HHS Public Access

Author manuscript

Technometrics. Author manuscript; available in PMC 2020 January 01.

Published in final edited form as:

Technometrics. 2019 ; 61(2): 176–186. doi:10.1080/00401706.2018.1497544.

Convex Bidirectional Large Margin Classifiers

Zhengling Qi[1], **Yufeng Liu**[1,2,*]

[1]Department of Statistics and Operations Research, University of North Carolina, Chapel Hill

[2]Department of Genetics, Department of Biostatistics, Carolina Center for Genome Sciences, Lineberger Comprehensive Cancer Center, University of North Carolina, Chapel Hill

Abstract

Classification problems are commonly seen in practice. In this paper, we aim to develop classifiers that can enjoy great interpretability as linear classifiers, and at the same time have model flexibility as nonlinear classifiers. We propose convex bidirectional large margin classifiers to fill the gap between linear and general nonlinear classifiers for high dimensional data. Our method provides a new data visualization tool for classification of high dimensional data. The obtained bilinear projection structure makes the proposed classifier very interpretable. Additional shrinkage to approximate variable selection is also considered. Through analysis of simulated and real data in high dimensional settings, our method is shown to have superior prediction performance and interpretability when there are potential subpopulations in the data. The computer code of the proposed method is available as supplemental materials.

Keywords

Bilinear Classification; Interpretability; Variable Selection; Visualization

1 Introduction

Classification is a typical supervised learning problem in machine learning and statistics. For classification, one needs to identify a decision rule based on a training dataset which consists of input variables and their corresponding class labels. Once the rule is obtained, a classification rule can predict the label for a new instance using information of the input variables. Many classification algorithm or methods can be viewed as large margin classifiers (Hastie et al., 2001). This ranges from classical ones such as perceptron algorithm and logistic regression, to modern machine learning techniques such as Boosting (Freund and Schapire (1997), Freund et al. (1999); Schapire et al. (1998)) and the Support Vector Machine (SVM; Vapnik (1998)). For an overview of large margin classifiers, one can refer to Bartlett et al. (2003); Liu et al. (2011).

* yfliu@email.unc.edu.

7 Supplemental Materials

The computer code for the proposed methods is available with this article as Supplemental Materials.

The development of classification methods mainly focuses on two aspects: One is to find a classification rule which can correctly identify the true class, that is a rule with high classification accuracy; The other is to get an interpretable or meaningful model, for example, whether we can recognize important features and understand their contributions to the classifier.

Using a linear combination of the variables, linear classifiers are one of the most widely used classification tools. Training and testing procedures for linear classifiers are relatively efficient compared with nonlinear ones, especially in high dimensional spaces. Moreover, due to the simple linear form, the corresponding interpretation can be straightforward. Despite its simplicity, however, a linear classifier may fail to handle classification problems with nonlinear boundaries and thus the prediction performance can be suboptimal.

To overcome linearly nonseparable data in the input feature space and get more accurate results, linear classifiers can be extended to nonlinear ones by mapping variables into higher dimensional feature spaces. A well known technique is the kernel trick used in the SVM to capture the nonlinear patterns of the data. In general, nonlinear classifiers are more flexible than linear ones and it can achieve better prediction performance when the underlying true classification boundary is nonlinear. However, compared with linear ones, nonlinear classifiers in general do not provide intuitive interpretation about the difference between classes based on the input variables. For example, it can be hard to explain the effect of each input variable for a nonlinear classifier. In addition, its training and testing procedures may not be as efficient as linear ones, especially for high dimensional problems, and the model is more likely to overfit due to the use of multiple tuning parameters.

Our proposed work is motivated by the limitations of both linear and nonlinear classifiers. Our goal is to design a classifier with interpretability similar to linear ones, but with much more flexibility. In the literature, there exists some work on simplifying nonlinear classifiers such as Bach (2009); Lin et al. (2006); Bertsimas et al. (2012). However, most of them focus on variable selection on nonlinear classifiers. Huang et al. (2012) proposed to use multiple linear functions to achieve the goal, but the corresponding computation is nonconvex. In this paper, we propose the Convex Bilinear large margin Classifier (CBC), which maintains the great interpretability of linear classifiers and also keeps accurate prediction performance especially when sub-group structures exist in the data.

An important characteristic of our proposed CBC method is that it can be computed through convex optimization, and meanwhile automatically provide a new way to visualize high dimensional data. The CBC method can construct an effective low dimensional subspace for classification of data with subpopulation structures via bilinear projection. Unlike unsupervised learning, the CBC makes use of both input variables and label information to construct a low dimensional space to retain most informative structure in the data for classification. Comparison with the principal component analysis (PCA) shows that our CBC method can be also used as an useful visualization tool for high dimensional classification problems.

In order to further enhance the interpretability of our approach, we implement an additional one-step shrinkage for our CBC method to approximate variable selection. In particular, we propose a weighted CBC method by using the weighted l_2 penalty to shrink variables and achieve approximate sparsity with efficient computation for the projection. In this way, we can not only identify a low dimensional subspace for high dimensional data, but also discriminate the most important features that contribute to the projection. This further strengthens the interpretability of our proposed methods.

The rest of this article is organized as follows. In Section 2, we introduce our CBC method and its related properties. In Section 3, we discuss the variable selection procedure for the CBC method. In Section 4, the computational algorithm is provided. In Section 5, we compare our method with related work via simulation studies and a real data application. We conclude this paper with some discussion in Section 6.

2 Bilinear Large Margin Classification Framework

In supervised learning, we are given a set of training data $\{(\boldsymbol{x}_i, y_i) : i = 1, 2, \cdots, n\}$, where $\boldsymbol{x}_i \in \mathcal{R}^p$ represents a p dimensional input vector including the intercept and y_i is an output label. We consider the standard binary classification problem with $y_i \in \{1,-1\}$. One important goal of classification is to find a classifier from the training data, so that one can predict the class label y for any given new instance $\boldsymbol{x}$.

In this article, we mainly focus on large-margin classifiers. Specifically, a margin-based classifier tries to obtain a function $f(\boldsymbol{x})$, mapping from $\mathcal{R}^p$ to $\mathcal{R}$, and use sign($f(\boldsymbol{x})$) as the classification rule. According to the classification rule, $yf(\boldsymbol{x})$ decides the classification result on the point $(\boldsymbol{x}, y)$. Correct classification happens if and only if $yf(\boldsymbol{x}) > 0$. The quantity $yf(\boldsymbol{x})$ is usually referred as the functional margin and used for many large-margin classification techniques.

It is well known that many large-margin classifiers can be fit into the regularization framework of loss + penalty. The loss term controls the model fitting of the data and the regularization term is used to prevent overfitting. In general, the regularization formulation of binary large-margin classification can be expressed as follows:

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L\left(y_i f\left(\boldsymbol{x}_i\right)\right) + \tau J(f), \quad (1)$$

where $\mathcal{F}$ refers to some function class, L is a loss function on the margin $yf(\boldsymbol{x})$, $J(f)$ is the regularization term and τ is a non-negative tuning parameter to balance the two terms. A natural choice of the loss function in (1) is the 0–1 loss with $L = \mathbb{1}(yf(\boldsymbol{x}) \leq 0)$, which assigns the loss of 1 for misclassification and 0 otherwise. However, since the 0–1 loss is a non-convex and non-smooth function that is difficult to optimize, many convex surrogate loss functions have been proposed. For example, the SVM uses the hinge loss $L(yf(\boldsymbol{x})) = \max(0, 1 - yf(\boldsymbol{x}))$. Other popular loss functions include the binomial deviance loss for penalized

logistics regression (Lin et al. (2000)), the exponential loss for AdaBoost (Freund and Schapire (1997)), and the Huber loss for robust classifiers (Rosset and Zhu (2007)). Figure 1 compares different loss functions on $yf(\boldsymbol{x})$. Recently, Liu et al. (2011) proposed a family of large margin classifiers to unify many large-margin machines.

Among various large margin classifiers, according to the functional class $\mathcal{F}$ in (1), one can divide them into two major groups: linear or nonlinear classifiers. The comparison between linear and nonlinear classifiers in the introduction motivates us to propose a new method to combine the strengths of both methods. One possible approach is to use two linear hyperplanes to separate two classes. In this situation, the functional margin becomes $yf_1(\boldsymbol{x})f_2(\boldsymbol{x}) = y(\boldsymbol{x}^T\boldsymbol{w}_1)(\boldsymbol{x}^T\boldsymbol{w}_2)$ and can be further expressed as $y\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$ by replacing $\boldsymbol{w}_1\boldsymbol{w}_2^T$ with a $p \times p$ matrix $\boldsymbol{A}$. In order to get two linear hyperplanes, the rank of $\boldsymbol{A}$ should be equal to 1, which inspires us to propose the following bilinear optimization problem:

$$\min_{\boldsymbol{A} \in \mathcal{R}^{p \times p}} \quad \frac{1}{n}\sum_{i=1}^{n} L\left(y_i \boldsymbol{x}_i^T \boldsymbol{A} \boldsymbol{x}_i\right), \quad (2)$$

$$\text{subject to} \quad \text{rank}(\boldsymbol{A}) = 1.$$

Note that $\boldsymbol{x}_i^T\boldsymbol{A}\boldsymbol{x}_i = \left\langle \boldsymbol{A}, \boldsymbol{x}_i\boldsymbol{x}_i^T \right\rangle$, which is the inner product between two matrices. The (k, m)-th entry of matrix $\boldsymbol{A}$, denoted as A_{km}, corresponds to the contribution of interaction between the k-th and m-th variables in the classification rule. Since $\boldsymbol{x}_i$ includes the constant term, $\boldsymbol{x}_i^T\boldsymbol{A}\boldsymbol{x}_i$ covers both linear and quadratic terms. Our matrix representation can be viewed as a data-driven classification technique with rank-one approximation to the coefficient matrix $\boldsymbol{A}$. Under the assumption that the coefficient matrix is of a low rank, we can further extend the constraint in (2) to rank($\boldsymbol{A}$) = r by Solving

$$\min_{\boldsymbol{U}_r, \boldsymbol{V}_r \in \mathcal{R}^{p \times r}} \quad \frac{1}{n}\sum_{i=1}^{n} L_i\left(y_i \boldsymbol{x}_i^T \boldsymbol{A} \boldsymbol{x}_i\right), \quad (3)$$

$$\text{subject to} \quad \boldsymbol{A} = \boldsymbol{U}_r\boldsymbol{V}_r^T,$$

where $\boldsymbol{U}_r$ and $\boldsymbol{V}_r$ form a matrix factorization for $\boldsymbol{A}$ and their own columns are orthogonal. Note that r needs to be decided in advance or be viewed as a tuning parameter similar to principal component analysis or k-means clustering.

