k(\boldsymbol{x}) \geq 0\}. \tag{62}$$

This shows that $\zeta_k(\boldsymbol{x})$ is the probability that data is wrongly labeled to class k. The mislabel model which is defined by Equation (60) has the below properties.

Property 1. Posterior probability of $p_\zeta(\boldsymbol{x})$ obtains the consistency of Bayes rule.

$M(\boldsymbol{x})$ is positive because of the definition thus we only consider inside the bracket then add and subtract $\zeta_k(\boldsymbol{x})$,

$$\left\{ 1 - \sum_{k \neq g} \zeta_k(\boldsymbol{x}) \right\} p(g|\boldsymbol{x}) + \sum_{k \neq g} \zeta_k(\boldsymbol{x}) p(k|\boldsymbol{x}) \tag{63}$$

$$= \left\{ 1 - \sum_{k \neq g} \zeta_k(\boldsymbol{x}) \right\} p(g|\boldsymbol{x}) - \zeta_k(\boldsymbol{x}) + \sum_{k \neq g} \zeta_k(\boldsymbol{x}) p(k|\boldsymbol{x}) + \zeta_k(\boldsymbol{x}) \tag{64}$$

$$= \left\{ 1 - \sum_{k \in \mathcal{G}} \zeta_k(\boldsymbol{x}) \right\} p(g|\boldsymbol{x}) + \sum_{k \in \mathcal{G}} \zeta_k(\boldsymbol{x}) p(k|\boldsymbol{x}) \tag{65}$$

$$= \{1 - K\zeta(\boldsymbol{x})\} p(g|\boldsymbol{x}) + \zeta(\boldsymbol{x}), \tag{66}$$

which implies that the Bayes rule based on the original posterior $p(g|\boldsymbol{x})$ is the same as that based on the current posterior $p_\zeta(\boldsymbol{x})$ for any $\zeta(\boldsymbol{x}) \in \mathcal{D}$. From this result, for any $\zeta \in \mathcal{D}$, $F_p^*(\boldsymbol{x}) = F_{p_\zeta}(\boldsymbol{x})$.

This can be shown as follows. For any $g, k \in \mathcal{G}$ and $\boldsymbol{x} \in \mathcal{X}$, we observe

$$p_\zeta(g|\boldsymbol{x}) - p_\zeta(k|\boldsymbol{x}) = \{1 - G\zeta(\boldsymbol{x})\} \{p(g|\boldsymbol{x}) - p(k|\boldsymbol{x})\} \tag{67}$$

Here $1 - K\zeta(\boldsymbol{x})$ is positive because $\zeta(\boldsymbol{x}) > 0$, $\zeta(\boldsymbol{x}) \in \mathcal{D}$ and g are invariant.

Property 2. Mislabel model probability $p_\zeta(\boldsymbol{x})$ follows uniform distribution when mislabel probability $\zeta_k(\boldsymbol{x})$ follows uniform distribution.

The proof is below. First set

$$p_\zeta(g|\boldsymbol{x}) = \frac{1}{M(\boldsymbol{x})}\left[\left\{1 - \sum_{k\neq g}\zeta_k(\boldsymbol{x})\right\}p(g|\boldsymbol{x}) + \sum_{k\neq g}\zeta_k(\boldsymbol{x})p(k|\boldsymbol{x})\right].$$

Set $\zeta_k(\boldsymbol{x}) = 1/K$ then

$$\begin{aligned}
&= \frac{1}{M(\boldsymbol{x})}\left[\left\{1 - \sum_{k\neq g}\frac{1}{K}\right\}p(g|\boldsymbol{x}) + \sum_{k\neq g}\frac{1}{g}p(k|\boldsymbol{x})\right] \\
&= \frac{1}{M(\boldsymbol{x})}\left\{\frac{1}{K}\sum_{k\in\mathcal{G}}p(k|\boldsymbol{x})\right\} \qquad (68)\\
&= \frac{1}{M(\boldsymbol{x})}\frac{1}{K}. \qquad (69)\\
& \qquad (70)
\end{aligned}$$

From the definition, $M = 1$ then we could prove that $p_\zeta(g|\boldsymbol{x}) = 1/K$.
In the next property, we look at the details of error bound. First the error rate is defined by

$$\begin{aligned}
\mathrm{err}(F, p, q) &= \mathrm{Prob}_{p,q}(F(\boldsymbol{X}) \neq G) \qquad (71)\\
&= 1 - \sum_{g\in\mathcal{G}}\int_{\mathcal{X}_g^F} p(g|\boldsymbol{x})q(\boldsymbol{x})d\boldsymbol{x}, \qquad (72)
\end{aligned}$$

where $\mathcal{X}_g^F = \{\boldsymbol{x}|\boldsymbol{x} \in \mathcal{X}, F(\boldsymbol{x}) = g\}$.

Then consider a lower bound errbd(p,q) of the error rate under $p(g|\boldsymbol{x})q(\boldsymbol{x})$

$$\text{errbd}(p, q) = \text{err}(F_p^*, p, q), \tag{73}$$

where F_p^* is Bayes rule. Since Bayes rule is the optimal rule, it is justified to set Bayes rule as the lower bound.

Property 3. For any $\zeta \in \mathcal{D}$, the relation of the error bound $\text{errbd}(p_\zeta, q)$ and $\text{errbd}(p, q)$ is

$$\text{errbd}(p_\zeta, q) \geq \text{errbd}(p, q). \tag{74}$$

From the property 2, we know that a region $\mathcal{X}_{F_p^*,g}$ coincides with $\mathcal{X}_{F_{p_\zeta},g}$. Then

$$\begin{aligned} \text{errbd}(p, q) &= \text{err}(F_p^*, p, q) \\ &= 1 - \sum_{g \in \mathcal{G}} \int_{\mathcal{X}_{F_p^*,g}} p(g|\boldsymbol{x})q(\boldsymbol{x})d\boldsymbol{x} \\ \text{errbd}(p_\zeta, q) &= \text{err}(F_{p_\zeta}, p, q) \\ &= 1 - \sum_{g \in \mathcal{G}} \int_{\mathcal{X}_{F_p^*,g}} p_\zeta(g|\boldsymbol{x})q(\boldsymbol{x})d\boldsymbol{x}. \end{aligned} \tag{75}$$

Consider

$$
\begin{aligned}
&\mathrm{errbd}(p,q) - \mathrm{errbd}(p_\zeta,q) \\
&= \left\{1 - \sum_{g\in\mathcal{G}} \int_{\mathcal{X}_{F_p^*,g}} p(g|\boldsymbol{x})q(\boldsymbol{x})d\boldsymbol{x}\right\} - \left\{1 - \sum_{g\in\mathcal{G}} \int_{\mathcal{X}_{F_p^*,g}} p_\zeta(g|\boldsymbol{x})q(\boldsymbol{x})d\boldsymbol{x}\right\} \quad (76)\\
&= \sum_{g\in\mathcal{G}} \int_{\mathcal{X}_{F_p^*,g}} p(g|\boldsymbol{x})q(\boldsymbol{x})d\boldsymbol{x} - \sum_{g\in\mathcal{G}} \int_{\mathcal{X}_{F_p^*,g}} p_\zeta(g|\boldsymbol{x})q(\boldsymbol{x})d\boldsymbol{x} \quad (77)\\
&= \sum_{g\in\mathcal{G}} \int_{\mathcal{X}_{F_p^*,g}} \{p(g|\boldsymbol{x}) - (1 - K\zeta(\boldsymbol{x}))p(g|\boldsymbol{x}) + \zeta(\boldsymbol{x})\}\, q(\boldsymbol{x})d\boldsymbol{x} \quad (78)\\
&= \sum_{g\in\mathcal{G}} \int_{\mathcal{X}_{F_p^*,g}} \{K\zeta(\boldsymbol{x})p(g|\boldsymbol{x}) + \zeta(\boldsymbol{x})\}\, q(\boldsymbol{x})d\boldsymbol{x} \quad (79)\\
&= \sum_{g\in\mathcal{G}} \int_{\mathcal{X}_{F_p^*,g}} \{\{Kp(g|\boldsymbol{x}) + 1\}\zeta(\boldsymbol{x})\}q(\boldsymbol{x})d\boldsymbol{x} \quad (80)\\
&= \sum_{g\in\mathcal{G}} \int_{\mathcal{X}_{F_p^*,g}} \{K\{p(g|\boldsymbol{x}) + p_U(g|\boldsymbol{x})\}\zeta(\boldsymbol{x})\}q(\boldsymbol{x})d\boldsymbol{x}, \quad (81)
\end{aligned}
$$

where $p_U(g|\boldsymbol{x})$ is an uniform distribution. We observe for any class label $g \in \mathcal{G}$ that

$$
\boldsymbol{x} \in \mathcal{X}_{F_{p,g}^*} \Rightarrow p(g|\boldsymbol{x}) = \max_{k\in\mathcal{G}} p(k|\boldsymbol{x}) \geq p_U(g). \quad (82)
$$

Therefore $\mathrm{errbd}(p,q) - \mathrm{errbd}(p_\zeta,q)$ is not negative, which completes the proof.

2.6.3 LogitBoost

We review one more AdaBoost variant, LogitBoost which is introduced by Friedman (2001). LogitBoost relies on the binominal log-likelihood as a loss function, which is a more natural criterion in classification than the exponential criterion underlying the AdaBoost algorithm. Since LogitBoost increases linearly instead of exponentially for negative margins. Therefore it can be expected that LogitBoost is more robust in noisy problems. LogitBoost algorithm is written as follows:

1. Set $w_1(i) = 1/n$ and $F_0 = 0$ and initial probability estimates $p_0(\boldsymbol{x}_i) = 1/2$.

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

a. Calculate

$$w_t(i) = p_{t-1}(\boldsymbol{x}_i)(1 - p_{t-1}(\boldsymbol{x}_i)), \tag{83}$$

$$z_{t-1}(i) = \frac{y_i - p_{t-1}(\boldsymbol{x}_i)}{w_t(i)}. \tag{84}$$

b. Update

$$F_t(\boldsymbol{x}) = \sum_{t=1}^{T} \frac{1}{2} f_t(\boldsymbol{x}) \tag{85}$$

$$p_t(\boldsymbol{x}_i) = (1 + \exp(-2 \cdot F_t(\boldsymbol{x}_i)))^{-1}. \tag{86}$$

We point out that $F(\boldsymbol{x})$ which LogitBoost algorithm estimates each step is equivalent to estimating of half of the log-odds ratio

$$F(\boldsymbol{x}) = \frac{1}{2} \log \left(\frac{p(\boldsymbol{x})}{1 - p(\boldsymbol{x})} \right). \tag{87}$$

2.6.4 SparseL_2Boost

We review the other Boosting variant aiming for a sparse solution which is named SparseL_2Boost proposed by Bühlmann and Yu (2006). The title of their paper is Sparse Boosting however they call their algorithm SparseL_2Boost in their paper. To avoid confusion with our proposed method and for consistency, we call their method SparseL_2Boost in this thesis. SparseL_2Boost uses the squared error for the loss function and adds a regularization term. Friedman (2001) proposed L_2Boosting which uses the squared loss for Boosting. SparseL_2Boost is based on L_2Boosting.

Their motivation is similar to ours. In the high dimensional data, there are many irrelevant predictors for classification. Therefore a sparse solution is necessary. The main procedure of L_2Boosting simply fits squared error to the current response then uses the residuals from the previous iteration as the new response vector and so on.

We first show L_2Boosting algorithm then mention SparseL_2Boost algorithm. The current response is denoted $U = \{U_1, \ldots, U_n\}$.

1. Set $F_0 = 0$.

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

a. calculate residuals $U_i = Y_i - \hat{F}_{t-1}(X_i)$

b. $\mathcal{S}_t = \underset{1 \le j \le p}{\arg\min} \sum_{i=1}^{n} (U_i - \gamma_j x_{ij})^2$
where $\gamma_j = \sum_{i=1}^{n} (U_i x_{ij} / x_{ij}^2)$
$f_t(\boldsymbol{x}) = \gamma_{\mathcal{S}_t} \boldsymbol{x}$
$F_t(\boldsymbol{x}) = F_{t-1}(\boldsymbol{x}) + \nu f_t(\boldsymbol{x})$
where $0 < \nu \le 1$ is a pre-specified step size parameter.

3. Repeat step 2 until some stopping value for the number of iterations is reached.

L_2Boosting finds the predictor variable which reduces the residual sum of square most when using least square fitting. They show some variants to choose the best predictor variables in their paper. Here we showed the simplest case. Next we see the sparse L_2 Boost.

As L_2Boosting is trying to find the predictor variable which minimizes the residual every iteration, Bühlmann and Yu (2006) considered L_2Boosting is learning in a greedy way. Bühlmann and Yu (2006) proposed to choose the predictor variable in the sense of model selection. They construct a function which measures the

complexity of Boosting. First define a hat-operator as follows:

$$\mathcal{H}_{\mathcal{S}} : U \mapsto (g_{(\boldsymbol{x}_{\mathcal{S}},U)}(\boldsymbol{x}_1), \ldots, g_{(\boldsymbol{x}_{\mathcal{S}},U)}(\boldsymbol{x}_n)), U = (U_1, \ldots, U_n), \tag{88}$$

where $g_{(\boldsymbol{x}_{\mathcal{S}},U)} = \gamma_{\mathcal{S}_t}\boldsymbol{x}$.

$$\mathcal{B}_t = I - (I - \nu\mathcal{H}_{\mathcal{S}_t})(I - \nu\mathcal{H}_{\mathcal{S}_{t-1}}) \cdots (I - \nu\mathcal{H}_{\mathcal{S}_1}). \tag{89}$$

The degree of freedom of Boosting is then defined by

$$\text{trace}(\mathcal{B}_t) = \text{trace}(I - (I - \nu\mathcal{H}_{\mathcal{S}_t})(I - \nu\mathcal{H}_{\mathcal{S}_{t-1}}) \cdots (I - \nu\mathcal{H}_{\mathcal{S}_1})) \tag{90}$$

This is a standard definition for degree of freedom, see Green and Silverman (1994). Therefore the final prediction error criterion proposed by Akaike (1970) can be formed as follows:

$$\sum_{i=1}^{n}(y_i - F_t(\boldsymbol{x}_i))^2 + \tau \cdot \text{trace}(\mathcal{B}_t). \tag{91}$$

SparseL_2Boost uses Equation (91) as the criterion to move iterations from $t-1$ to t. Precisely they set the below objective function so that model complexity can be controlled. The criterion is defined by

$$T(Y, \mathcal{B}) = C(\sum_{i=1}^{n}(Y_i - (\mathcal{B}Y_i))^2, \text{trace}((B))) \tag{92}$$

where τ is a parameter for some $\tau > 0$. Bühlmann and Yu (2006) mentioned that τ can be decided by cross validation or simply fix 2 or $\log n$ such as AIC or BIC. They

also proposed another criterion named gMDL to eliminate the parameter as below

$$C_{gMDL}(RSS, \text{trace}(\mathcal{B})) = \log(J) + \frac{\text{trace}(\mathcal{B})}{n} \log(K), \tag{93}$$

$$J = \frac{RSS}{n - \text{trace}(\mathcal{B})} \tag{94}$$

$$K = \frac{\sum_{i=1}^{n} y_i^2 - RSS}{J\text{trace}(\mathcal{B})} \tag{95}$$

where RSS denotes the residual sum of squares as in Equation (92).

Then Sparse$L2$Boost algorithm is as follows:

1. Set $F_0 = 0$

2. For any $t = 1, \ldots, T$ Search for the best predictor

$$\mathcal{S}_t = \underset{1 \le j \le p}{\arg\min} T(Y, \mathcal{B}(\mathcal{S}))$$
$$\mathcal{B}_{\sqcup}(\mathcal{S}) = I - (I - \mathcal{H}_\nu \mathcal{S}_{\sqcup})(I - \nu \mathcal{H}_{\mathcal{S}_{\sqcup - \infty}}) \cdots (I - \mathcal{H}_\nu \mathcal{S}_\infty)$$
$$f_t(\boldsymbol{x}) = g_{\mathcal{S}_m(X,U)}(\boldsymbol{x})$$

3. Update

$$F_t(\boldsymbol{x}) = F_{t-1}(\boldsymbol{x}) + \nu f_t(\boldsymbol{x})$$

4. Repeat step 2 and 3 for a large number of iterations T.

5. Estimate the stopping iterations by

$$t^* = \underset{1 \le t \le T}{\arg\min} \mathcal{T}(Y, \text{trace}(\mathcal{B})).$$

Comparing to the ordinary Boost, SparseL_2Boost changed the criterion to choose the best $f_t(\boldsymbol{x})$ using the regularization term. Taking into account the model complexity in the criteria, Bühlmann and Yu (2006) considered that the final model could be sparser. Regarding the degree of freedom Hastie (2007) pointed that Equation (91) underestimates of the true degree of freedom using two real data set.

In this section, we overviewed the details of Boosting methods and their modifications. Boosting methods were studied from different view points. AdaBoost can be seen as additive logistic regression. Freund and Schapire (1997) introduced the bound of training error and generalization error. The bound of training gives an idea why training error approaches zero exponentially in terms of number of iterations. Schapire et al. (1998) presented the concept of margin, furthermore they showed the relation to the margin of Support Vector Machine. AdaBoost increases the margin even after training error reaches zero; this may account for the success at reducing generalization error however it was not always true. Rosset et al. (2004) extended the margin theory to the relation to Lasso.

Boosting methods were modified by several approaches. One of them is using different loss function, the second one is using regularization term with loss function. Number of iterations was considered to cause overfitting. Early stopping using cross validation was proposed as a solution. Loss functions and number of iterations have been studied for good classification performance however weak classifiers were not studied enough even the complexity of weak learners are in both upper bound of training error and generalization error. Our proposed method focus on the complexity of weak learners which we present further in the next section.

3 Sparse Learner Boosting

In this section, we introduce a new method, Sparse Learner Boosting. Sparse Learner Boosting prevents overfitting by reduction of weak learner candidates. First we mention the motivation of Sparse Learner Boosting. Then we detail the algorithm and examine the model complexity using VC dimension. We show the results of simulation studies using synthesized data and real data.

3.1 Motivation

Boosting methods were studied and improved as we reviewed in the previous section. The key ingredients of Boosting methods are loss function, number of iteration and weak learners. Choosing appropriate of them affects classification performance. A variety of loss functions were studied, early stopping was proposed to prevent overfitting. On the other hand, choosing weak learners was not studied enough. In gene expression data, size of p is large, hence the candidate weak learners becomes large as well. Using all of them can be superfluous. Therefore we propose a new Boosting method trimming the weak learner candidates named Sparse Learner Boosting (Pritchard (2010)).

3.2 Algorithm

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 by

$$\mathcal{F} = \{f_j(\boldsymbol{x}, a, b) = a \cdot \mathrm{sgn}(x_j - b) : j \in \{1, \cdots, p\}, b \in \mathbb{R}\}, \tag{96}$$

where a is given a value 1 or -1 and $b \in \mathbb{R}$ is a threshold value.

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 and false negative rate. We review the concept of false positive and false negative first then explain our implimentaion.

False Positive and False Negative

False positive and false negative are type of errors in which a statistical test wrongly rejects or accepts the null hypothesis. They are often used in diagnostics. In the context of a classification problem, considering the cancer diagnostics prediction model, if a patient who does not have cancer is incorrectly diagnosed as having cancer, the consequences may be patient distress plus the need for further investigation. Conversely, if a patient with cancer is diagnosed as healthy, the result may be premature death due to lack of treatment. In that case, false positive and false negative are considered separately. As an example, let us set that a patient whose label $y = +1$ has disease and $y = -1$ has healthy status. If a sick patient is diagnosed as healthy, it is called false negative. If a healthy patient is diagnosed as diseased, it is called false positive. Table 1 shows false positive and false negative in binary classification case. Figure 8 illustrates the relationship between false positive and false negative.

Table 1: False Positive and False Negative: Rows express the class label returned by classification rule. Columns express the true labels.

	$y = +1$	$y = -1$
$H(\boldsymbol{x}) = +1$	True Positive	False Positive
$H(\boldsymbol{x}) = -1$	False Negative	True False

The data points following the gray distribution and the white distribution have class label $+1$ and -1 respectively. Threshold is defined by a classifier. The right side of the threshold is classified as $+1$. The left side of the threshold is classified as -1. Choice of threshold 2 results in both false positive and false negative. Threshold 1 does not include any false negatives however a lot of false positive are in the result. Conversely threshold 3 does not include any false positive though the result contains a lot of false negative. Next we detail our proposed criterion which is named trim over lapping learners criterion to trim the initial set of weak learner candidates using false positive and false negative.

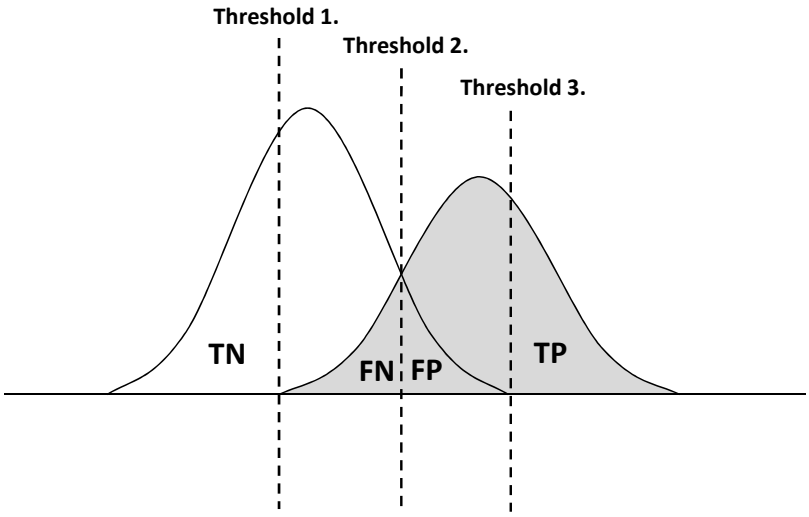

Figure 8: Example of false positive and false negative. Gray distribution presents $+1$ class distribution and white distribution presents -1 class distribution. Dashed lines show thresholds. False positive and false negative are related to where threshold is set.

3.2.1 Trim Overlapping Learners criterion

Firstly, we define false positive rate $\overline{\mathrm{FP}}(f)$ and false negative rate $\overline{\mathrm{FN}}(f)$ as follows:

$$\overline{\mathrm{FP}}(f) = \frac{1}{n_1} \sum_{i=1}^{n} I(f(\boldsymbol{x}_i) \neq y_i, y_i = -1) \tag{97}$$

$$\overline{\mathrm{FN}}(f) = \frac{1}{n_2} \sum_{i=1}^{n} I(f(\boldsymbol{x}_i) \neq y_i, y_i = +1), \tag{98}$$

where n_1 is the number of subjects having class label -1 and n_2 having class label $+1$. Trim overlapping learners criterion (TOL criterion) aims to trim weak learners which reside in the overlapping area. We define weak learners which are trimmed by TOL criterion as follows:

$$\mathcal{F}_\xi = \{f : \min(\overline{\mathrm{FP}}(f), \overline{\mathrm{FN}}(f)) \leq \xi, f \in \mathcal{F}, \xi \in \mathbb{R}\}. \tag{99}$$

Parameter ξ controls the sparseness. For instance, $\xi = 0$ means that no weak learner which has either false positive or false negative is included in the set of weak learners $\mathcal{F}_\xi$. If ξ is 1 which means that all weak learners remain and $\mathcal{F}_\xi$ is same as $\mathcal{F}_{\mathrm{ds}}$.

We illustrate a simple example of TOL criterion in Figure 9. The figure presents one dimension of data points. Black circles and white circles denote class labels $+1$ and -1 respectively. Dashed lines are thresholds which were defined by b in Equation (96). The right side of each threshold is classified as $+1$ and the left side of each threshold is classified as -1. In this figure we only consider the case of $a = 1$ for simplicity. Figure 9(a) presents the thresholds which are defined by Equation (96). Figure 9(b) shows weak learners that do not have any false positive or false negative. Top three weak learners do not classify black data points as $-$. Figure 9(c) represents trimmed weak learners in the case of $\xi = 0$. Four weak learners which have false positive or false negative are trimmed and five weak learners remain.

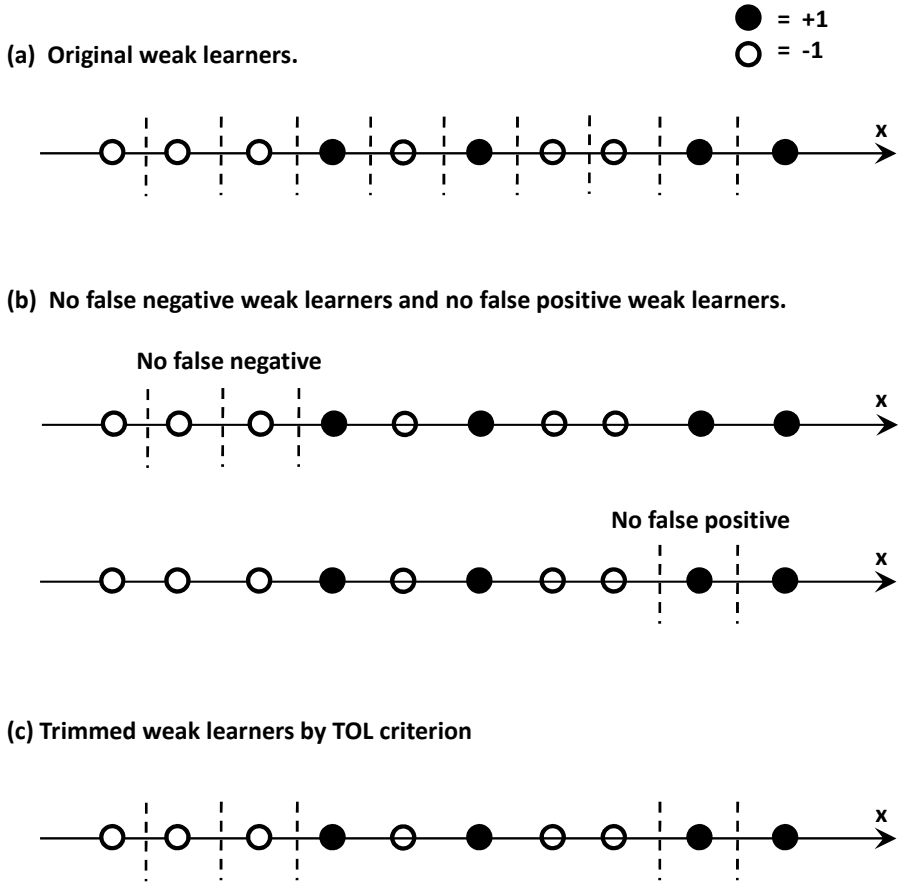

Figure 9: Example of TOL criterion using one dimensional simple data points. Black circles and white circles have class label $+1$ and -1 respectively. Dashed lines are thresholds which are used for weak learners. (a) Original weak learners. Thresholds are set between each data point. The left side of the threshold is classified -1 and the right side of the threshold is classified $+1$. (b) Weak learners which do not have any false positive or false negative. top three weak learners do not have any false negative. (c) Trimmed weak learners by TOL criterion. After trimmed weak learners, five weak learners remain; these do not have any false positive or false negative.