For illustration, we consider some toy examples in Figure 2. We consider $\boldsymbol{A}$ as a size 5 by 5 matrix. On the left panel, we let $\boldsymbol{A} = \boldsymbol{u}_1\boldsymbol{v}_1^T$ of rank 1, where $\boldsymbol{u}_1$ is a vector of length 5 with all entries 0 except the first entry being 1 and the entries for v_1 are 0 except the last entry being 1. On the right panel, we consider a rank-2 matrix $\boldsymbol{A}$, where $\boldsymbol{A}$ can be expressed as

$\boldsymbol{u}_1\boldsymbol{v}_1^T + \boldsymbol{u}_2^T\boldsymbol{v}_2^T$ with $\boldsymbol{u}_1\boldsymbol{v}_1$ as the left panel, and $\boldsymbol{u}_2$ is a vector of length 5 with all entries 0 except the second entry being 1 and the entries for $\boldsymbol{v}_2$ are 0 except the fourth entry being 1.

To further illustrate the idea, we consider the classifier in the form of sign($(x^T w_1)(x^T w_2)$) in (2), where $w_1 = (0,1,0)$ and $w_2 = (0,0,1)$. The corresponding classification boundary is shown in Figure 3. Based on this plot, we can see that a decision boundary using two cross lines identifies the class structure. Note that the decision function in the form of $(x^T w_1)(x^T w_2)$ can be quite general. It covers linear classifiers if $x^T w_2$ is estimated to be a constant. Moreover, if each class contains subpopulations such as a mixture of multiple Gaussian components, such a decision function can capture the classification structure by constructing two hyperplanes. Classification problems with within class subpopulations can be commonly seen in practice. For example, in cancer gene expression study of classifying cancer versus normal samples, the cancer class may have multiple subtypes due to the disease heterogeneity (Tibshirani et al. (2002)).

Although problems (2) and (3) have clear motivations, the corresponding optimization problems are non-covex due to the rank constraint. For example, for any matrix A_1 and A_2 with rank r, the rank of $\lambda A_1 + (1 - \lambda)A_2$ is not necessarily still r, where $\lambda \in (0, 1)$. As a result, the rank constraint is not a convex set and thus the optimization problem can be difficult to solve. In Section 2.1, we propose a convex surrogate classifier to solve such problems.

2.1 Convex Bilinear Large Margin Classifier Framework

Before we introduce our CBC method, we first introduce notations to be used for the rest of this article. For a vector $\boldsymbol{w} \in \mathscr{R}^p$, the Euclidean norm is denoted as $\|\boldsymbol{w}\| = \sqrt{\sum_{i=1}^p \boldsymbol{w}_i^2}$. For a matrix $\boldsymbol{A} \in \mathscr{R}^{m \times n}$, we denote the Frobenius norm as $\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n A_{ij}^2}$, where A_{ij} denotes the (i, j)-th entry of $\boldsymbol{A}$. If the rank of $\boldsymbol{A}$ is r, then the condensed singular value decomposition (SVD) of $\boldsymbol{A}$ can be expressed as $\boldsymbol{A} = \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r^T$, where $\boldsymbol{U}_r \in \mathscr{R}^{m \times r}$ and $\boldsymbol{V}_r \in \mathscr{R}^{n \times r}$ satisfy $\boldsymbol{U}_r^T \boldsymbol{U}_r = \boldsymbol{I}_r$ and $\boldsymbol{V}_r^T \boldsymbol{V}_r = \boldsymbol{I}_r$ respectively, and $\boldsymbol{\Sigma}_r = \text{diag}(\sigma_1(\boldsymbol{A}), \cdots, \sigma_r(\boldsymbol{A}))$ with $\sigma_1(\boldsymbol{A}) \geq \cdots \geq \sigma_r(\boldsymbol{A}) > 0$. In addition, $\|\boldsymbol{A}\|_* = \sum_{i=1}^r \sigma_i(\boldsymbol{A})$ denotes the nuclear norm of $\boldsymbol{A}$.

Since (2) involves a non-convex optimization problem, it may be hard to solve when the dimension is large. Convex relaxation of the rank constraint using the nuclear norm has been shown to be successful in solving rank-constrained problems with some theoretical guarantees (Candès and Tao (2010)). Minimizing the nuclear norm can help to reduce the rank stably and perform shrinkage at the same time. This encourages us to reformulate (2) into the following convex minimization problem

$$\min_{\boldsymbol{A} \in \mathscr{R}^{p \times p}} \quad F(\boldsymbol{A}) = \frac{1}{n} \sum_{i=1}^n L\left(y_i, \boldsymbol{x}_i^T \boldsymbol{A} \boldsymbol{x}_i\right) + \tau \|\boldsymbol{A}\|_*. \quad (4)$$

Note that for a given τ, solving (4) does not necessarily give us an exactly rank r solution. One intuitive approach is to solve (4) using a series of τ to get a solution with rank r. However, replacing the rank constraint by the nuclear norm may lead to infinite solutions due to the special form $\boldsymbol{x}_i^T \boldsymbol{A} \boldsymbol{x}_i$ by the following theorem.

Theorem 1. *For any matrix* $\boldsymbol{A} \in \mathcal{R}^{p \times p}$ *and any* $\tau \geq 0$, *there exists a symmetric matrix* $\boldsymbol{Z}$ *such that* $F(\boldsymbol{Z}) \leq F(\boldsymbol{A})$. *Furthermore, if an optimal solution* $\boldsymbol{A}^*$ *to* (4) *is not symmetric, any convex combination of* $\boldsymbol{A}^*$ *and* $\boldsymbol{A}^{*T}$ *is also an optimal solution.*

From Theorem 1, we know that there always exists a symmetric matrix $\boldsymbol{A}^*$ optimizing problem (4). Since both $\boldsymbol{x}_i^T \boldsymbol{A} \boldsymbol{x}_i$ and $\|\boldsymbol{A}\|_*$ are invariant to the transpose operator, we need to add a symmetric constraint on matrix $\boldsymbol{A}$ in the optimization problem (4) as follows to make it meaningful:

$$\min_{\boldsymbol{A} \in \boldsymbol{S}_p} \quad G(\boldsymbol{A}) = \sum_{i=1}^{n} L\left(y_i, \boldsymbol{x}_i^T \boldsymbol{A} \boldsymbol{x}_i\right) + \tau \|\boldsymbol{A}\|_*, \quad (5)$$

where S_p denotes the class of $p \times p$ symmetric matrices. For problem (5) we can still control τ to get a solution with rank r. However, it may lead to unsatisfactory classifiers with useless prediction. For example, if we search a series of τ in problem (5) to get the solution with rank $r = 1$, the optimal $\boldsymbol{A}^*$ can always be degenerated as $\boldsymbol{A}^* = \lambda_1 \|\boldsymbol{u}\|_2^2$ by spectral decomposition. In this situation, the classification rule will become $\text{sign}\left(\lambda_1 \|\boldsymbol{u}\|_2^2 \|\boldsymbol{x}\|_2^2\right) = \text{sign}\left(\lambda_1\right)$ for any given new instance $\boldsymbol{x}$, which means that the corresponding classifier will always assign a new instance to one class. In order to address this difficulty incurred by the symmetric constraint especially when our purpose is to get rank $r = 1$, Theorem 1 inspires us to search the solution with rank higher than r first and then find the equivalent best rank r solution. Here the equivalent best rank r solution means achieving the same objective value in (5) but without being symmetric. For example, if we want to get a solution with rank $r = 1$, we may search some τ's to get the solution with $r = 2$ first and then find the equivalent best rank 1 matrix.

Suppose $\boldsymbol{A}(\tau)$ be an optimal solution to (5) with the tuning parameter τ. We assume that rank($\boldsymbol{A}(\tau)$) is monotonely non-increasing in τ and there exists $0 = \tau_0 < \tau_1 < \cdots < \tau_p < \tau_{p+1} = \infty$ such that for $\tau \in [\tau_k, \tau_{k+1})$, the corresponding rank of $\boldsymbol{A}(\tau)$ is k. This assumption is reasonable because as τ increases, a large penalty on the nuclear norm of A is imposed, and consequently leads to smaller ranks in general. We have the following theorem.