Figure 10 shows an example of weak learners with two dimensional synthesized data. 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 weak learners in overlapped areas are congested. TOL criterion trimmed those overcrowded weak learners nicely. The sparseness can be controlled by ξ. Sparse Learner Boosting uses

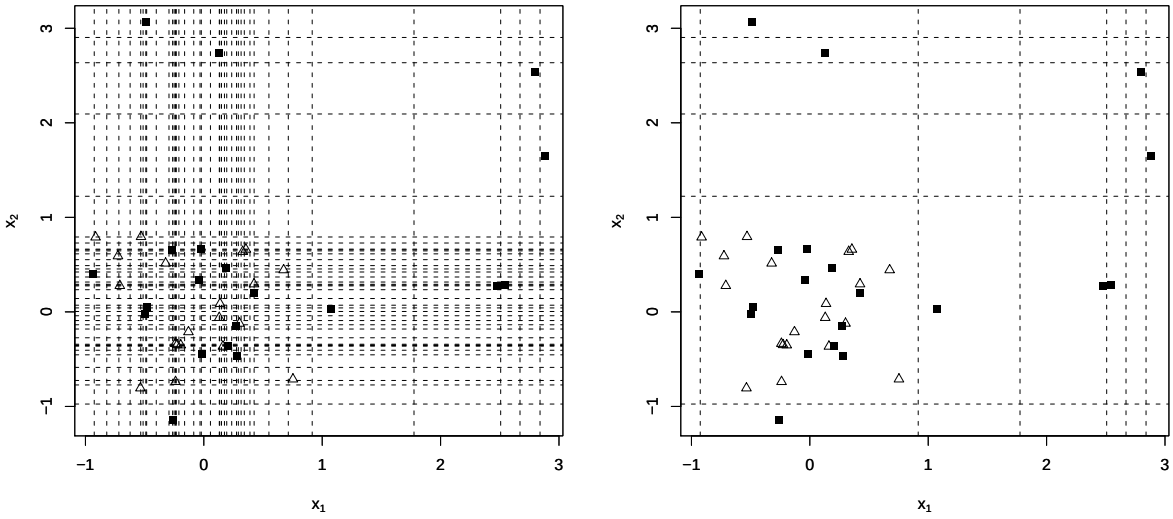

Figure 10: Typical example for weak learner candidates. The dashed lines denote thresholds which are used in weak learners. The left panel shows the weak learners for ordinary boosting. The right panel is for Sparse Learner Boosting where $\xi = 0$

these trimmed weak learner candidates. 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,

$$F(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t f_t(\boldsymbol{x}), \tag{100}$$

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

is defined by

$$c^* = \arg\min_c \sum_{i=1}^{n} I(\mathrm{sgn}(F(\boldsymbol{x}_i) - c) \neq y_i). \tag{101}$$

We presented the details of TOL criterion and Sparse Learner Boosting. Trimming weak learners using false positive and false negative is a simple procedure. Next we examine how our modification affects the final classification model from the view point of Vapnik-Chervonenkis dimension.

3.3 Complexity of Sparse Learner Boosting

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 (1971). VC dimension is a standard measure of classification model complexity. First we introduce an important concept "Shatter ".

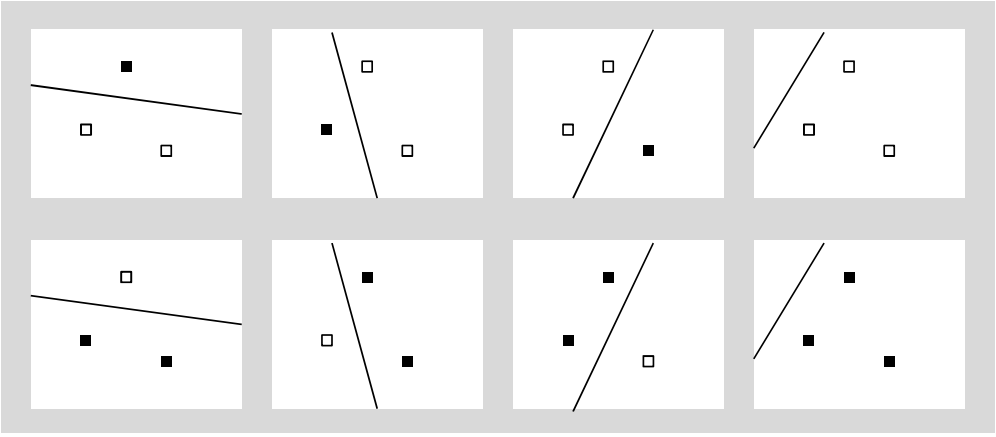

Figure 11: Simple examples for shatter concepts. These panels present possible labeling combinations where $n = 3$. If given function class can shatter possible labeling, it is called "shatter".

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 by

$$S_n(\mathcal{F}_{ds}) = \max_{\{x_1,\ldots,x_n\}\in\mathbb{R}^p} |\{x_1, \ldots, x_n\} \cap \{x | f(\boldsymbol{x}, a, b) = 1 | f(\boldsymbol{x}, a, b) \in \mathcal{F}_{ds}\}|. \quad (102)$$

VC dimension is defined by the maximum number of subjects which $\mathcal{F}_{ds}$ can shatter,

$$\mathrm{VC}(\mathcal{F}_{ds}) = \max_{n'} \{n' | S_{n'}(\mathcal{F}_{ds}) = 2^{n'}\}. \quad (103)$$

Figure 11 shows a simple example where $n = 3$. Possible labeling combination equals 8. Let $\mathcal{H} = \left\{g(\boldsymbol{x}) = I((\beta_0 + \boldsymbol{\beta}^T\boldsymbol{x}) > 0)\right\}$ is a class of functions. The function class $\mathcal{H}$ can shatter possible labeling combinations, therefore shatter coefficient is 8 and VC dimensions $\mathrm{VC}(\mathcal{H})$ is 3. 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:

$$2^n \leq 2pn. \quad (104)$$

Take logarithm with base 2,

$$n - \log_2 n \leq 1 + \log_2(p).$$

For any positive integer r

$$\frac{r}{2} \leq r - \log_2(p). \quad (105)$$

We show the details of Equation (105) in Appendix. From the above we have

$$\mathrm{VC}(\mathcal{F}_{ds}) \leq \lfloor 2(1 + \log_2 p) \rfloor, \tag{106}$$

where $\lfloor \cdot \rfloor$ is the floor function. See Kawakita and Eguchi (2008) 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 (104) for Sparse Learner Boosting is

$$2^n \leq 2p\delta n. \tag{107}$$

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

$$\mathrm{VC}(\mathcal{F}_\xi) \leq \lfloor 2(1 + \log_2 \delta p) \rfloor. \tag{108}$$

The Equation (108) expresses that the trimmed weak learners by TOL criterion is equivalent to the δp variable selection. And also as a result we 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 able to have a less complex model than ordinary boosting methods.

3.4 Numerical experiments

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. Each experiment, we used cross validation to decide the number of iterations. Before going into the details of the experiment, we review the cross validation method.

3.4.1 Cross Validation

Cross validation is the one of the most widely used methods for estimating prediction error which we refer as CV error in this thesis. If we have enough data, we set aside a validation set and use it to assess the performance of a prediction model. However often we face a situation where we do not have enough data. To handle this situation, there we split the data into K roughly equal-sized parts. We build a model using $K-1$ parts data then calculate the prediction error by Kth part data. This process is repeated K times and average K time prediction errors. Let $\kappa : \{1, \ldots, n\} \mapsto \{1, \ldots, K\}$ be an indexing function that splits data into K parts. Denote $\hat{f}^{-k(i)}(x)$ as the fitted function without kth part data. The kth part is used for calculating the CV error. The CV error is

$$CV(\hat{f}) = \frac{1}{n} \sum_{i=1}^{n} L(\hat{f}^{-k(i)}(x_i)). \tag{109}$$

The case $K = n$ is known as leave-one-out cross validation. In this case $k(i) = i$, and ith data is used for validation and the model is constructed using all the data except i. Hastie et al. (2005) mentioned that with $K = n$, the cross validation estimator is approximately unbiased for the true prediction error, but can have high variance because the n training sets are so similar each other. They also mentioned that the number of K should be considered by the number of observation. For example, if number of observations is 200, five-fold cross validation would estimate the performance of classifier overtraining sets of size 160, however if the number of

observation is 50, five-fold cross validation estimates the performance over training sets of size 40.

Cross validation is also used to choose parameter value. Dividing data into training data and validation data, perform classification with multiple parameters then evaluate the performance using validation data.

3.4.2 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 by $\{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, 100\}$ 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 Figure 12, 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.

3.4.3 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 $N(\boldsymbol{\mu}_{+1}, 0.5\boldsymbol{I}_p)$.

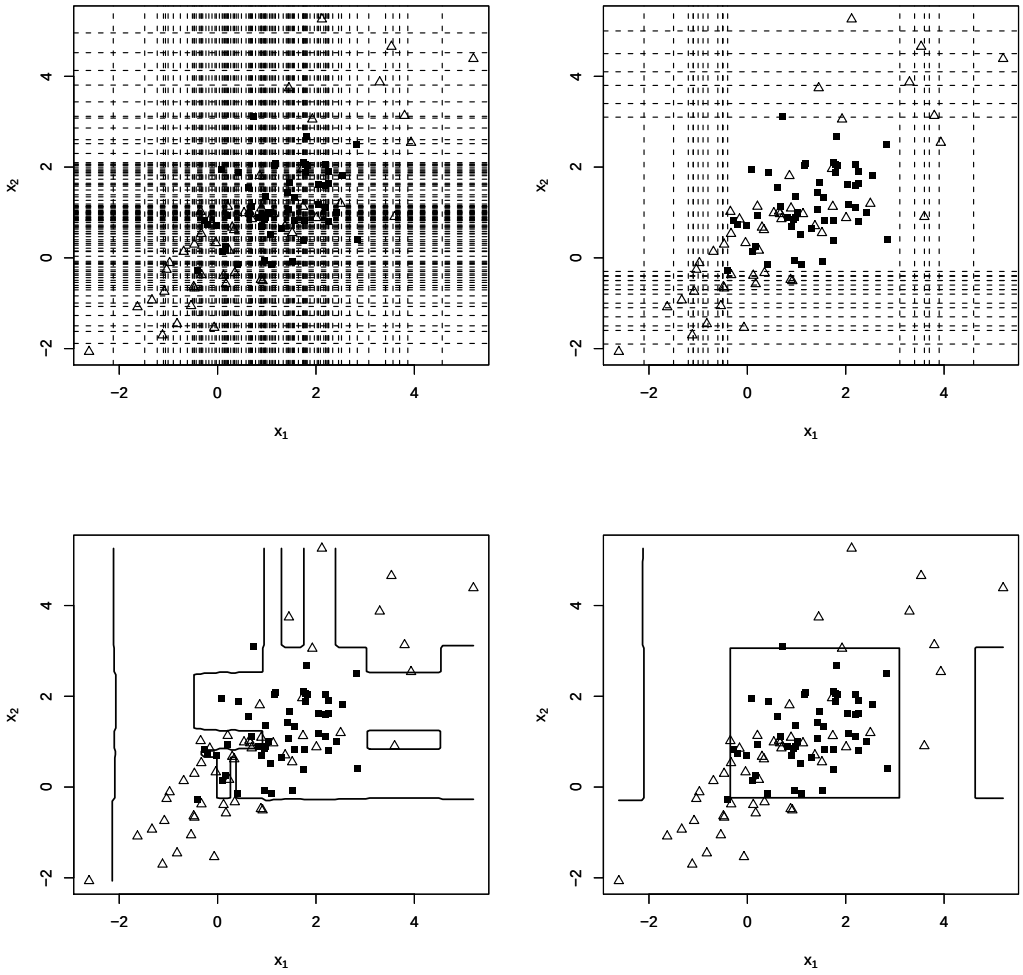

Figure 12: The data generated by case 1. Squares and triangles denote data points with class labels 1 and -1 respectively. The upper left panel is weak learners for AdaBoost. The upper right panel shows weak learners for Sparse Learner AdaBoost. The lower left panel shows boundary which was built by AdaBoost. The lower right panel is the boundary for Sparse Learner AdaBoost.

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 Equation (99). 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 2 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.

Table 2: 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

Table 3 presents the average of sparseness for each data. Sparseness δ was

calculated by number of trimmed weak learners divided by number of original weak learners. Parameter ξ for TOL criterion is 0. This means that weak learners which does not have any false positive and false negative are chosen.

Table 3: Average sparseness δ for each data. Sparseness shows the degree of sparseness which is trimmed by TOL criterion. Parameter ξ is 0 for all data.

		Variables			
		10	100	300	500
Sparseness	(δ)	0.1265	0.1266	0.1263	0.1259

3.5 Real data analysis

We used the gene expression data reported by van't Veer et al. (2002). 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. Before going into the details of data processing explanation, we describe about Microarray technology.

3.5.1 Microarray technology

Microarray is an important technology and rapidly glow the usage in a variety of area. Microarray measures the amount of mRNA or gene expression. There are two major technologies available for gene expression measurement. One is GeneChip system provided by Affymetrix Inc. GeneChip uses prefabricated short length of oligonucleotide. Another is cDNA array which is originally developed by Schena et al. (1996). We briefly mention both technologies.

GeneChip uses a pair of short length of oligonucleotide attaching on the solid surface, such as glass, plastic or silicon. The short pair of oligonucleotides is called

probe pair. Each probe pair is composed of a perfect match (PM) probe and a mismatch (MM) probe. PM is a section of the mRNA of interest and MM is created by changing the middle (13th) base of the PM with the intention of measuring non-specific binding. mRNA sample is collected from subjects such as cancer patients then fluorescence dye is labeled. The labeled mRNA sample is hybridized to spot of microarray if the short oligonucleotide is matched with the mRNA sample. If the labeled mRNA and the probe match perfectly, they bind strongly otherwise they bind weakly. Those weak binding or non-specific binding is washed out by washing buffer then only strongly bound mRNA sample fluorescence is measured by a scanner. Scanned measurement needs further processing before using analysis such as outlier detection, background subtraction and normalization. These process is called pre-processing.

In the early stage of microarray, the quality of microarray measurements contains a lot of variance. Therefore pre-processing was very active research area. Affymetrix recommended to use both PM and MM probes to subtract non-specific binding and implement MASS algorithm to their software however Irizarry et al. (2003) and Naef et al. (2001) pointed out that normalization model considering MM captures non-specific affect more than in real. Currently Robust Multichip Average (RMA) which is introduced by Irizarry et al. (2003) is also widely used.

cDNA array uses glass slide to attach also short oligonucleotides probes. cDNA array uses the inkjet printing technology. GeneChip use one color fluorescence dye, on the other hand, cDNA utilize two different color fluorescence dye. One of the color is for control mRNA and another color is for treatment mRNA. Both samples are hybridized on the same array. Scanner detects both fluorescence dye separately. Data processing is slightly different from GeneChip. As cDNA uses two fluorescence dye, scanned data is normally treated as ratio data of treatment over control.

Microarray technology was improved in the last decades including to reduce the variance, normalization procedure does not affect that much comparing before. An interest of research area moved to data analysis.

3.5.2 Pre-processing

We used the gene expression data reported by van't Veer et al. (2002). 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 78 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. We applied the same filtering criteria which van't Veer et al. (2002) used. A two-fold change of expression with a P-value < 0.01 in five or more patients yielded approximately 5000 genes. For calculation of p-value, see Roberts et al. (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. This data is available at http://www.rii.com/publications/default.htm.

3.5.3 Boosting method

We applied Sparse Learner Boosting and non-Sparse Learner Boosting for both AdaBoost and η-Boost using the top 200 probes. 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.

We compared Sparse Learner Boosting with DLDA as well.

3.5.4 DLDA

Fisher linear discriminant analysis is lead by maximum likelihood using mean vector for each class $\boldsymbol{\mu}_y$ and common covariance matrix $\widehat{\boldsymbol{\Sigma}}$ assuming that $\boldsymbol{x}$ has p dimensions and the class label y is given, $\boldsymbol{x}$ follows multivariate normal distribution $N(\boldsymbol{\mu}_y, \boldsymbol{\Sigma})$. The discriminant function is

$$F(\boldsymbol{x}) = \operatorname{sgn}(\widehat{\boldsymbol{\alpha}}_1^T \boldsymbol{x} - \widehat{\alpha}_0). \tag{110}$$

where $\widehat{\boldsymbol{\alpha}}_1 = \widehat{\boldsymbol{\Sigma}}^{-1}(\widehat{\boldsymbol{\mu}}_{+1} - \widehat{\boldsymbol{\mu}}_{-1})$, $\widehat{\alpha}_0 = \frac{1}{2}(\widehat{\boldsymbol{\mu}}_{+1} - \widehat{\boldsymbol{\mu}}_{-1})^T \widehat{\boldsymbol{\Sigma}}^{-1}(\widehat{\boldsymbol{\mu}}_{+1} + \widehat{\boldsymbol{\mu}}_{-1}) + \log \widehat{p}(y = +1) - \log \widehat{p}(y = -1)$, where $\widehat{\boldsymbol{\Sigma}} = \operatorname{diag}(\sigma_1^2, \ldots, \sigma_p^2)$. If the dimension size is larger than the sample size, $\widehat{\boldsymbol{\Sigma}}^{-1}$ does not exist. Therefore either regularized inverse matrix or diagonal matrix for $\widehat{\boldsymbol{\Sigma}}$ was used.

3.6 Result and discussion

The result was shown in the table 4. The Sparse AdaBoost showed 0.158 test error. Neither of non-Sparse AdaBoost nor non-Sparse η-Boost could not achieve the error rate given by Sparse Learner Boosting. DLDA showed 0.211 test error. Sparse AdaBoost could show the better performance than DLDA. The sparseness δ was 0.084. This means that Sparse Learner Boosting reduced dimension from 200 to 16. We plotted the top five variables so that overlap between two classes is examined. Even top five variables, we can see that the two class data points are heavily overlapped. We suppose that non-sparse Boosting overfit the data therefore they could not outperformed the sparse learner boosting because of the heavy overlap.

Simple modification of the current Boosting methods was performed, resulting in

Table 4: Test error rates for real data. The variables were sorted by correlation to the class label. 200 variables were used.

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

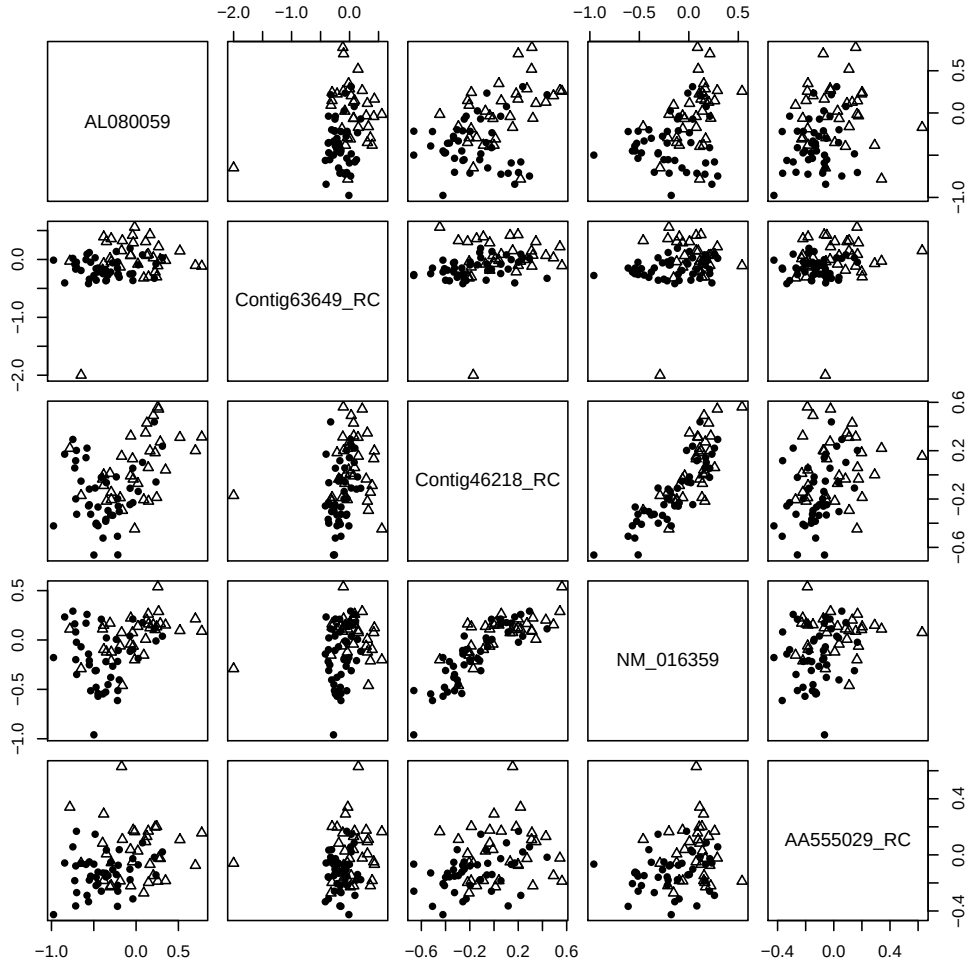

Figure 13: The top five variables are plotted. Triangles show poor prognosis patients and circles show good prognosis patients.

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 (2006) 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.

We note that Sparse Learner Boosting works well when data points are heavily overlapped between classes however when only certain amount of data points are overlapped, we cannot expect improvement of classification. The degree of overlapp is one of future works. In conclusion, we propose a new boosting method, Sparse Learner Boosting, and confirm its ability to analyze high-dimensional data.

3.7 Future work

We focused on the complexity of weak learners and developed criterion to trim the set of weak learners to improve classification performance. We believe that our approach leads further modification and improvement to Boosting methods. One of further possible modifications is trying a different criterion to trim weak learners. For instance our TOL criterion considered both false positive and false negative in a same way. Using one of them can be a good criteria. Our idea is simple and easy to implement to different Boosting methods, plug-in to different Boosting methods is also possible.

4 Existence of multiple predictive optimum gene sets

In this section, we address another challenge to be aware of in Microarray data analysis. As a result of technological progress, the possible number of gene expression measurements in one microarray has been rapidly growing, however many genes are not differentially expressed across subjects to any significant degree. These genes are irrelevant for classification purpose. Many feature selection methods are considered by many authors for example Ben-Dor et al. (2000) and Dudoit and Fridlyand (2002). Each method sets some criteria and calculates scores for each variables then decides which combination of gene sets is the best set. As microarray usage for clinical research is becoming popular, different researchers in different laboratories followed the similar procedure or tried different procedures. They expected to see similar genes being selected however the overlap was small because there is no unique gene set in any single classification problem.

We talk about this problem in the following sections. First we review the history of classification using microarray. Then address the details of feature selection methods and we show the existence of multiple optimum gene sets using real data.

4.1 Current classification analysis using microarray data

Golub et al. (1999) performed leukemia subclass classification using microarray. They classified acute myeloid leukemia (AML) into acute lymphoblastic leukemia (ALL). Their analysis gave a great hope to scientists because of the possibility for clinical usage of microarray. Following them, Armstrong et al. (2001) classified myelomonocytic leukemia (MML) patients from ALL/AML patients. Bittner et al. (2000) studied subclass classification for melanoma patients and Perou et al. (2000)

performed subclass classification for breast cancer patients. Further prognosis prediction models are proposed for breast cancer patients however some concerns are pointed out. Some papers regarding breast cancer prognosis prediction were published however the genes which were used for the prediction models do not match up with each other. Some reasons are considered such as different types of microarray, different data pre-processing, different data analysis and different disease status of patients. Ein-Dor et al. (2005) studied this discrepancy using data of van't Veer et al. (2002). In the original paper, van't Veer et al. (2002) sorted genes by correlation coefficient with outcome label. Then determined the top 70 genes as the gene set which has good predictive power. Ein-Dor et al. (2005) sorted genes in the same way but they tested gene sets such as 71st through 140th. They found that several gene sets could show the similar performance except for the top 70 genes. The absolute value of correlation coefficient between the top 231 genes and outcome is 0.3 to 0.5. One reason why several gene sets shows similar performance is that top 231 genes show similar correlations with outcome, Ein-Dor et al. (2005) considered.

4.2 Prediction concordance with small overlap gene sets

Fan et al. (2006) compared the predictions derived from different gene sets with small overlap in terms of gene identity. They obtained a single data set of 295 samples and applied five classification models, intrinsic subtype (Sorlie et al. (2003), Hu et al. (2006)), recurrence score, 70-Gene profile (van't Veer et al. (2002),van de Vijver et al. (2002)), wound response (Chang et al. (2005),Chang et al. (2004)) and two-gene ratio (Goetz et al. (2006)). They found that most models had high rates of concordance in their outcome predictions for the individual samples. They concluded that overlap in gene identity among gene expression data is not a good measure of reproducibility. The classification of individual samples is the relevant

measure of concordance. They also mentioned that even with different gene sets being used as predictors, they each track a common set of biologic characteristics that are present in different group of patients with breast cancer, resulting in similar predictions of outcome.

From their conclusions, the existence of multiple predictive optimum gene sets is no longer a concern. The selection of low overlap gene sets among different gene expression data occurs because of the large variety of given data. On top of that, the variety of feature selection methods is related to the different gene set selection. Next we overview the current feature selection methods for microarray data.

4.3 Feature selection methods

Reducing of dimension size is necessary as superfluous features can cause overfitting and interpretation of classification model becomes difficult. Reducing dimension while keeping relevant features is important. There are some feature selection methods proposed. Saeys et al. (2007) provided a taxonomy of feature selection methods and discussed their use, advantages and disadvantages. They mentioned the objectives of feature selection (a) to avoid overfitting and improve model performance, i.e. prediction performance in the case of supervised classification and better cluster detection in the case of clustering, (b) to provide faster and more cost-effective models and (c) to gain a deeper insight into the underlying processes that generated the data. Table (5) provides their taxonomy of feature selection methods. In the context of classification, feature selection methods are organized into three categories; filter methods, wrapper methods and embedded methods. Feature selection methods are categorized depending on how they are combined with the construction of a classification model.

Filter methods calculate statistics such as t-statistics then filter out those which

does not meet the threshold value. Advantages of filter methods are easy implementation, computational simplicity and speed. Filter methods are independent of classification methods; therefore different classifiers can be used. Two of the disadvantages of filter methods are that they ignore the interaction with the classification model and most of the proposed methods are univariate.

Whereas filter techniques treat the problem of finding a good feature subset independently of the model selection step, wrapper methods embed the model hypothesis search within the feature subset search. Wrapper methods search all feature subsets, the feature subsets space grows exponentially with the number of features. One of advantages of wrapper methods includes the interaction between feature subset search and model selection. Computational burden is one of the disadvantages of this approaches, especially if building the classifier has a high computational cost.