Theorem 2. *For any* $\tau \in [\tau_2, \tau_3)$ *with* $\boldsymbol{A}(\tau)$ *as the solution for* (5), *there always exists a* $\boldsymbol{A}^*(\tau)$ *with rank* $r = 1$ *such that* $G(\boldsymbol{A}^*(\tau)) = G(\boldsymbol{A}(\tau))$. *Furthermore,* $\boldsymbol{A}^*(\tau)$ *is also a global minimizer to*

$$\min_{\boldsymbol{A} \in \mathcal{R}^{p \times p}} \quad \sum_{i=1}^{n} L_i\left(y_i \boldsymbol{x}_i^T \boldsymbol{A} \boldsymbol{x}_i\right) + \tau \|\boldsymbol{A}\|_*, \quad (6)$$
$$subject\ to \quad rank(\boldsymbol{A}) = 1,$$

where $\tau \in [\tau_2, \tau_3)$. *Let two eigenvalues of* $\mathbf{A}(\tau)$ *be* λ_1, λ_2 *with* $\lambda_1 \geq \lambda_2$ *and their corresponding two eigenvectors be* $\boldsymbol{q}_1, \boldsymbol{q}_2$. *Then we can find* $\boldsymbol{A}^*(\tau) = \boldsymbol{U}_1 \boldsymbol{V}_1^T$ *explicitly with*

$$\begin{aligned} \boldsymbol{U}_1 &= \sqrt{|\lambda_1|}\boldsymbol{q}_1 + \sqrt{|\lambda_2|}\boldsymbol{q}_2, \quad (7) \\ \boldsymbol{V}_1 &= \sqrt{|\lambda_1|}\boldsymbol{q}_1 - \sqrt{|\lambda_2|}\boldsymbol{q}_2. \end{aligned}$$

Remark 1. *Note that the rank* 1 *solution* $\mathbf{A}^*(\tau)$ *with* $\mathbf{U}_1, \mathbf{V}_1$ *can be interpreted as two linear hyperplanes to separate the input space. Two orthogonal eigenvectors* $\boldsymbol{q}_1, \boldsymbol{q}_2$ *could be used for supervised dimension reduction by projecting data into the corresponding two-dimensional orthogonal space. This projection provides us a new way of data visualization for high dimensional data and can possibly help to detect subcluster structure within each class as we will demonstrate in Section 5. Figure 4 gives us one toy example. The dataset has four clusters and two for each class respectively. Their class label is decided by the first two features with additional* 48 *dimension noisy features, as seen in the left panel of Figure 4. On the right panel, features are projected onto orthogonal directions* $\boldsymbol{q}_1, \boldsymbol{q}_2$. *Clearly this new data visualization approach captures the structure of two clusters in each class and their relative location by making use of both input features and label information.*

Remark 2. *For a higher rank-constrained problem, we can use a similar strategy as in Theorem 2. However, the corresponding computation may be more intensive since we do not have an explicit form as Equation* (7) *in Theorem 2. Although higher rank solutions may be able to identify more complicated classification structures, we may loss the interpretation of the corresponding classifiers.*

For simplicity, we mainly consider how to get the rank $r = 1$ solution based on problem (5), although the higher rank setting can be extended directly. Note that the loss function L on the margin $yf(x)$ can be general convex loss functions for theorems discussed so far. For an illustration, we use the smooth hinge loss function proposed by Rennie and Srebro (2005) that shares some similarity with the standard hinge loss and can be solved by gradient-decent-type algorithms with $L(z) = 1/2 - z$ if $z \leq 0$, $(1 - z)^2/2$, if $0 < z < 1$, and 0 if $z \geq 1$.

In Section 2.2, we discuss the close connection between (6) and the classification problems in Huang et al. (2012). This connection further motivates the idea of our proposed methods.

2.2 Properties and Related Literature

In Theorem 2, we conclude that $\boldsymbol{U}_1$ and $\boldsymbol{V}_1$ form a global solution to (6) for $\tau \in [\tau_2, \tau_3)$. Note that the optimization problem (6) controls both the nuclear norm and rank

simultaneously, which is closely related to the standard classification problem as shown in the following theorem.

Theorem 3. *The solution to* (6) *with* $\boldsymbol{A} = \boldsymbol{U}_1 \boldsymbol{V}_1^T$ *lies in the solution path of the following problem:*

$$\min_{\boldsymbol{U}_1, \boldsymbol{V}_1 \in \mathscr{R}^{p \times 1}} \quad \sum_{i=1}^{n} L\left(y_i \boldsymbol{x}_i^T \boldsymbol{U}_1 \boldsymbol{V}_1^T \boldsymbol{x}_i\right) + \frac{\tau}{2}\left(\|\boldsymbol{U}_1\|_F^2 + \|\boldsymbol{V}_1\|_F^2\right). \quad (8)$$

We observe that the problem is exactly to find two linear hyperplanes similar to the usual classification framework. In this setting, (8) is the same to the bidirectional discrimination proposed by Huang et al. (2012), where in their paper they use the standard hinge loss and didn't include the constant term inside the features $\boldsymbol{x}$ to be regularized. They used the block-coordinate decent algorithm to solve (8) by iteratively fixing $\boldsymbol{U}_1$ or $\boldsymbol{V}_1$ to solve the other via standard quadratic programming. However, (8) is a non-convex problem with possibly undesirable local minimums. In contrast, our method for rank $r = 1$ solves a series of convex optimization with guarantee of global solutions for different tuning parameters τ within a specific range decided by problem (5). Thus our CBC method has the potential of more accurate performance as we will demonstrate through our numerical examples.

In Section 3, we propose a weighted CBC method to approximate variable selection. This extension can be useful when there are a lot of noise variables, especially under high dimensional settings. In particular, we consider to modify the optimization problem (8) by adding a weighed L_2 penalty and transform it back into nuclear norm minimization problems. This can further improve the interpretability of our method by identifying important variables.

3 Weighted CBC in Bilinear Large Margin Classifiers

For high dimensional classification problems, there is a high risk of model overfitting. Therefore, it is desirable to perform further shrinkage in order to improve the prediction accuracy and model interpretation.

From the statistical function estimation perspective as we discussed before, large margin classifiers, for example the SVM, can be viewed in the regularization form of loss + L_2 penalty (Hastie et al., 2001). It is well known that applying the L_2 regularization has the effect of shrinking the coefficients toward zero, while reducing variances by sacrificing unbiasedness. The same L_2 regularization is used in ridge regression for regression problems. However, the L_2 penalty does not produce exactly 0 solutions for coefficients and thus it may be difficult to interpret the model. Using a similar idea from LASSO (Tibshirani, 1996) in linear regression, the L_1-SVM was proposed to both increase the prediction accuracy and automatically shrink some coefficients to exact 0 (Bradley and Mangasarian, 1998). In addition, efficient algorithm has also been proposed to compute the whole solution path (Zhu et al. (2004)). However, for highly correlated features, L_2 regularization may yield

better prediction power than the L_1 because the L_1 regularization tends to select only a few among highly correlated variables and remove the rest. In addition, the total number of selected features is bounded by the sample size.

In our bidirectional large margin framework, in order to implement variable shrinkage, we start from problem (8). In order to get sparse solutions, we could apply similar ideas as in the L_1-SVM (Zhu et al. (2004)). However, introducing the L_1 regularization may make the optimization problem challenging. In particular, unlike the transformation of convex optimization when the L_2 penalty is used, we do not have such convex transformation anymore when the L_1 penalty is used. As a result, it can be more difficult to solve. Therefore, in order to achieve further shrinkage and avoid nonconvexity, we propose the weighted CBC to approximate variable selection by using the weighted L_2 penalty as follows:

$$\min_{\boldsymbol{U}_1, \boldsymbol{V}_1 \in \mathscr{R}^{p\times 1}} \quad \sum_{i=1}^{n} L\left(y_i, \left(\boldsymbol{x}_i^T \boldsymbol{U}_1\right)\left(\boldsymbol{V}_1^T \boldsymbol{x}_i\right)\right) + \frac{\tau}{2}\left(\left\|\frac{\boldsymbol{U}_1}{\boldsymbol{\alpha}_1}\right\|^2 + \left\|\frac{\boldsymbol{V}_1}{\boldsymbol{\alpha}_2}\right\|^2\right), \quad (9)$$

where the division is element-wise and $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$ are data driven weights. We can use the solution $\boldsymbol{U}_1$, $\boldsymbol{V}_1$ in (7) as weights. Although our weighted L_2 penalty cannot produce exact sparse solutions, it can be viewed as an approximation of the L_0 regularization, where $L_0\left(\boldsymbol{U}_1\right) = \sum_{i=1}^{p} \mathbb{I}\left(\left|\boldsymbol{U}_{1i}\right| > 0\right)$. The L_0 regularization corresponds to the best subset variable selection technique. For our weighted penalty, if one of $\boldsymbol{\alpha}_{1i}$ is large, the corresponding U_{1i} will have a small penalty for being non-zero. On the other hand, a small $\boldsymbol{\alpha}_{1i}$ will yield a large penalty on U_{1i} not being zero, shrinking it towards 0. We can iteratively optimize (9) to get a nearly sparse solution. In our numerical study, we find that our one-step weighted CBC performs well.

Note that optimization problem (9) can be rewritten as

$$\min_{\boldsymbol{U}_1, \boldsymbol{V}_1 \in \mathscr{R}^{p\times 1}} \quad \sum_{i=1}^{n} L\left(y_i, \left(\boldsymbol{x}_i \bullet \boldsymbol{\alpha}_1\right)^T \boldsymbol{U}_1 \boldsymbol{V}_1^T \left(\boldsymbol{x}_i \bullet \boldsymbol{\alpha}_2\right)\right) + \frac{\tau}{2}\left(\|\boldsymbol{U}_1\|^2 + \|\boldsymbol{V}_1\|^2\right), \quad (10)$$

which is similar to problem (8). Basically, (10) is the weighted solution of (8). But the key difference between (10) and (8) lies in that, in (10), we consider the modified or weighted covariates $x_i \cdot \boldsymbol{\alpha}_1$ and $x_i \cdot \boldsymbol{\alpha}_2$. This motivates us to develop a different approach to handle it, using the following corollary.