The third class of feature selection is embedded techniques. Like the wrapper methods, embedded techniques search for an optimal subset of features with classifier construction. Thus embedded approaches are also specific to a given learning algorithm. The difference from wrapper methods is that embedded techniques are guided by the learning process. Whereas wrapper methods search all possible combinations of gene sets, embedded techniques search for the combination based on a criteria. This enables reduction in computational burden.

4.4 Feature selection using annotation information

In the previous subsection, we mentioned several feature selection methods. Those methods use given numerical values or gene expression measurements. There is a case which use another information to choose genes for predictors. van't Veer et al. (2002) used filter methods then found 70 genes which showed high correlation with outcome however the function of the some of 70 genes were not known. After the

Table 5: A taxonomy of feature selection techniques. Main three feature selections are addressed. Each type has subcategory. Advantages, disadvantages and example methods are shown.

Model Search	Advantages	Disadvantages	Examples
Filter	Univariate		
	Fast Scalable Independent of the classifier Multivariate	Ignores feature dependencies Ignores interaction with the classifier	χ^2 Euclidean distance t-test Information gain
	Models feature dependencies Independent of the classifier Better computational complexity than wrapper methods	Slower than univariate techniques Less scalable than univariate techniques Ignores interaction with the classifier	Correlation-based feature selection Markov blanket filter Fast correlation-based feature selection
Wrapper	Deterministic		
	Simple Interacts with the classifier Models feature dependencies Less computationally intensive than randomized methods Randomized	Risk of overfitting More prone than randomized algorithms to getting stuck in a local optimum Classifier dependent selection	Sequential forward selection Sequential backward selection
	Less prone to local optima Interacts with the classifier Models feature dependencies	Computationally intensive Classifier dependent selection Higher risk of overfitting than deterministic algorithms	Simulated annealing Randomized hill climbing Genetic algorithms Estimation of distribution algorithms
Embedded	Interacts with the classifier Better computational complexity than wrapper methods Models feature dependencies	Classifier dependent selection	Decision trees Weighted naive Bayes RFE-SVM

genome project was completed, sequence information has been available online. To make use of it most, related biological information is also added to the sequence information if it is already discovered, such as name of the gene, what protein is produced by the gene and function of the gene and so on. The information is called annotation. The annotation information is used widely to help understanding of the role of the gene. All of the genome sequences have been read however still a lot of gene function is unclear. Therefore biologists prefer to using genes whose function are known for predictors of the classification model if the performance is same or similar. Fan et al. (2006) compared five classification algorithms, intrinsic subtype, recurrence score, 70-Gene profile, wound response and two-gene ratio, as we mentioned in an earlier section, three of them, intrinsic subtype, recurrence score and wound response used annotation information to choose genes for the classification model. Using annotation information for the selection of predictor can be helpful from the view point of interpretation however there is a possibility those genes does not show statistical significant difference between the classes.

4.5 Multiple predictive optimum gene sets

In an earlier section, we described that the existence of multiple optimum gene sets. Next we show an example using the data of van't Veer et al. (2002) with several classification methods.

4.6 Experiment

We performed similar experiments as Ein-Dor et al. (2005). We used three classification algorithms, AdaBoost, van't Veer Method and Diagonal Linear Discriminant Analysis (DLDA) to examine the existence of multiple optimum gene sets. van't Veer Method is the same method which van't Veer used in their paper. We call the

method van't Veer method in this thesis. The data is sorted by the absolute value of correlation coefficient with outcome then built the classification model using 70 probes, such as top 1 through 70, 6 through 75 and so on. We performed 3-fold cross validation to the training data to decide the number of the iterations. Then test error rate was calculated using the test data. We explain the details of van't Veer Method.

4.6.1 van't Veer Method

van't Veer Method is the same method as in the paper of van't Veer et al. (2002). The van't Veer Method constructs the discriminant function by the correlation between the vector of class which shows good recurrence and the given data. 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 $+1$. The data included 97 primary breast cancer patients. The data which has class label $+1$ is denoted $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n1})$ and the data which has class label -1 is referred to as $(\boldsymbol{x}_{n1+1}, \ldots, \boldsymbol{x}_n)$. The average vector of data which has class label $+1$ is

$$\bar{\boldsymbol{x}}_1 = \frac{1}{n1} \sum_{i=1}^{n1} \boldsymbol{x}_i, \tag{111}$$

and the correlation between the given data and Equation (111) is

$$S(\boldsymbol{x}, \bar{\boldsymbol{x}}_1) = \frac{|\sum_{j=1}^{p'} (x_j - \bar{x})(\bar{x}_{1j} - \tilde{x}_1)|}{\sqrt{\sum_{j=1}^{p'} (x_j - \bar{x})^2} \sqrt{\sum_{j=1}^{p'} (\bar{x}_{1j} - \tilde{x}_1)^2}}. \tag{112}$$

Then

$$\boldsymbol{x} = (x_1, \ldots, x_{p'})^T, \qquad \bar{x} = \frac{1}{p'} \sum_{j=1}^{p'} x_j, \tag{113}$$

$$\bar{\boldsymbol{x}}_1 = (\bar{x}_{11}, \ldots, \bar{x}_{1p'})^T, \qquad \tilde{x}_1 = \frac{1}{p'} \sum_{j=1}^{p'} \bar{x}_{1j}, \tag{114}$$

where p' is the number of probes which were used for calculation. Next sort the training data according to the correlation with Equation (112) as follows:

$$S(\boldsymbol{x}_1, \bar{\boldsymbol{x}}_1) \geq S(\boldsymbol{x}_2, \bar{\boldsymbol{x}}_1) \geq, \ldots, \geq S(\boldsymbol{x}_N, \bar{\boldsymbol{x}}_1). \tag{115}$$

Set the threshold which makes the false positive rate less than 10%. Calculate the value of Equation (112) for the test data in the same manner. Then apply the threshold which is set by the training data.

4.7 Result and discussion

We show the results the van't Veer Method, DLDA and AdaBoost in figure (14), (15) and (16) respectively. The x-axis is the number of probes which were used for building the model. They y-axis is the test error. The probes were ordered by their correlation coefficient with the label. The first probes had the highest correlation coefficient. In the figure (14), the performance of classification was not deteriorated much by using lower correlation coefficient variables. Also the test error using 141 through 210 probes showed the lowest test error or 5%.

Figure 15 shows the DLDA result. As in the van't Veer Method, the test error was not decreased by the use of lower correlation probes. Through the all probes, training error was approximately 20%. The lowest training error was 16%. We consider that the overlap between class label was heavy. Therefore linear discriminant boundary

limit precision in classifying two group data.

The result of AdaBoost is figure (16). As AdaBoost learns from the training data in a greedy way, the training error is always 0%. The test error rate with other methods reached approximately 20 % on most probes. AdaBoost reached this level for some probes. We show the learning progress of AdaBoost in figure (17). Training error reached 0% very quickly. This suggests that the algorithm finished learning at a very early stage of iterations.

To investigate further why AdaBoost did not show the good performance using the top 231 probes, we randomly changed the order of probes then applied AdaBoost. When we use only 6 probes, we saw that the test error rate reached 10 %. The progress of the error is figure (18). Comparing the result using the top 70 probes, AdaBoost learns slower.

We applied Sparse Learner Boost to test the performance. As Sparse Learner Boost shows good prediction performance in high dimension, we used 100 probes. We searched from top 1 through 5000 probes. When we applied the top 2401 through 2500 probes, the minimum test error rate reached 10 %. We plotted 2401 probe through 2405 probe in figure (20). As in the top 70 probes, heavy overlapping was confirmed. Furthermore, less correlation than with the top 70 was observed. We need further study to find if the correlation is related to performance however this result suggests that Sparse Learner Boosting can perform well even using lower correlation with outcome. As we reviewed in an early section, there are a variety of feature selection methods and some method does not take into account correlation with outcome label always.

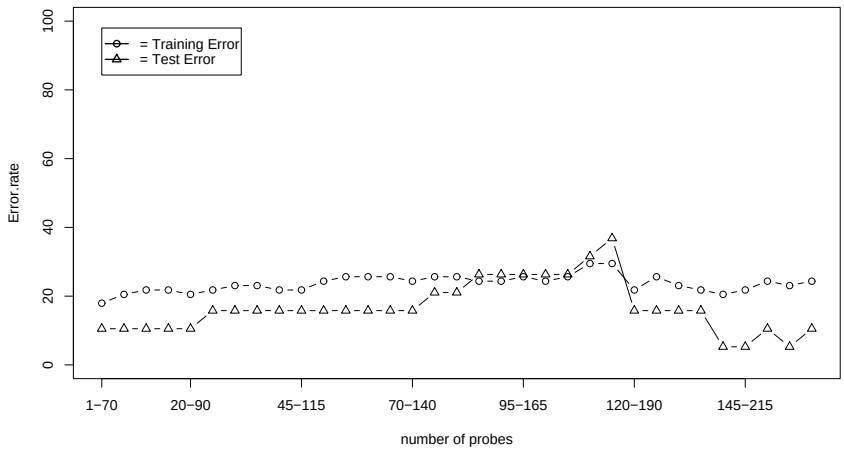

Figure 14: The training error and test error by van't Veer Method. The x-axis is the number of genes used to construct the discriminative function. The y-axis is the error rate. Circles show the error rate for the training data and triangles show the error rate for the test data.

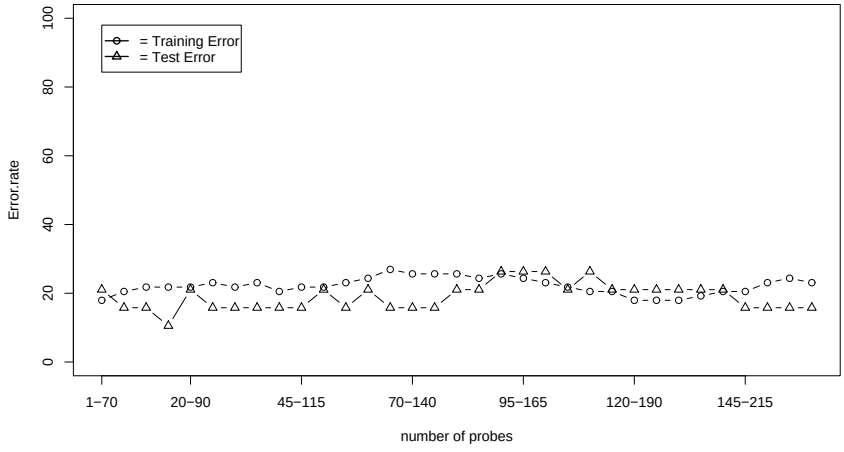

Figure 15: The training error and test error by DLDA. The x-axis is the number of genes used to construct the discriminative function. The y-axis is the error rate. Circles show the error rate for the training data and triangles show the error rate for the test data.

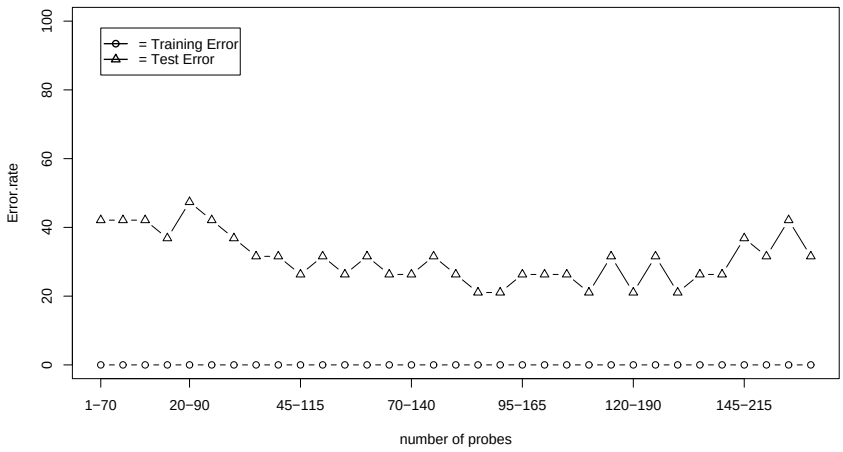

Figure 16: The training error and test error by AdaBoost. The x-axis is the number of genes used to construct the discriminative function. The y-axis is the error rate. Circle shows the error rate for the training data and triangles show the error rate for the test data.

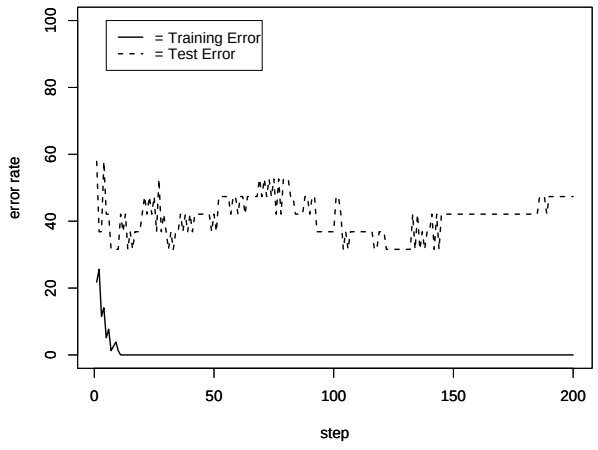

Figure 17: The progress of learning in AdaBoost. The x-axis is the number of steps in the learning process, the y-axis is the error rate. The solid line is the training error and the dashed line is the progress of the test error. The top 70 genes are used.

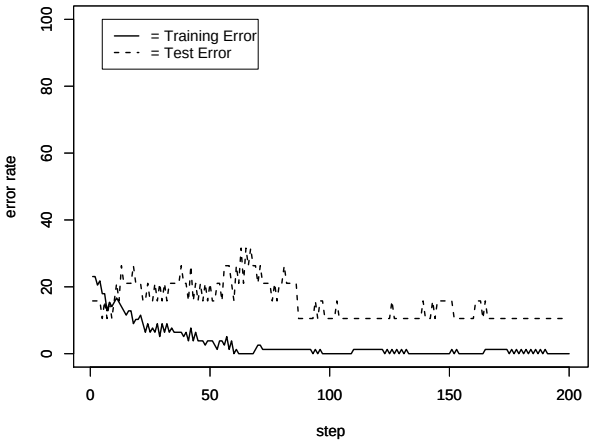

Figure 18: The progress of learning in AdaBoost. The x-axis is the number of step in the learning process, the y-axis is the error rate. The solid line is the training error and the dashed line is the progress of the test error rate. 6 randomly selected probes were used.

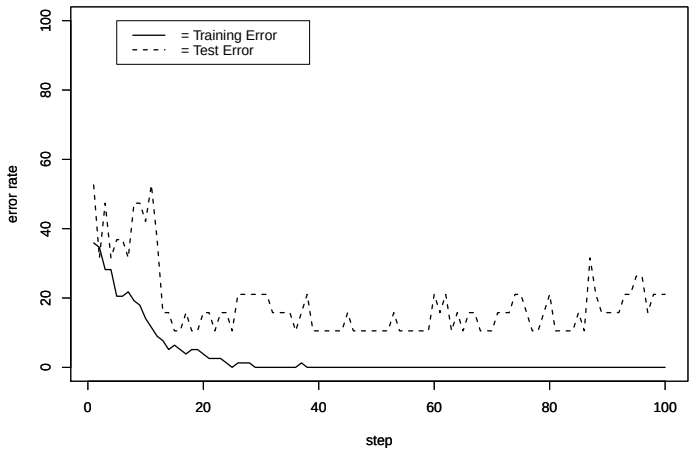

Figure 19: The progress of learning in Sparse Learner. The x-axis is the number of step in the learning process, the y-axis is the test error. The solid line is the training error and the dashed line is the progress of the test error. The top 2401 through 2500 genes are used.

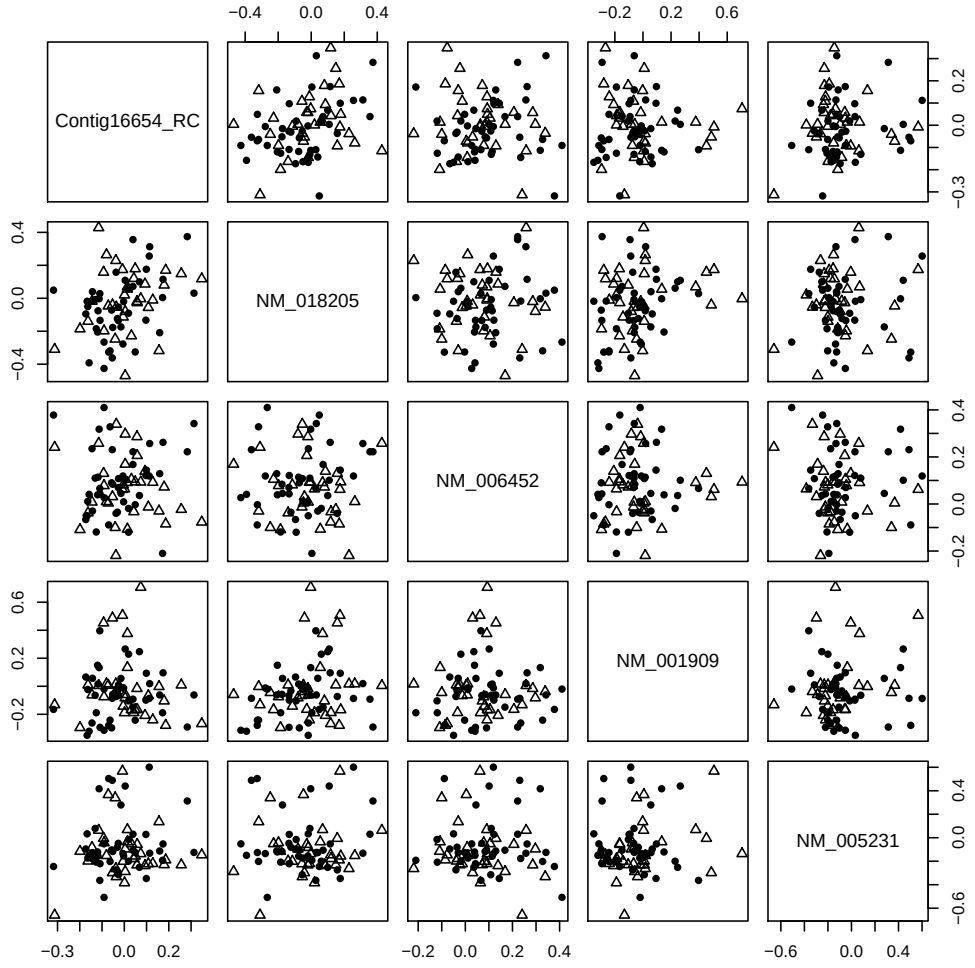

Figure 20: The top 2401 through 2405 variables which Sparse Learner Boost showed the good performance are plotted. Triangles show poor prognosis patients and circles show good prognosis patients.

5 Concluding remarks

We discussed two topics regarding boosting methods in bioinformatics application. Boosting methods have been widely used in a variety of areas because of their high performance however analyzing omics data remains very challenging. $p \gg n$ issue becomes well known by statisticians as an important problem. There are many modifications to tackle with this problem however those modifications are not always applicable. This is true in boosting as well. The greedy learning nature of Boosting became an issue in bioinformatics data. Friedman et al. (2000) pointed out that in the context of Boosting all weak learners are not equivalent, and there is no universally best choice over all situations. Weak learners are ingredients of discriminant functions of Boosting therefore their choice is an important problem. Hence we proposed Sparse Learner Boosting which trims the weak learner candidates based on given data using the false positive rate and false negative rate. The simulation study and real data experiments confirmed the ability of performance. Gene expression data contains a lot of biological variance for example status of disease or different type of treatment. Controlling all of these situations is impossible, however further study of what kind of data set or gene set is good for AdaBoost and what is for Sparse Learner Boosting is possible. We hope further study will be able to solve this issue.

Another important issue we addressed in this thesis is the existence of multiple optimum gene sets. Because dimension size of gene expression data is usually huge, possible combinations of gene sets is astronomical. Usually ranking and filtering methods is just one way to return one combination of gene set. Yet there are more combinations that can return the same or similar performance. From our empirical study, we observed that ranking criteria, which we used correlation with outcome, was not the best for Boosting. The performance of classification does not

depend on the ranking. As we mentioned the interpretability is important beside the classification performance. It would be ideal to retain both balance then return reasonable predictor variables and performance during the Boosting algorithm.

Boosting is a simple procedure thus includes a lot of potential to improve performance for high-dimensional data because of easy implementation and modification. We strongly hope that further studies progress Boosting enhancement and more application of statistics in Bioinformatics.

6 Acknowledgement

I cannot find the right word how to express my appreciation to Professor Eguchi. I deeply appreciate his consistent support and suggestions. Without his supervision, I could not have gone this far. I also want to say thank you Dr. Komori for his support and many helpful comments. And I want to acknowledge my husband for his unwavering support. Finally I want to appreciate all professors in ISM giving me a great opportunity to study statistics.

Appendix

A Fisher Linear Discriminant Analysis and its variants

Fisher Linear discriminant analysis (FLDA) is the most popular linear classification method which is proposed by Fisher (1936). Suppose that a probability density function of $\boldsymbol{x}$ is $g_y(\boldsymbol{x})$ and let π_y is the prior probability of each class with $\pi_{+1}+\pi_{-1} = 1$. Let $g_y(\boldsymbol{x})$ follow multivariate Gaussian distributions,

$$g_y(\boldsymbol{x}) = \frac{1}{2\pi^{p/2}|\Sigma_y|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_y)^T\Sigma_y^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_y)\right\}. \tag{116}$$

Assuming that both the conditional distributions have a common covariance matrix $\Sigma = \Sigma_{-1} = \Sigma_{+1}$. We write class posterior $\mathrm{P}(\hat{y} = y|X = \boldsymbol{x})$. Bayes optimal solution is given maximum different between Comparing two classes, using log ratio and

$$\begin{aligned}
\log\frac{\mathrm{P}(\hat{y}=+1|X=\boldsymbol{x})}{\mathrm{P}(\hat{y}=-1|X=\boldsymbol{x})} &= \log\frac{g_{+1}(\boldsymbol{x})}{g_{-1}(\boldsymbol{x})} + \log\frac{\pi_{+1}}{\pi_{-1}} \\
&= \log\frac{\pi_{+1}}{\pi_{-1}} - \frac{1}{2}\{(\boldsymbol{x}-\boldsymbol{\mu}_{+1})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_{+1}) \\
&\quad +(\boldsymbol{x}-\boldsymbol{\mu}_{-1})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_{-1})\} \\
&= \log\frac{\pi_{+1}}{\pi_{-1}} + \boldsymbol{x}^T\Sigma^{-1}(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1}) \\
&\quad -\frac{1}{2}(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1})^T\Sigma^{-1}(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1}),
\end{aligned} \tag{117}$$

where $\boldsymbol{\mu}_y$ is a mean of the conditional distributions. The equality of covariance matrices causes the second order terms to cancel. This leads to the linear discriminant function

$$F(\boldsymbol{x}) = \boldsymbol{x}^T\Sigma^{-1}(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1}) - \frac{1}{2}(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1})^T\Sigma^{-1}(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1}) + \log\frac{\pi_{+1}}{\pi_{-1}}. \tag{118}$$

The $\mu_{\pm 1}$ and Σ are estimated by the given data set as follows:

$$\hat{\mu}_{+1} = \frac{1}{n_1} \sum_{i=1}^{n} I(y_i = +1) \boldsymbol{x}_i \tag{119}$$

$$\hat{\mu}_{-1} = \frac{1}{n_2} \sum_{i=1}^{n} I(y_i = -1) \boldsymbol{x}_i \tag{120}$$

$$\hat{\Sigma} = \frac{n_1}{n-2} \sum_{i=1}^{n} I(y_i = +1)(\boldsymbol{x}_i - \hat{\mu}_{+1})(\boldsymbol{x}_i - \hat{\mu}_{+1})^T + \tag{121}$$

$$\frac{n_2}{n-2} \sum_{i=1}^{n} I(y_i = -1)(\boldsymbol{x}_i - \hat{\mu}_{-1})(\boldsymbol{x}_i - \hat{\mu}_{-1})^T, \tag{122}$$

where n_1 is the number of subjects having class label $+1$ and n_2 is the number of subjects having class label -1.

A new feature vector $\boldsymbol{x}$ is classified $\hat{y} = +1$ if $F(\boldsymbol{x}) > 0$. FDA assume that both the conditional distributions have a common covariance matrix however it often different. In that case Quadratic Discriminant Analysis (QDA) can be used which we will address in the next section.

A.1 Quadratic Discriminant Analysis

FLDA is a special case assuming that the both classes have common variance covariance matrices. If the variance covariant matrices Σ_{-1} and Σ_{+1} are not assumed to be equal, then convenient cancellations in Equation (117) do not occur. Therefore when we assume that both covariance matrices are not equal, we get

$$\begin{aligned} \log \frac{\mathrm{P}(\hat{y} = +1 | X = \boldsymbol{x})}{\mathrm{P}(\hat{y} = -1 | X = \boldsymbol{x})} &= \log \frac{g_{+1}(\boldsymbol{x})}{g_{-1}(\boldsymbol{x})} + \log \frac{\pi_{+1}}{\pi_{-1}} \\ &= \log \frac{\pi_{+1}}{\pi_{-1}} + \frac{1}{2}\{\log |\Sigma_{-1}| - \log |\Sigma_{+1}|\} - \frac{1}{2}\{\boldsymbol{x}^T(\Sigma_{+1}^{-1} - \Sigma_{-1}^{-1})\boldsymbol{x} \\ &\quad + \boldsymbol{\mu}_{+1}^T \Sigma_{+1}^{-1} \boldsymbol{\mu}_{+1} - \boldsymbol{\mu}_{-1}^T \Sigma_{-1}^{-1} \boldsymbol{\mu}_{-1}\} + \boldsymbol{x}^T\{\Sigma_{+1}^{-1}\boldsymbol{\mu}_{+1} - \Sigma_{-1}^{-1}\boldsymbol{\mu}_{-1}\}. \end{aligned}$$

This leads to the quadratic discriminant function

$$\begin{aligned} F(\boldsymbol{x}) &= \frac{1}{2}\{\log|\Sigma_{-1}| - \log|\Sigma_{+1}|\} - \frac{1}{2}\{\boldsymbol{x}^T(\Sigma_{+1}^{-1} - \Sigma_{-1}^{-1})\boldsymbol{x} \\ &\quad + \boldsymbol{\mu}_{+1}^T\Sigma_{+1}^{-1}\boldsymbol{\mu}_{+1} - \boldsymbol{\mu}_{-1}^T\Sigma_{-1}^{-1}\boldsymbol{\mu}_{-1}\} + \boldsymbol{x}^T\{\Sigma_{+1}^{-1}\boldsymbol{\mu}_{+1} - \Sigma_{-1}^{-1}\boldsymbol{\mu}_{-1}\} + \log\frac{\pi_{+1}}{\pi_{-1}}. \end{aligned}$$

The $\mu_{\pm 1}$ and $\Sigma_{\pm 1}$ are estimated by the given data set as follows:

$$\hat{\mu}_{+1} = \frac{1}{n_1}\sum_{i=1}^{n} I(y_i = +1)\boldsymbol{x}_i \tag{123}$$

$$\hat{\mu}_{-1} = \frac{1}{n_2}\sum_{i=1}^{n} I(y_i = -1)\boldsymbol{x}_i \tag{124}$$

$$\hat{\Sigma}_{+1} = \frac{n_1}{n-2}\sum_{i=1}^{n} I(y_i = +1)(\boldsymbol{x}_i - \hat{\mu}_{+1})(\boldsymbol{x}_i - \hat{\mu}_{+1})^T \tag{125}$$

$$\hat{\Sigma}_{-1} = \frac{n_2}{n-2}\sum_{i=1}^{n} I(y_i = -1)(\boldsymbol{x}_i - \hat{\mu}_{-1})(\boldsymbol{x}_i - \hat{\mu}_{-1})^T, \tag{126}$$

where n_1 is the number of subjects having class label $+1$ and n_2 is the number of subjects having class label -1. A new feature vector $\boldsymbol{x}$ is classified $\hat{y} = +1$ if $F(\boldsymbol{x}) > 0$. QDA does not require the common covariance matrix among the both conditional distributions however generally QDA requires larger size of samples than LDA since the inverse of covariance matrix has to be calculated for each class. To accommodate with this situation, Regularized Linear Discriminant Analysis is introduced.