Corollary 3.1. *The following nuclear norm minimization problem is equivalent to* (9)*:*

$$\min_{\boldsymbol{A} \in \mathscr{R}^{p \times p}} \quad \sum_{i=1}^{n} L\left(y_i, (\boldsymbol{x}_i \bullet \boldsymbol{\alpha}_1)^T \boldsymbol{A}(\boldsymbol{x}_i \bullet \boldsymbol{\alpha}_2)\right) + \tau_2 \|\boldsymbol{A}\|_* \quad (11)$$
$$\text{subject to} \quad rank(\boldsymbol{A}) = 1,$$

where • *denotes the element-wise product.*

In order to solve the problem (11), we no longer need to search a series of τ to get the rank 2 solution first and transform the rank 2 solution into rank 1 solution. This is because $(\boldsymbol{x}_i \bullet \boldsymbol{\alpha}_2)$ no longer equals to $(\boldsymbol{x}_i \bullet \boldsymbol{\alpha}_1)$ and Theorem 1 does not apply anymore. In this situation, by controlling τ_2 for rank 1 solutions, we can solve it directly.

To summarize the relationship among different formulations, problem (9) is a weighted version of (8). Since (10) is equivalent to (9), (10) can be viewed as a weighted version of (8) as well. The key difference between (10) and (8) lies in that we consider modified $(\boldsymbol{x}_i \bullet \boldsymbol{\alpha}_1)$ and $(\boldsymbol{x}_i \bullet \boldsymbol{\alpha}_2)$ in (10).

4 Computational Algorithm

In this section, we present our algorithm for solving the nuclear norm minimization problem (5). To that end, we first introduce an important lemma.

Lemma 1. *Suppose we have* $\boldsymbol{X} \in \mathscr{R}^{p \times p}$*, then the unique solution to the optimization problem:*

$$\min_{\boldsymbol{Y} \in \mathscr{R}^{p \times p}} \quad \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \tau \|\boldsymbol{Y}\|_*$$

is $\boldsymbol{Y} = S_\tau(\boldsymbol{X})$*, where* $S_\tau(\boldsymbol{X}) = \boldsymbol{U}\boldsymbol{\Sigma}_\tau \boldsymbol{V}^T$*,* $\boldsymbol{U}$ *and* $\boldsymbol{V}$ *are left and right singular vectors of* $\boldsymbol{X}$*, and* $\boldsymbol{\Sigma}_\tau = diag((\sigma_1(\boldsymbol{X}) - \tau)_+, \cdots, (\sigma_p(\boldsymbol{X}) - \tau)_+)$*, where the function* $m_+ = \max(m, 0)$.

Algorithm 1

basis estimation algorithm

1: **procedure** BASIS ESTIMATION ALGORITHM
2: Given $\tau_1 > \tau_2 > \cdots > \tau_k$ and Initialize $\boldsymbol{A}^{\text{last}}$ by random
3: $\boldsymbol{A}^{\text{last}} = \frac{\boldsymbol{A}^{\text{last}} + \boldsymbol{A}^{\text{last}T}}{2}$
4: **while** $\text{rank}\left(\boldsymbol{A}_{\tau_k}\right) \leq 2$ **do**
5: $\delta = 1$
6: **while** $\delta > \epsilon$ **do**

7: $\left.\frac{\partial L}{\partial \boldsymbol{A}}\right|_{\boldsymbol{A}^{\text{last}}} = \sum_{i=1}^{n} L'\left(y_i, \boldsymbol{x}_i^T \boldsymbol{A}^{\text{last}} \boldsymbol{x}_i\right) y_i \boldsymbol{x}_i \boldsymbol{x}_i^T$

8: $\boldsymbol{A}^{\text{next}} = S_{t\tau_k}\left(\boldsymbol{A}^{\text{last}} - t\left.\frac{\partial L}{\partial \boldsymbol{A}}\right|_{\boldsymbol{A}^{\text{last}}}\right)$

9: $\delta = \frac{\|\boldsymbol{A}^{\text{last}} - \boldsymbol{A}^{\text{next}}\|_F}{\max(1, \|\boldsymbol{A}^{\text{last}}\|_F)}$

10: $\boldsymbol{A}^{\text{last}} = \boldsymbol{A}^{\text{next}}$

11: **end while**

12: $\boldsymbol{A}_{\tau_k} = \boldsymbol{A}^{\text{last}}$

13: **end while**

14: **end procedure**

This operator $S_{\boldsymbol{\tau}}(\boldsymbol{X})$ is known as soft-thresholding and proofs can be found in Cai et al. (2009) and Mazumder et al. (2010). The closed form solution in Lemma 1 motivates us to use the proximal gradient decent method.

4.1 Proximal gradient algorithm for subspace estimation

In order to solve (5), by using the proximal gradient decent method, we compute a series of solutions with different $\boldsymbol{\tau}$ in a decreasing order and the previous solution can be a warm start for the next solution. Algorithm 1 provides the details. After getting a series of solutions of $A_{\boldsymbol{\tau}i}$, we pick all the rank 2 solutions. Note that there are potential multiple rank 2 solutions based on pre-specified $\boldsymbol{\tau}$. We choose the best rank 2 solution by cross validation discussed in Section 4.3.

We set the starting point to be symmetric and all the steps of the computation will keep our solution always be symmetric. Thus the constraint is always satisfied. The assumption behind line 3 in Algorithm 1 is that the rank of solution A is non-increasing with respect to $\boldsymbol{\tau}$. For line 7 we perform the proximal operator following Lemma 1 and the step size t can either be a Lipschitz constant or decided by the line search. Many types of line search work and here we use the simple one proposed in Beck and Teboulle (2009b). Within the inner loop, the sequence of solution $\{\boldsymbol{A}\}$ will converge to the optimal solution with rate $O\left(\frac{1}{k}\right)$, where k is the number of iterations, as discussed in Parikh and Boyd (2014). Since the proximal gradient algorithm only shares a sublinear global rate of convergence, Beck and Teboulle (2009a) proposed Fast Iterative Shrinkage Thresholding Algorithm (FISTA) to improve the rate to $O\left(\frac{1}{k^2}\right)$. For our specific problem, we modify our algorithm following the spirit of FISTA. See Algorithm 2 for details.

Algorithm 2

basis estimation algorithm

1: **procedure** BASIS ESTIMATION ALGORITHM USING FISTA

2: Given $\tau_1 > \tau_2 > \cdots > \tau_k$ and Initialize $\boldsymbol{A}^{\text{last}}$ by random

3: $\boldsymbol{A}^{\text{last}} = \frac{\boldsymbol{A}^{\text{last}} + \boldsymbol{A}^{\text{last}T}}{2}$

4: **while** rank $\left(\boldsymbol{A}_{\tau_k}\right) \leq 2$ **do**

5: $\delta = 1$, $\theta 1 = 1$, $\boldsymbol{B}^{\text{last}} = \boldsymbol{A}^{\text{last}}$

6: **while** $\delta < \epsilon$ **do**

7: $\frac{\partial L}{\partial \boldsymbol{A}}\Big|_{\boldsymbol{A}^{\text{last}}} = \sum_{i=1}^{n} L'\left(y_i, \boldsymbol{x}_i^T \boldsymbol{A}^{\text{last}} \boldsymbol{x}_i\right) y_i \boldsymbol{x}_i \boldsymbol{x}_i^T$

8: $\boldsymbol{B}^{\text{next}} = \text{prox}_{t\tau_i}(\boldsymbol{A}^{\text{old}} - t\frac{\partial L}{\partial \boldsymbol{A}}\Big|_{\boldsymbol{A}^{\text{old}}}) = S_{t\tau_i}(\boldsymbol{A}^{\text{old}} - t\frac{\partial L}{\partial \boldsymbol{A}})$

9: $\theta_2 = \frac{1 + \sqrt{1 + 4\theta_1^2}}{2}$

10: $\boldsymbol{A}^{\text{next}} = \boldsymbol{B}^{\text{next}} + (\frac{\theta_1 - 1}{\theta_2})(\boldsymbol{B}^{\text{last}} - \boldsymbol{B}^{\text{next}})$

11: $\delta = \frac{\|\boldsymbol{A}^{\text{last}} - \boldsymbol{A}^{\text{next}}\|_F}{\max\left(1, \|\boldsymbol{A}^{\text{last}}\|_F\right)}$

12: $\theta_2 = \theta_1$, $\boldsymbol{A}^{\text{last}} = \boldsymbol{A}^{\text{next}}$, $\boldsymbol{B}^{\text{last}} = \boldsymbol{B}^{\text{next}}$

13: **end while**

14: $\boldsymbol{A}_{\tau_k} = \boldsymbol{A}^{\text{last}}$

15: **end while**

16: **end procedure**

For algorithms 1 and 2, the most time consuming step is the proximal operator $S_{t\tau_i}(\boldsymbol{A}^{\text{old}} - t\frac{\partial L}{\partial \boldsymbol{A}})$ in each iteration. This requires to compute the truncated SVD of a possibly low rank matrix especially when we use the previous low rank solution as the warm start. There is a large literature regarding SVD in numeric algebra such as Golub and Van Loan (1996) for general SVD. However, in order to handle large datasets, truncated SVD could be computed efficiently by Krylov subspace projection methods (Saad, 1992). Note that although Algorithm 2 has a faster convergence rate than Algorithm 1, Algorithm 1 may not perform worse than Algorithm 2 in terms of actual computational time due to the truncated SVD in each iteration. In particular, in line 8 of Algorithm 2, the next step is decided by the extrapolation of two previous stages and it is not necessarily a low rank matrix and may cost more time in computation than Algorithm 1 when performing SVD.