Regularized Linear Discriminant Analysis

Friedman (1989) proposed Regularized Discriminant Analysis (RDA) which is a compromise between FLDA and QDA. RDA shrinks covariance of QDA toward a

common covariance as in LDA. The covariance matrix for RDA has the form

$$\Sigma_y(\lambda) = \lambda\Sigma_y + (1-\lambda)\Sigma, \tag{127}$$

where $\hat{\Sigma}$ is the pooled covariance matrices as in FLDA. The parameter λ is normally decided by cross validation. We will review the details of cross validation later.

B Logistic Regression

The logistic regression model was considered to address the posterior probabilities of class labels via linear functions in $\boldsymbol{x}$. The model has the form

$$\log \frac{\mathrm{P}(Y=1|X=\boldsymbol{x})}{\mathrm{P}(Y=-1|X=\boldsymbol{x})} = \beta_0 + \boldsymbol{\beta}_1^T\boldsymbol{x}, \tag{128}$$

where β_0 is an intercept and $\boldsymbol{\beta}_1$ is a vector of coefficients. The form of logistic regression is log-odds. Take logarithm for both term then we can rewrite as follows:

$$\frac{\mathrm{P}(Y=1|X=\boldsymbol{x})}{\mathrm{P}(Y=-1|X=\boldsymbol{x})} = \exp(\beta_0 + \boldsymbol{\beta}_1^T\boldsymbol{x}). \tag{129}$$

The left part is called log odds.

Logistic regression models are usually fit by maximum likelihood. The log-likelihood for n observation is

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left\{ y_i \log p(\boldsymbol{x}_i|\boldsymbol{\beta}) + (1-y_i)\log(1-p(\boldsymbol{x}_i|\boldsymbol{\beta})) \right\}, \tag{130}$$

where

$$p(\boldsymbol{x}_i|\boldsymbol{\beta}_i) = \frac{e^{\boldsymbol{\beta}^T\boldsymbol{x}_i}}{1+e^{\boldsymbol{\beta}^T\boldsymbol{x}_i}}. \tag{131}$$

Applying this to Equation (130)

$$\begin{aligned} l(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \left\{ y_i \log e^{\boldsymbol{\beta}^T \boldsymbol{x}_i} - y_i \log(1 + e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}) + (1 - y_i) \log 1 - (1 - y_i) \log(1 + e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}) \right\} \\ &= \sum_{i=1}^{n} \left\{ y_i \log e^{\boldsymbol{\beta}^T \boldsymbol{x}_i} - y_i \log(1 + e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}) + \log 1 - y_i \log 1 - \log(1 + e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}) + y_i(1 + e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}) \right\} \\ &= \sum_{i=1}^{n} \{ y_i \boldsymbol{\beta}^T \boldsymbol{x}_i - \log(1 + e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}) \}. \end{aligned}$$

The results is set to zero to maximize the log-likelihood. Then

$$\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} \boldsymbol{x}_i (y_i - p(\boldsymbol{x} | \boldsymbol{\beta})) = 0. \tag{132}$$

Note that intersect is included in $\boldsymbol{\beta}$. To solve this, we use the Newton-Raphson algorithm which updates $\boldsymbol{\beta}$

$$\boldsymbol{\beta}^{\text{new}} = \boldsymbol{\beta}^{\text{old}} - \left(\frac{\partial^2 l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} \right)^{-1} \frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}, \tag{133}$$

where $\boldsymbol{\beta}^{\text{new}}$ is updated from $\boldsymbol{\beta}^{\text{old}}$. We need the second-derivative or Hessian matrix to solve this, which is

$$\begin{aligned} \frac{\partial^2 l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} &= \frac{\partial^2}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} \left\{ \sum_{i=1}^{n} \boldsymbol{x}_i \left(y_i - \frac{e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}}{1 + e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}} \right) \right\} && (134) \\ &= -\sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^T \frac{1}{(1 + e^{\boldsymbol{\beta}^T \boldsymbol{x}_i})} \frac{e^{\boldsymbol{\beta}^T \boldsymbol{x}_i}}{(1 + e^{\boldsymbol{\beta}^T \boldsymbol{x}_i})} && (135) \\ &= -\sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^T p(\boldsymbol{x}_i | \boldsymbol{\beta})(1 - p(\boldsymbol{x}_i | \boldsymbol{\beta})) && (136) \end{aligned}$$

Using matrices expression, the first derivative and the second derivative can be

written simply, therefore

$$\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \mathbf{X}^T(\mathbf{Y} - \mathbf{p}) \tag{137}$$

$$\frac{\partial^2 l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} = -\mathbf{X}^T\mathbf{W}\mathbf{X}, \tag{138}$$

where $\mathbf{X}$ the $N \times (p+1)$ matrix of x_i, $\mathbf{Y}$ denotes the vector of y_i values, $\mathbf{p}$ is the vector of fitted probabilities with ith element $p(\boldsymbol{x}|\beta^{\text{old}})$ and $\mathbf{W}$ is a $n \times n$ diagonal matrix of weights with ith diagonal element $p(\boldsymbol{x}|\beta^{\text{old}})(1-p(\boldsymbol{x}|\beta^{\text{old}}))$. Then the Newton step is written as follows:

$$\begin{aligned}
\boldsymbol{\beta}^{\text{new}} &= \boldsymbol{\beta}^{\text{old}} + (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y} - \mathbf{p}) && (139)\\
&= (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}(\mathbf{X}\boldsymbol{\beta}^{\text{old}} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p})) && (140)\\
&= (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{z}, && (141)
\end{aligned}$$

where

$$\mathbf{z} = \mathbf{X}\boldsymbol{\beta}^{\text{old}} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p})). \tag{142}$$

Equation (141) is referred to as Iteratively Reweighted Least Squares or IRLS (Green (1984)) since each iteration solves the weighted least squares problem:

$$\boldsymbol{\beta}^{\text{new}} = \arg\min_{\boldsymbol{\beta}} (\mathbf{z} - \mathbf{X}\boldsymbol{\beta})^T\mathbf{W}(\mathbf{z} - \mathbf{X}\boldsymbol{\beta}). \tag{143}$$

C Derivation of the α_t in the AdaBoost algorithm

AdaBoost minimizes loss function and α is lead by minimizing the loss function.

$$L_{\exp}(F_{t-1} + \alpha_t f_t) = \sum_{i=1}^{n} \exp\{-y_i(F_{t-1} + \alpha_t f_t(\boldsymbol{x}_i))\} \quad (144)$$

The first derivative of the loss function is used to calculate α_t then

$$\begin{aligned} L_{\exp}(F_{t-1} + \alpha_t f_t) &= \sum_{i=1}^{n} \exp\{-y_i(F_{t-1} + \alpha_t f_t(\boldsymbol{x}_i))\} \\ &= e^{-\alpha_t} \sum_{i=1}^{n} I(y_i = f_t(\boldsymbol{x}_i)) e^{-y_i F_{t-1}(\boldsymbol{x}_i)} + e^{\alpha_t} \sum_{i=1}^{n} I(y_i \neq f_t(\boldsymbol{x}_i)) e^{-y_i F_{t-1}(\boldsymbol{x}_i)} \\ &= e^{-\alpha_t}(1 - \varepsilon_t(f_t)) + e^{\alpha_t}\varepsilon_t(f_t) \end{aligned}$$

The optimal α_t is

$$\begin{aligned} \arg\min_{\alpha_t \in \mathbb{R}} L_{\exp}(F_{t-1} + \alpha_t f_t) &= \frac{\partial}{\partial \alpha_t} L_{\exp}(F_{t-1} + \alpha_t f_t) \\ &= -e^{-\alpha_t}(1 - \varepsilon_t(f_t)) + e^{\alpha_t}\varepsilon_t(f_t) \\ &= -e^{\alpha_t}\{e^{-2\alpha_t}(1 - \varepsilon_t(f_t)) - \varepsilon_t(f_t)\}, \end{aligned}$$

$\frac{\partial}{\partial \alpha_t} L_{\exp}(F_{t-1} + \alpha_t f_t) = 0$ therefore

$$\begin{aligned} \{e^{-2\alpha_t}(1 - \varepsilon_t(f_t)) - \varepsilon_t(f_t)\} &= 0 \\ e^{-2\alpha_t} &= \frac{\varepsilon_t(f_t)}{1 - \varepsilon_t(f_t)} \\ \alpha_t &= \frac{1}{2} \log \frac{1 - \varepsilon_t(f_t)}{\varepsilon_t(f_t)}. \end{aligned}$$

D Derivation of the η-Boost algorithm

We here show derivations of Equations (53) and (54) from the η-Boost algorithm. The η-Boost algorithm was derived by minimizing the loss function (51). Loss function (51) is rewritten as follows:

$$L_\eta(F+\alpha f) = \sum_{i=1}^{n}[(1-\eta)\exp\{-y_i(F(\boldsymbol{x}_i)+\alpha f)\} - \eta y_i(F(\boldsymbol{x}_i+\alpha f))]. \quad (145)$$

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

$$\frac{\partial}{\partial\alpha}L_\eta(F+\alpha f)|_{\alpha=0} = \sum_{i=1}^{n}[-y_i f(\boldsymbol{x}_i)\{(1-\eta)\exp(-y_i F(\boldsymbol{x}_i)) + \eta\}]. \quad (146)$$

We rewrite Equation (146) using the indicated function as follows:

$$\begin{aligned}\frac{\partial}{\partial\alpha}L_\eta(F+\alpha_t f_t)|_{\alpha=0} &= \sum_{i=1}^{n}[-I\{y_i = f(\boldsymbol{x}_i)\}w_t + I\{y_i \neq f(\boldsymbol{x}_i)\}w_t] \\ &= 2\sum_{i=1}^{n}[w_t I\{y_i \neq f(\boldsymbol{x}_i)\}] - \sum_{i=1}^{n} w_t, \end{aligned} \quad (147)$$

where $w_t = (1-\eta)\exp(-y_i F(\boldsymbol{x}_i)) + \eta$. From Equation (147), we find a value of f_t to minimize the weighted error rate. This is the derivative of Equation (53) Next α_t is calculated by minimizing η-Loss as follows:

$$\alpha_t = \arg\min_\alpha \frac{\partial}{\partial\alpha}L_\eta(F+\alpha f), \quad (148)$$

which implies that α_t is a solution of Equation

$$\frac{\partial}{\partial\alpha}L_\eta(F+\alpha f) = 0. \quad (149)$$

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

$$\begin{aligned} A &= \sum_{y_i f(\boldsymbol{x}_i)=1} e^{-y_i F(\boldsymbol{x}_i)} \\ B &= \sum_{y_i f(\boldsymbol{x}_i)=-1} e^{-y_i F(\boldsymbol{x}_i)} \\ C &= 2\left(\sum_{y_i f(\boldsymbol{x}_i)=-1} +1\right) - N. \end{aligned}$$

This is the derivation of Equations (53) and (54).

E Details of Equation (105)

We show the proof of Equation (105) here. For any positive integer r, below inequality can be applied,

$$\frac{r}{2} \leq r - \log_2 r.$$

To prove this, what we need to show is

$$r - 2\log_2 r \quad \geq \quad 0.$$

Set

$$h(r) \quad = \quad r - 2\log_2 r \tag{150}$$

$$h'(r) \quad = \quad 1 - \frac{2}{r\log 2} \tag{151}$$

$h'(r) = 0$, where $r = 2/\log 2$.

$$h\left(\frac{2}{\log 2}\right) \quad = \quad \frac{2}{\log 2} - 2\log_2 \frac{2}{\log 2} \tag{152}$$

$$= \quad 1.943 \tag{153}$$

Then Equation (105) is proved.