4.2 Algorithm for Weighted CBC

In order to solve weighted CBC, we can use a similar algorithm as in Section 4.1. However, in this situation the while condition in line 4 of both algorithms becomes $rank(A) \leq 1$. After obtaining the best rank one solution $\boldsymbol{A}^*$ to (9) by cross validation, we need to transform the solution γ_1, γ_2 in (9) back to $\boldsymbol{U}_1, \boldsymbol{V}_1$ by SVD.

To solve problem (11) without the rank constraint, we use the equivalent problem (11). Thus we use a similar idea in Section 2 and the algorithm remains the same except that we stop our algorithm when the rank of solution is higher than 1. Furthermore, as we include the intercept term in the input variables, we also include it in the regularization term. In order to remove the regularization on the intercept term, we use a similar idea as in the weighted CBC by setting the first elements of both α_1 and α_2 in (11) to be large so that the corresponding penalties to be small.

4.3 Choice of Rank-based Tuning parameters τ and τ_2

For the two algorithms in Section 4.1, we pre-specify a series of decreasing values τ in order to get rank two solutions. Basically we consider an equal spaced grid of τ between $[\tau_{\min}, \tau_{\max}]$. However, there may have multiple τ corresponding to rank 2 solutions, and cross validation is used to choose the best tuning parameter. Specifically, we use 5-fold cross validation and repeat 10 times to find the one with the lowest out of sample test error. For τ_2 in weighted CBC, we use a similar strategy. Since we have weights already, the range of the pre-specify τ_2 should be relatively small. We recommend to choose $\tau_2^{\max} = \tau$, which is selected in the unweighted CBC.

5 Numerical Results

In this section, we use both simulated and real data to compare CBC with other methods.

5.1 Simulation Study

Using simulation, we compare our proposed CBC and weighted CBC methods with the BDD method in Huang et al. (2012), the linear SVM, quadratic SVM, Gaussian Kernel SVM, generalized additive model (GAM) from Chouldechova and Hastie (2015), one hidden layer with one-node neural network (1–1-NN), and one hidden layer with two-node neural network (1–2-NN) (Hansen and Salamon (1990)) based on the prediction performance and model interpretability. For data visualization, we compare CBC with PCA.

In order to demonstrate that our method can identify sub-clusters within each class, we first simulate three bilinear examples shown in Figure 5. Within each class, the subclusters are generated from shifted bivariate normal distributions with parameter μ. The details are provided below.

1. Example 1 is a four-cluster-twisted case shown in the left panel of Figure 5, which includes four clusters and two for each class respectively. The first two features of each cluster are shifted by bivariate normal with means (μ, μ), $(-\mu, -\mu)$, $(-\mu, \mu)$, $(\mu, -\mu)$ correspondingly and variance equal to 1. Other dimensions

are just standard white noise. In this case, linear classifiers may be difficult to find a good classifier. The perfect classifier has a quadratic form as $sign(x_1 x_2)$. Thus we expect CBC to estimate two linear hyperplanes such as $f_1 = x_1$ and $f_2 = x_2$ up to some constant.

2. Example 2 is a four-cluster-straight case shown in the middle panel of Figure 5, which includes four clusters and two for each class respectively but at the same side. The first two features of each cluster are shifted by bivariate normal with means (μ, μ), $(\mu, -\mu)$, $(-\mu, \mu)$, (μ, μ) correspondingly and variance equal to 1. Other dimensions are also just white noise. In this case, linear classifiers are expected to perform well. The perfect classifier is a linear form as $sign(x_1)$. Thus we expect our CBC method to estimate two linear hyperplanes such as $f_1 = x_1$ and $f_2 = 1$ up to some constant.

3. Example 3 is a three-cluster-triangle case shown in the right panel of Figure 5, which includes three clusters, two for one class and the rest for the other. The first two features of each cluster are shifted by bivariate normal with means $(\mu, 0)$, $(-\mu, 0)$, $(0, \mu)$ correspondingly and variance equal to 1. Other dimensions are white noise. Similar to Example 1, this is also challenging for linear classifiers. The perfect classifier has a quadratic form as $\text{sign}((x_1 + x_2)(x_1 - x_2))$. Thus we expect our CBC method to estimate two linear hyperplanes such as $f_1 = x_1 + x_2$ and $f_2 = x_1 - x_2$ up to some constant.

4. In Example 4, we use the same generation scheme as in Example 1, but different for the last two features, which are generated by the 2-dimensional Gaussian mixture model with 4 cluster components. Each component follows bivariate normal with means (μ, μ), $(-\mu, -\mu)$, $(-\mu, \mu)$, $(\mu, -\mu)$ and variance equal to 1. Note that the last two features are useless for classification. The perfect classifier is the same as Example 1.

We evaluate our method on both low dimensional cases with dimension $p = 50$ and high dimensional cases with $p = 1000$. Both settings have the training sample size $n_1 = 100$ and testing sample size $n_2 = 1000$. In the low dimensional case, we set the shifted mean μ to be $\sqrt{5}$. For the high dimensional case, we maintain an appropriate signal to noise ratio by letting μ to be $\frac{\sqrt{p}}{8}$. The experiments are repeated for 100 times.

Tables 1 and 2 summarize prediction errors for different methods. The last column gives the estimated Bayes errors for each example. Both CBC and weighted CBC achieve the smallest misclassification rates among all these methods in all examples, and weighted CBC approximates the best performance. Note that BDD performs significantly worse in Example 4 and the corresponding standard error is large because of the difficulty of finding a good initial solution. In contrast, our convex CBC methods are robust to initial points and perform well in all these cases.

For variable selection, since the rank 1 solution of CBC and weighted CBC can be decomposed into $\boldsymbol{U}_1$, $\boldsymbol{V}_1$ in Theorem 2, we compare both methods with BDD for Example 1. In this example, the perfect classifier is the sign of the product of the first two features. In

Figure 6, we observe that compared with other methods, our weighted CBC can successfully select the first two variables and keep the other coefficients extremely small.

For data visualization, CBC can provide us a new way to visualize the data in a lower dimensional space. The main difference with PCA is that we also make use of the class label information. The top two panels of Figure 7 are 2-D projection plots generated by PCA and CBC for Example 1. Both methods can preserve the true structure of the data in the 2-D space compared with the raw data in the first plot of Figure 5. However, for Example 4, as shown in the bottom two panels of Figure 7, PCA fails to identify the within-class subcluster structure, while CBC can still identify the structure very clearly. In addition, in Figure 8, we also compare our proposed methods with BDD via projection of test data on two directional coefficients for Example 4 when $p = 50$. BDD performs worse than our proposed CBC methods. One potential reason is that the Gaussian mixture noise variables in this example make the optimization problem more challenging for BDD.

The advantage of our methods over BDD comes from several aspects. The main reason is that BDD solves a non-smooth and non-convex optimization problem (8) by using alternative minimization methods, which can not provide any guarantee of convergence to even a local solution (See examples in Powell (1973) and Razaviyayn et al. (2013)). In contrast, our proposed methods solve a series of more robust convex optimization subproblems. Second, using a smooth hinge loss instead of the standard hinge loss can avoid potential data pilling issues, especially for high dimensional low sample size applications (Marron et al. (2007)). In addition, using approximate variable selection, our weighted CBC can further improve the classification performance by strengthening the signals and reducing the effect of noise features.

5.2 Real Data Analysis

In this section, we apply our CBC and weighed CBC to the prostate cancer dataset, available at ftp://stat.ethz.ch/Manuscripts/dettling/prostate.rda, to detect whether there exists sub-clusters in each class. Prostate cancer is one of the most common cancers among men. The dataset contains 52 patients and 50 normal people with expression values for 6033 genes.

For illustration, we only keep the top 200 genes based on largest absolute values of the two sample t-statistics. We randomly split data into 80% for training and 20% for testing. Within the training data, we use 5-fold cross validation for tuning parameter selection. We repeat this procedure for 120 times. Table 3 summarizes the misclassification error rates. Our weighted CBC achieves the lowest error rate among all these methods. Furthermore, there appears to have potential subtypes in normal-like samples as shown in Figure 9. To explore further on whether these clusters indicate potential new subtype for the prostate cancer, we perform k-means clustering and select 3 clusters according to the elbow method. The results indicate that the finding of subtypes behind normal-like samples may be worthwhile to investigate further. Finally, Figure 9 shows the effective variable selection of our W-CBC method.

6 Conclusion

In this paper, we propose a convex bidirectional large margin classifier framework for high dimensional classification. Our method not only enjoys high predictive accuracy but also has good interpretability. In addition, our method provides a new data visualization tool by making use of class label information. It can be a useful tool to discover potential subclusters within each class as we demonstrate using both simulated and real data applications.

Appendix

Proof of Theorem 1

The first statement is just a special case of the latter one. Thus, it is sufficient to prove the second statement. Suppose an optimal solution A^* is not symmetric, then let $\boldsymbol{Z} = \alpha \boldsymbol{A}^* + (1 - \alpha)\boldsymbol{A}^{*T}$, which is a convex combination of $\boldsymbol{A}^*$ and $\boldsymbol{A}^{*T}$. Since (4) is a convex function, according to the definition of convexity, we have $F(\boldsymbol{Z}) \leq \alpha F(\boldsymbol{A}^*)+(1-\alpha)F(\boldsymbol{A}^{*T}) = F(\boldsymbol{A}^*)$. The last equality holds because $F(\boldsymbol{A}^*) = F(\boldsymbol{A}^{*T})$. The first statement is true when $\alpha = \frac{1}{2}$.

Proof of Theorem 2

Since $\boldsymbol{A}(\tau)$ is an optimal solution with rank $r = 2$ to (5), by spectral decomposition, $\boldsymbol{A}(\tau) = \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{U}_r^T$. Let $z_i = \boldsymbol{x}_i U_r$, where $z_i \in \mathscr{R}^r$, then we can rewrite the objective value as $G(\boldsymbol{A}(\tau)) = \sum_{i=1}^n L\left(y_i, \boldsymbol{x}_i^T \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{U}_r^T \boldsymbol{x}_i\right) + \tau \|\boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{U}_r^T\|_* = \sum_{i=1}^n L\left(y_i, z_i^T \boldsymbol{\Sigma}_r z_i\right) + \tau \mathrm{Tr}\left(|\boldsymbol{\Sigma}_r|\right)$, where $\mathrm{Tr}(\boldsymbol{X})$ is the trace norm function over square matrices.

In order to find a $\boldsymbol{A}^*(\tau)$ with rank $k = 1$, we claim that the space of $\boldsymbol{A}^*(\tau)$ is formed by $\boldsymbol{U}_r$. Thus we want to find two matrices $\boldsymbol{V}_1, \boldsymbol{V}_2 \in \mathscr{R}^{r \times k}$ such that $\boldsymbol{A}^*(\tau) = \boldsymbol{U}_r \boldsymbol{V}_1 \boldsymbol{V}_2^T \boldsymbol{U}_r^T$. Since we need $G(\boldsymbol{A}(\tau)) = G(\boldsymbol{A}^*(\tau))$, we have the following equation:

$$\sum_{i=1}^n L\left(y_i, z_i^T \boldsymbol{\Sigma}_r z_i\right) + \tau \mathrm{Tr}\left(|\boldsymbol{\Sigma}_r|\right) = \sum_{i=1}^n L\left(y_i, \boldsymbol{x}_i^T \boldsymbol{U}_r \boldsymbol{V}_1 \boldsymbol{V}_2^T \boldsymbol{U}_r^T \boldsymbol{x}_i\right) + \tau \|\boldsymbol{U}_r \boldsymbol{V}_1 \boldsymbol{V}_2^T \boldsymbol{U}_r^T\|_* = \sum_{i=1}^n L\left(y_i, z_i^T \boldsymbol{V}_1 \boldsymbol{V}_2^T z_i\right) + \tau \|\boldsymbol{V}_1 \boldsymbol{V}_2^T\|_*.$$

In order to get the equality for every $z_i \in \mathscr{R}^r$, we need to solve the following two equations:

$$\frac{\left(\boldsymbol{V}_1 \boldsymbol{V}_2^T + \boldsymbol{V}_2 \boldsymbol{V}_1^T\right)}{2} = \boldsymbol{\Sigma}_r \quad (1)$$

$$\|\boldsymbol{V}_1 \boldsymbol{V}_2^T\|_* = \mathrm{Tr}\left(|\boldsymbol{\Sigma}_r|\right). \quad (2)$$

Then we can check that the following $\boldsymbol{V}_1$, $\boldsymbol{V}_2$ satisfy the equation above:

$$\begin{aligned} \boldsymbol{V}_1 &= (\sqrt{|\lambda_1|}, \sqrt{|\lambda_2|}) \\ \boldsymbol{V}_2 &= (\sqrt{|\lambda_1|}, -\sqrt{|\lambda_2|}), \end{aligned} \tag{12}$$

where λ_1, λ_2 are the diagonal values of Σ_r.

Proof of Theorem 3

We first introduce the following lemma:

Lemma 2. *For any matrix* $\boldsymbol{A} \in \mathscr{R}^{m \times n}$ *with* $r = rank(\boldsymbol{A}) < \min(m, n)$, *we have the following equation:*

$$\|\boldsymbol{A}\|_* = \min_{\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{A} = \boldsymbol{U}\boldsymbol{V}^T} \frac{1}{2}\left(\|U\|_F^2 + \|V\|_F^2\right),$$

where the minimum is attained at a factor decomposition $\boldsymbol{A} = \boldsymbol{U}_r \boldsymbol{V}_r^T$

Proof. See the proof in Lemma 6 of Mazumder et al. (2010).

Note that (8) can also be represented as:

$$\begin{aligned} &\underset{\boldsymbol{A} \in \mathscr{R}^{p \times p}}{\text{minimize}} \quad \sum_{i=1}^{n} L\left(y_i \boldsymbol{x}_i^T \boldsymbol{A} \boldsymbol{x}_i\right) + \frac{\tau}{2}\left(\|\boldsymbol{U}_r\|_F^2 + \|\boldsymbol{V}_r\|_F^2\right) \\ &\text{subject to} \quad \boldsymbol{A} = \boldsymbol{U}_r \boldsymbol{V}_r^T. \end{aligned}$$

By using Lemma 2, we get the following equivalent problem:

$$\begin{aligned} &\underset{\boldsymbol{A} \in \mathscr{R}^{p \times p}}{\text{minimize}} \quad \sum_{i=1}^{n} L\left(y_i \boldsymbol{x}_i^T \boldsymbol{A} \boldsymbol{x}_i\right) + \tau \|\boldsymbol{A}\|_* \\ &\text{subject to} \quad \boldsymbol{A} = \boldsymbol{U}_r \boldsymbol{V}_r^T. \end{aligned}$$

That is

$$\begin{aligned} &\underset{\boldsymbol{A} \in \mathscr{R}^{p \times p}}{\text{minimize}} \quad \sum_{i=1}^{n} L\left(y_i \boldsymbol{x}_i^T \boldsymbol{A} \boldsymbol{x}_i\right) + \tau \|\boldsymbol{A}\|_* \\ &\text{subject to} \quad \text{rank}(\boldsymbol{A}) = r. \end{aligned}$$

Note that $r = 1$ is a special case of Theorem 3. Thus we have the conclusion that the solution to (6) lies in the solution path of (8).

References

Bach F. High-dimensional non-linear variable selection through hierarchical kernel learning. arXiv preprint arXiv:0909.0844, 2009.

Bartlett PL, Jordan MI, and McAuliffe JD. Large margin classifiers: Convex loss, low noise, and convergence rates. In NIPS, pages 1173–1180, 2003.

Beck A and Teboulle M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM journal on imaging sciences, 2(1):183–202, 2009a.

Beck A and Teboulle M. Gradient-based algorithms with applications to signal recovery. Convex Optimization in Signal Processing and Communications, pages 42–88, 2009b.

Bertsimas D, Chang A, and Rudin C. An integer optimization approach to associative classification. Adv. Neur. Inf. Process. Syst, 25:269–277, 2012.

Bradley PS and Mangasarian OL. Feature selection via concave minimization and support vector machines. In ICML, volume 98, pages 82–90, 1998.

Cai J-F, Candes EJ, and Shen Z. A singular value thresholding algorithm for matrix completion. SIAM Journal on Optimization, 20(4):2010, 2009.

Candès EJ and Tao T. The power of convex relaxation: Near-optimal matrix completion. Information Theory, IEEE Transactions on, 56(5):2053–2080, 2010.

Chouldechova A and Hastie T. Generalized additive model selection. arXiv preprint arXiv:1506.038*50*, 2015.

Freund Y and Schapire RE. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences, 55(1):119–139, 1997.

Freund Y, Schapire R, and Abe N. A short introduction to boosting. Journal-Japanese Society For Artificial Intelligence, 14(771–780):1612, 1999.

Golub GH and Van Loan CF. Matrix computations. johns hopkins studies in the mathematical sciences, 1996.

Hansen LK and Salamon P. Neural network ensembles. IEEE transactions on pattern analysis and machine intelligence, 12(10):993–1001, 1990.

Hastie T, Tibshirani R, and Friedman J. The elements of statistical learning. 2001 NY Springer, 2001.

Huang H, Liu Y, and Marron JS. Bidirectional discrimination with application to data visualization. Biometrika, 99(4):851–864, 2012. [PubMed: 23843672]

Lin X, Wahba G, Xiang D, Gao F, Klein R, and Klein B. Smoothing spline anova models for large data sets with bernoulli observations and the randomized gacv. Annals of Statistics, pages 1570–1600, 2000.

Lin Y, Zhang HH, et al. Component selection and smoothing in multivariate nonparametric regression. The Annals of Statistics, 34(5):2272–2297, 2006.

Liu Y, Zhang HH, and Wu Y. Hard or soft classification? large-margin unified machines. Journal of the American Statistical Association, 106(493):166–177, 2011. [PubMed: 22162896]

Marron JS, Todd MJ, and Ahn J. Distance-weighted discrimination. Journal of the American Statistical Association, 102(480):1267–1271, 2007.

Mazumder R, Hastie T, and Tibshirani R. Spectral regularization algorithms for learning large incomplete matrices. The Journal of Machine Learning Research, 11:2287–2322, 2010. [PubMed: 21552465]

Parikh N and Boyd SP. Proximal algorithms. Foundations and Trends in optimization, 1(3): 127–239, 2014.

Powell MJ. On search directions for minimization algorithms. Mathematical Programming, 4(1): 193–201, 1973.

Razaviyayn M, Hong M, and Luo Z-Q. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. SIAM Journal on Optimization, 23(2):1126–1153, 2013.

Rennie JD and Srebro N. Loss functions for preference levels: Regression with discrete ordered labels. In Proceedings of the IJCAI multidisciplinary workshop on advances in preference handling, pages 180–186. Kluwer Norwell, MA, 2005.

Rosset S and Zhu J. Piecewise linear regularized solution paths. The Annals of Statistics, pages 1012–1030, 2007.

Saad Y. Numerical methods for large eigenvalue problems, volume 158 SIAM, 1992.

Schapire RE, Freund Y, Bartlett P, and Lee WS. Boosting the margin: A new explanation for the effectiveness of voting methods. Annals of statistics, pages 1651–1686, 1998.

Tibshirani R. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), pages 267–288, 1996.

Tibshirani R, Hastie T, Narasimhan B, and Chu G. Diagnosis of multiple cancer types by shrunken centroids of gene expression. Proceedings of the National Academy of Sciences, 99(10): 6567–6572, 2002.