F Support Vector Machine

We briefly overview that how Support Vector Machine (SVM) maximizes the margin. SVM defines the margin as the distance from hyperplane to the closest data point. The hyperplane which can have the maximum margin is considered as the best hyperplane. The hyperplane is defined by using normal vector $\boldsymbol{w}$,

$$F(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + c, \tag{154}$$

where $\boldsymbol{w} \cdot \boldsymbol{x}$ represents inner product. The distance from any data point in the hyperplane to origin is $|c|/||\boldsymbol{w}||$. If data can be separated by linear function, we can

say

$$\boldsymbol{w} \cdot \boldsymbol{x}_i + c \geq +1 \Leftrightarrow y_i = +1 \tag{155}$$

$$\boldsymbol{w} \cdot \boldsymbol{x}_i + c \leq -1 \Leftrightarrow y_i = -1. \tag{156}$$

These equations can be summarized,

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + c) - 1 \geq 0. \tag{157}$$

Now consider a subject $\boldsymbol{x}_i$ which satisfies Equation (155). The data point resides on the hyperplane $\boldsymbol{w} \cdot \boldsymbol{x}_i + c = +1$ and the distance from origin is $|1 - c|/||\boldsymbol{w}||$. In the same manner, a data point which satisfies Equation (156) is on the hyperplane $\boldsymbol{w} \cdot \boldsymbol{x}_i + c = -1$. Then margin can be defined by $1/||\boldsymbol{w}||$. Maximizing margin is equivalent to minimize $||\boldsymbol{w}||$. This is treated as an optimization problem with a constraint such as

$$\min \quad ||\boldsymbol{w}||^2 \tag{158}$$

$$\text{subject to} \quad y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + c) \geq 1. \tag{159}$$

Lagrange multipliers are used to solve this problem. Then Equation (158) is re-expressed,

$$H_p(\boldsymbol{w}, c) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{n} \psi_i[y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + c) - 1], \tag{160}$$

where ψ is Lagrange multipliers $\boldsymbol{\phi} = (\psi_1, \ldots, \psi_n)$. To solve Equation (160), setting respective derivatives to zero,

$$\frac{\partial H_p}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{n} \psi_i y_i \boldsymbol{x}_i = 0 \tag{161}$$

$$\boldsymbol{w} = \sum_{i=1}^{n} \psi_i y_i \boldsymbol{x}_i \tag{162}$$

$$\frac{\partial H_p}{\partial c} = \sum_{i=1}^{n} \phi_i y_i = 0. \tag{163}$$

Apply Equation (162) and (163) into Equation (160),

$$\begin{aligned} H_p(\boldsymbol{w}, c) &= \frac{1}{2}\boldsymbol{w} \cdot \boldsymbol{w} - \sum_{i=1}^{n} \psi_i [y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + c) - 1] \\ &= \frac{1}{2}\boldsymbol{w} \cdot \boldsymbol{w} - \sum_{i=1}^{n} \psi_i y_i \boldsymbol{w} \cdot \boldsymbol{x}_i - \sum_{i=1}^{n} \psi_i y_i c + \sum_{i=1}^{n} \psi_i \\ &= \frac{1}{2}\boldsymbol{w} \cdot \boldsymbol{w} - \boldsymbol{w} \cdot \boldsymbol{w} + \sum_{i=1}^{n} \psi_i \\ &= \sum_{i=1}^{n} \psi_i - \frac{1}{2}\boldsymbol{w} \cdot \boldsymbol{w} \\ &= \sum_{i=1}^{n} \psi_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \psi_i \psi_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j. \end{aligned}$$

We obtain the Lagrangian dual objective function,

$$\max H_D = \sum_{i=1}^{n} \psi_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \psi_i \psi_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \tag{164}$$

$$\text{subject to } \sum_{i=1}^{n} \psi_i y_i = 0. \tag{165}$$

Solving the Lagrangian dual objective function, we obtain the optimum ψ_i^*. Then $\boldsymbol{w}^*$ is also solved as

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \psi_i^* y_i \boldsymbol{x}_i \tag{166}$$

From KKT condition, below equation needs to be satisfied,

$$\psi_i^* [y_i(\boldsymbol{w}^* \cdot \boldsymbol{x}_i + c) - 1] = 0. \tag{167}$$

Only subjects which satisfy $\psi_i \neq 0$ are called Support Vector which construct hyperplane for classification. c^* is solved using $\boldsymbol{w}^*$ then classifier is

$$\hat{y} = \operatorname{sgn}(\boldsymbol{w}^* \cdot \boldsymbol{x} + c^*) \tag{168}$$

$$= \left(\sum_{i \in \mathrm{SV}} \psi_i^* y_i \boldsymbol{x}_i \cdot \boldsymbol{x} + b^* \right) \tag{169}$$

References

Akaike, H. (1970), "Statistical predictor identification," *Annals of the Institute of Statistical Mathematics*, 22, 203–217.

Armstrong, S., Staunton, J., Silverman, L., Pieters, R., den Boer, M., Minden, M., Sallan, S., Lander, E., Golub, T., and Korsmeyer, S. (2001), "MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia," *Nature genetics*, 30, 41–47.

Bartlett, P. (1998), "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE transactions on Information Theory*, 44, 525.

Ben-Dor, A., Bruhn, L., Friedman, N., Nachman, I., Schummer, M., and Yakhini, Z. (2000), "Tissue classification with gene expression profiles," *Journal of Computational Biology*, 7, 559–583.

Bittner, M., Meltzer, P., Chen, Y., Jiang, Y., Seftor, E., Hendrix, M., Radmacher, M., Simon, R., Yakhini, Z., Ben-Dor, A., et al. (2000), "Molecular classification of cutaneous malignant melanoma by gene expression profiling," *Nature*, 406, 536–540.

Breiman, L. (1996), "Bagging predictors," *Machine learning*, 24, 123–140.

— (1998), "Arcing classifiers," *Annals of statistics*, 26, 801–824.

Bühlmann, P. and Yu, B. (2006), "Sparse Boosting," *Journal of Machine Learning Research*, 7, 1001–1024.

Chang, H., Nuyten, D., Sneddon, J., Hastie, T., Tibshirani, R., Sørlie, T., Dai, H., He, Y., Van't Veer, L., Bartelink, H., et al. (2005), "Robustness, scalability, and

integration of a wound-response gene expression signature in predicting breast cancer survival," *PNAS*, 102, 3738–3743.

Chang, H., Sneddon, J., Alizadeh, A., Sood, R., West, R., Montgomery, K., Chi, J., Van De Rijn, M., Botstein, D., and Brown, P. (2004), "Gene expression signature of fibroblast serum response predicts human cancer progression: similarities between tumors and wounds," *PLoS biology*, 2, 206–214.

Dietterich, T. (2000), "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine learning*, 40, 139–157.

Dietterich, T. and Bakiri, G. (1995), "Solving Multiclass Learning Problems via Error-Correcting Output Codes," *Journal of Artificial Intelligence Research*, 2, 263–286.

Drucker, H., Schapire, R., and Simard, P. (1993), "Boosting performance in neural networks," *International Journal of Pattern Recognition and Artificial Intelligence*, 7, 705–705.

Dudoit, S. and Fridlyand, J. (2002), "Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data," *Journal of the American Statistical Association*, 97, 77–87.

Eguchi, S. and Copas, J. (2001), "Recent developments in discriminant analysis from an information geometric point of view," *J. Korean Statist. Soc*, 30, 247–264.

Ein-Dor, L., Kela, I., Getz, G., Givol, D., and Domany, E. (2005), "Outcome signature genes in breast cancer: is there a unique set?" *Bioinformatics*, 21, 171–178.

Fan, C., Oh, D., Wessels, L., Weigelt, B., Nuyten, D., Nobel, A., van't Veer, L., and Perou, C. (2006), "Concordance among gene-expression-based predictors for breast cancer," *New England journal of medicine*, 355, 560–569.

Fisher, R. (1936), "The use of multiple measurements in taxonomic problems." *Ann of Eugenics*, 7, 179–188.

Freund, Y. and Schapire, R. (1997), "A desicion-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Science*, 55, 119–139.

Friedman, J. (1989), "Regularized discriminant analysis," *Journal of the American statistical association*, 84, 165–175.

— (2001), "Greedy function approximation: a gradient boosting machine," *Annals of Statistics*, 29, 1189–1232.

Friedman, J., Hastie, T., and Tibshirani, R. (2000), "Special invited paper. additive logistic regression: A statistical view of boosting," *Annals of statistics*, 28, 337–374.

Goetz, M., Suman, V., Ingle, J., Nibbe, A., Visscher, D., Reynolds, C., Lingle, W., Erlander, M., Ma, X., Sgroi, D., et al. (2006), "A two-gene expression ratio of homeobox 13 and interleukin-17B receptor for prediction of recurrence and survival in women receiving adjuvant tamoxifen," *Clinical Cancer Research*, 12, 2080–2087.

Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M., et al. (1999), "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring," *Science*, 286, 531.

Green, P. (1984), "Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives," *Journal of the Royal Statistical Society. Series B (Methodological)*, 46, 149–192.

Green, P. and Silverman, B. (1994), *Nonparametric regression and generalized linear models: a roughness penalty approach*, Chapman & Hall.

Grove, A. and Schuurmans, D. (1998), "Boosting in the limit: Maximizing the margin of learned ensembles," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 692–699.

Hastie, T. (2007), "Comment-Boosting Algorithms: Regularization, Prediction and Model Fitting," *Statistical Science*, 22, 513–515.

Hastie, T., Tibshirani, R., Friedman, J., and Franklin, J. (2005), *The elements of statistical learning: data mining, inference and prediction*, Springer.

Hu, Z., Fan, C., Oh, D., Marron, J., He, X., Qaqish, B., Livasy, C., Carey, L., Reynolds, E., Dressler, L., et al. (2006), "The molecular portraits of breast tumors are conserved across microarray platforms," *BMC genomics*, 7, 96–108.

Irizarry, R., Hobbs, B., Collin, F., Beazer-Barclay, Y., Antonellis, K., Scherf, U., and Speed, T. (2003), "Exploration, normalization, and summaries of high density oligonucleotide array probe level data," *Biostatistics*, 4, 249–264.

Jiang, W. (2004), "Process consistency for adaboost," *Annals of Statistics*, 32, 13–29.

Kawakita, M. and Eguchi, S. (2008), "Boosting method for local learning in statistical pattern recognition," *Neural computation*, 20, 2792–2838.

Kearns, M. and Valiant, L. (1989), "Cryptographic limitations on learning Boolean formulae and finite automata," *Machine Learning*, 29–49.

Naef, F., Hacker, C., Patil, N., and Magnasco, M. (2001), "Characterization of the expression ratio noise structure in high-density oligonucleotide arrays," *Genome biology*, 3.

Perou, C., Sørlie, T., Eisen, M., van de Rijn, M., Jeffrey, S., Rees, C., Pollack, J., Ross, D., Johnsen, H., Akslen, L., et al. (2000), "Molecular portraits of human breast tumours," *Nature*, 406, 747–752.

Pritchard, M. (2010), "Sparse Learner Boosting for gene expression data," *IPSJ Transactions on Bioinformatics*, 3, 54–61.

Roberts, C., Nelson, B., Marton, M., Stoughton, R., Meyer, M., Bennett, H., He, Y., Dai, H., Walker, W., Hughes, T., et al. (2000), "Signaling and circuitry of multiple MAPK pathways revealed by a matrix of global gene expression profiles," *Science*, 287, 873–880.

Rosset, S., Zhu, J., and Hastie, T. (2004), "Boosting as a regularized path to a maximum margin classifier," *Journal of Machine Learning Research*, 5, 941–973.

Saeys, Y., Inza, I., and Larrañaga, P. (2007), "A review of feature selection techniques in bioinformatics," *Bioinformatics*, 23, 2507–2517.

Schapire, R. (1990), "The strength of weak learnability," *Machine learning*, 5, 197–227.

Schapire, R., Freund, Y., Bartlett, P., and Lee, W. (1998), "Boosting the margin: A new explanation for the effectiveness of voting methods," *Annals of statistics*, 26, 1651–1686.

Schapire, R. and Singer, Y. (1999), "Improved boosting algorithms using confidence-rated predictions," *Machine learning*, 37, 297–336.

Schena, M., Shalon, D., Heller, R., Chai, A., Brown, P., and Davis, R. (1996), "Parallel human genome analysis: microarray-based expression monitoring of 1000 genes," *PNAS*, 93, 10614–10619.

Sorlie, T., Tibshirani, R., Parker, J., Hastie, T., Marron, J., Nobel, A., Deng, S., Johnsen, H., Pesich, R., Geisler, S., et al. (2003), "Repeated observation of breast tumor subtypes in independent gene expression data sets," *PNAS*, 100, 8418.

Takenouchi, T. and Eguchi, S. (2004), "Robustifying AdaBoost by Adding the Naive Error Rate," *Neural Computation*, 16, 767–787.

Takenouchi, T., Eguchi, S., Murata, N., and Kanamori, T. (2008), "Robust boosting algorithm against mislabeling in multiclass problems," *Neural computation*, 20, 1596–1630.

van de Vijver, M., He, Y., van't Veer, L., Dai, H., Hart, A., Voskuil, D., Schreiber, G., Peterse, J., Roberts, C., Marton, M., et al. (2002), "A gene-expression signature as a predictor of survival in breast cancer," *New England journal of medicine*, 347, 1999.

van't Veer, L., Dai, H., Van de Vijver, M., He, Y., Hart, A., Mao, M., Peterse, H., Van der Kooy, K., Marton, M., et al. (2002), "Gene expression profiling predicts clinical outcome of breast cancer," *Nature*, 415, 530–536.

Vapnik, V. (1998), *Statistical learning theory*, Wiley New York.

Vender, J., Adams, M., Myers, E., et al. (2001), "The sequence of the human genome," *Science*, 291, 1304–1351.

Zhang, T. and Yu, B. (2005), "Boosting with early stopping: Convergence and consistency," *Annals of Statistics*, 33, 1538–1579.