Vapnik V. Statistical learning theory. 1998, 1998.

Zhu J, Rosset S, Tibshirani R, and Hastie TJ. 1-norm support vector machines In Thrun S, Saul LK, and Schölkopf B, editors, Advances in Neural Information Processing Systems 16, pages 49–56. MIT Press, 2004 URL http://papers.nips.cc/paper/2450-1-norm-support-vector-machines.pdf.

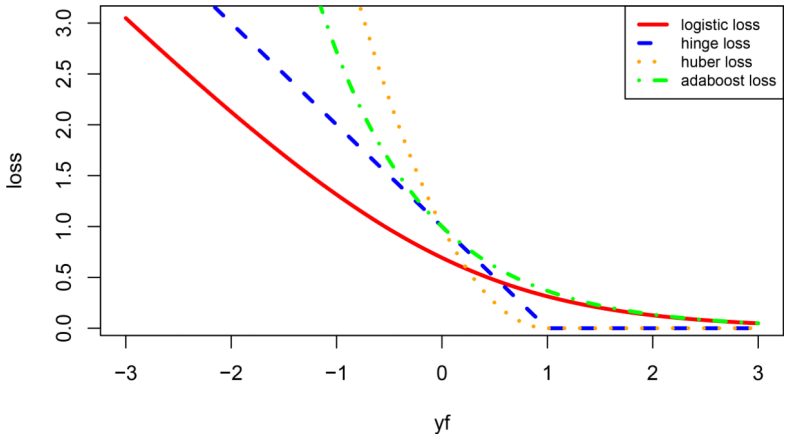


Figure 1:

A plot of different loss functions, including the hinge loss, logistic binomial deviance loss, exponential loss for AdaBoost and Huber loss. The horizontal axis represents the margin *yf*.

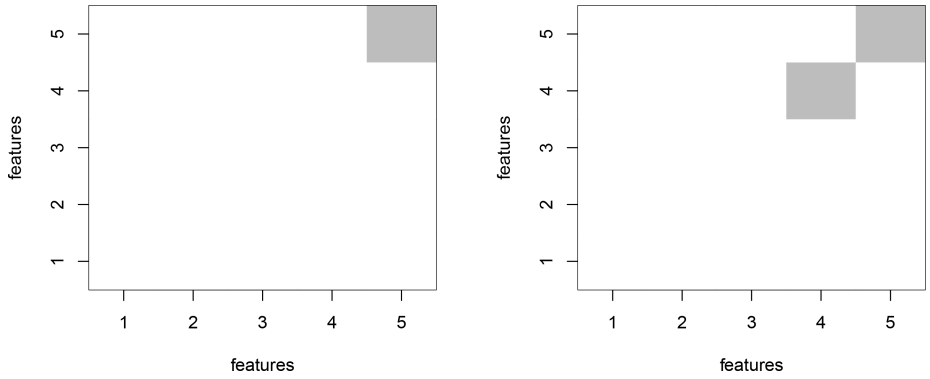


Figure 2:

Illustration of the coefficient matrix ***A*** with size 5 by 5. The coefficients are either 1 (gray) or 0 (white). The left panel shows a matrix of rank 1, which represents interaction between the first and the last features. The right panel shows a matrix with rank 2 with an additional interaction between the second and the fourth features.

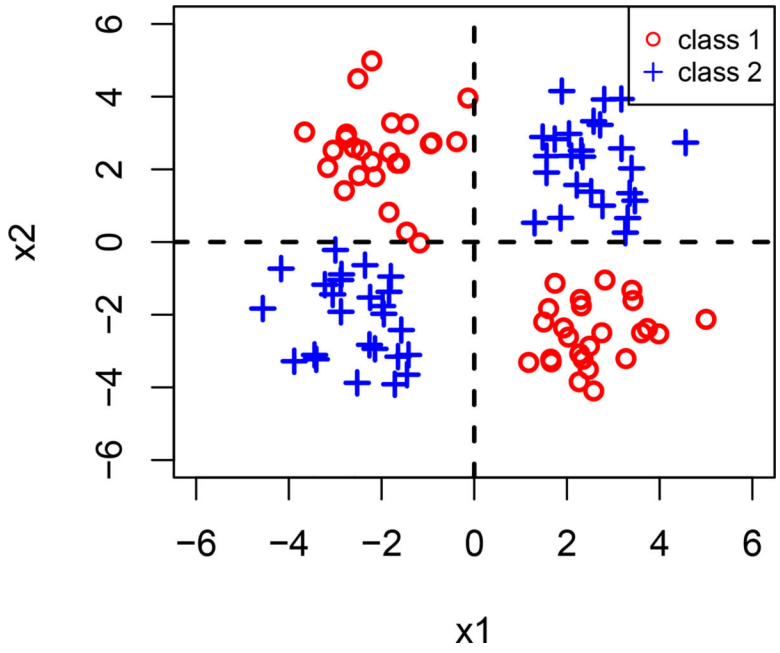
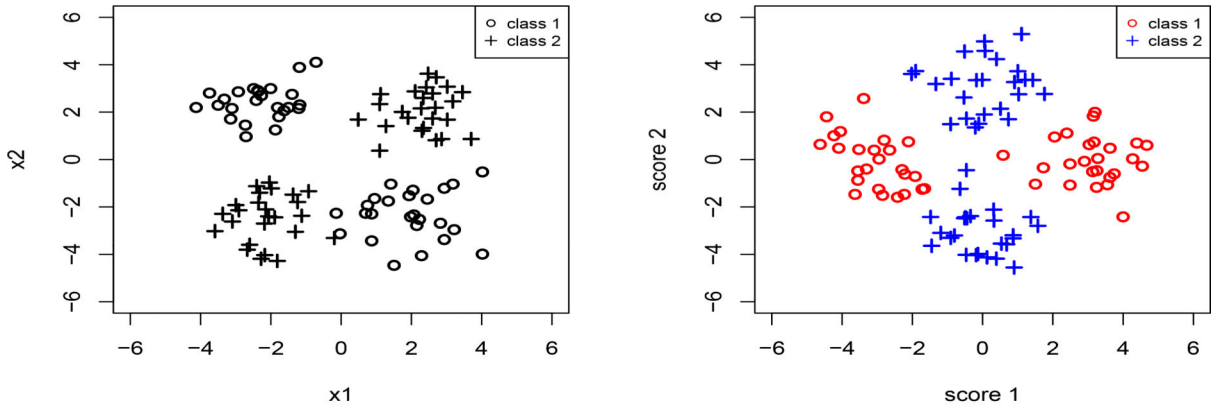


Figure 3:
Illustration of the decision function $(x^T w_1)(x^T w_2)$, where $w_1 = (0, 1, 0)$ and $w_2 = (0, 0, 1)$. Noe that the two classes can be separated by two hyperplanes.



(a) original plot (b) 2-D visualization plot

Figure 4:
(a) is a 4-cluster twisted example with 50 input features, whose class labels are decided by the sign of first two features $x1$, $x2$. (b) is 2-D visualization plot by projecting all the features into 2 orthogonal directions estimated by our CBC method.

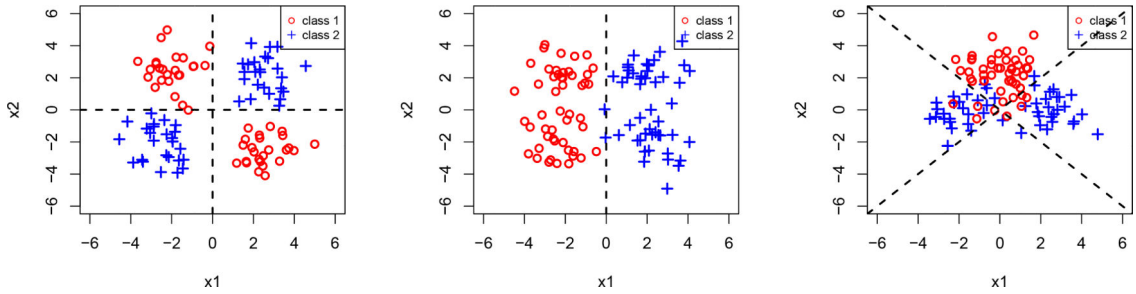


Figure 5:

The first panel shows a two-class-twisted example, the middle one corresponds to a two class-straight example, the right panel contains a 3-cluster-triangle example and their corresponding decision boundaries.

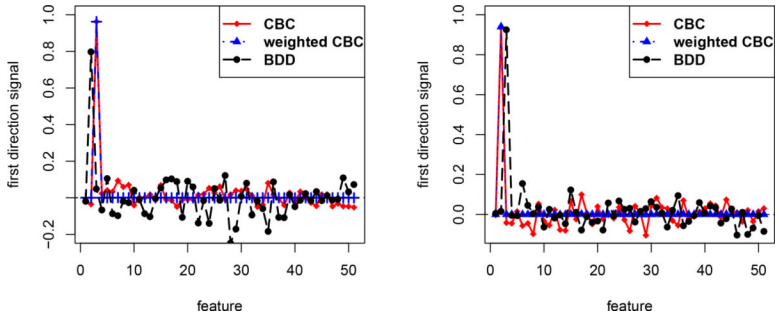


Figure 6:

Coefficient estimation for the two directions for Example 1. All the coefficients have been normalized. The results show that the weighted CBC works the best, followed by CBC, and BDD performs the worst.

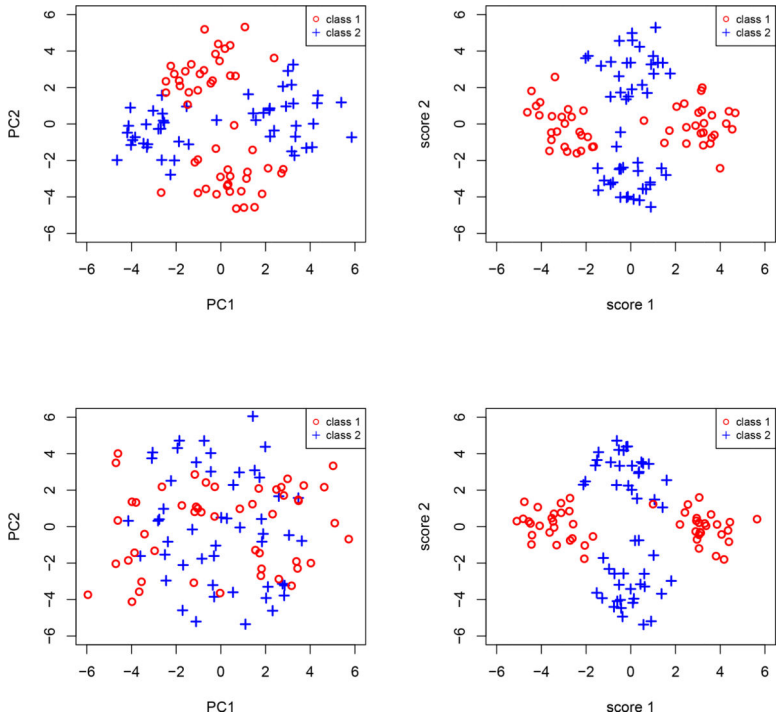


Figure 7:

The top two panels are 2-D visualization plots of Example 1 by PCA and CBC. The bottom two corresponding to Example 4 by PCA and CBC. In the normal setting in Example 1, both methods can capture sub-cluster structure. However, in Example 4 where there are uncorrelated but clustered noise features, CBC can still capture the classification structure correctly but PCA fails to do so.

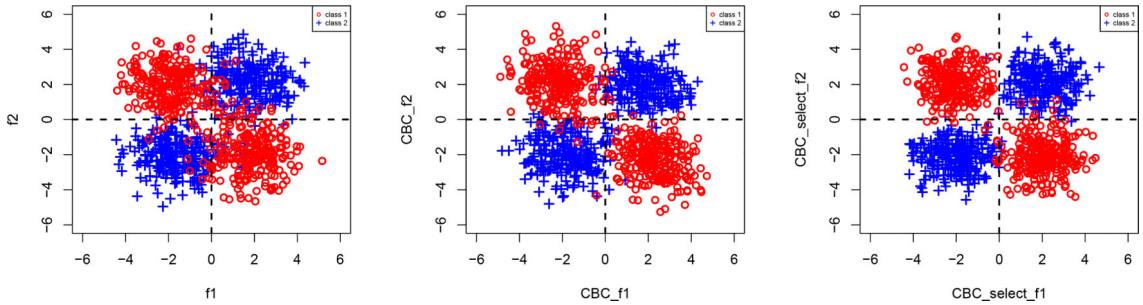


Figure 8:

Plots of the projection of testing data of Example 4 on two directional coefficients to get f_1 and f_2 by using BDD, CBC and weighted CBC, respectively. Note that BDD fails to correctly classify some instances while both CBC and weighted CBC work better. In addition, weighted CBC has slightly more accurate boundary than CBC.

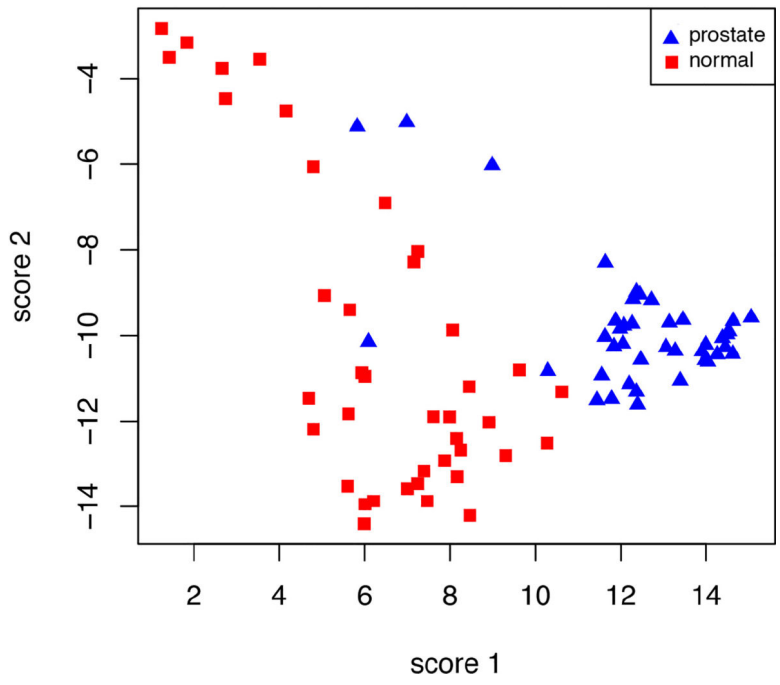


Figure 9:
The 2-D visualization plot of the prostate cancer data by our CBC method. There are potential two clusters within normal patients which are located at top left and bottom of the plot.

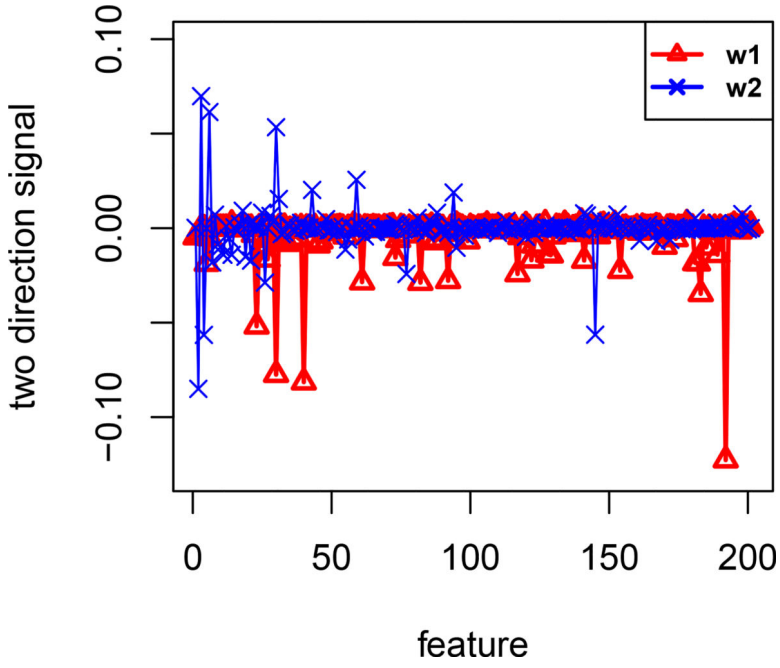


Figure 10:
The plot of two directional coefficients. We can see several genes play an important role in both directions.

Table 1:

Comparison of test errors (%) among different classifiers in four simulated examples ($p = 50$) and the corresponding standard errors (%) in parenthesis.

Method	Example 1	Example 2	Example 3	Example 4
W-CBC	**2.46 (0.00)**	**1.34 (0.36)**	**11.05 (1.06)**	**2.51 (0.45)**
CBC	3.47 (0.00)	1.83 (0.46)	12.51 (1.17)	3.97 (0.72)
BDD	5.68 (0.02)	2.98 (0.97)	16.04 (2.61)	9.61 (8.45)
Linear-SVM	49.9 (0.04)	2.17 (0.74)	18.5 (2.13)	49.74 (2.08)
Quad-SVM	7.22 (0.01)	21.71 (1.86)	20.57 (1.62)	10.52 (1.29)
Gaussian-SVM	43.36 (0.10)	13.62 (10.08)	44.42 (6.71)	45.07 (2.35)
GAM	50.00 (0.11)	1.39 (0.03)	14.80 (0.13)	50.03 (0.08)
1–1-NN	50.00 (0.14)	2.87 (0.01)	25.96 (0.22)	50.18 (0.14)
1–2-NN	47.00 (0.5)	3.18 (0.1)	24.73 (0.27)	48.65 (0.31)
Estimated optimal	2.46 (0.00)	1.34 (0.36)	10.79 (0.95)	2.50 (0.44)

Table 2:

Comparison of test errors (%) among different classifiers for four simulated examples ($p = 1000$) and the corresponding standard errors (%) in parenthesis.

Method	Example 1	Example 2	Example 3	Example 4
W-CBC	**0.01 (0.03)**	**0.00 (0.02)**	**1.15 (0.85)**	**0.01 (0.03)**
CBC	0.25 (0.21)	0.26 (0.39)	5.80 (0.88)	0.39 (0.24)
BDD	0.32 (0.18)	0.22 (0.16)	9.33 (9.39)	2.92 (5.60)
Linear-SVM	49.86 (1.60)	0.20 (0.15)	15.68 (1.32)	49.92 (1.40)
Quad-SVM	3.80 (1.66)	44.77 (1.63)	25.65 (3.67)	7.27 (2.07)
Gaussian-SVM	45.02 (4.73)	32.63 (16.04)	45.66 (5.26)	44.34 (4.34)
GAM	49.92 (0.12)	0.00 (0.00)	2.85 (0.06)	49.88 (0.10)
1–1-NN	49.89 (0.15)	1.18 (0.46)	27.39 (0.19)	49.94 (0.15)
1–2-NN	50.00 (0.15)	0.52 (0.22)	26.81 (0.26)	50.00 (0.15)
Estimated optimal	0.01 (0.03)	0.00 (0.02)	0.52 (0.22)	0.01 (0.03)

Table 3:

Comparison of test errors (%) among different classifiers for the prostate cancer dataset and the corresponding standard errors (%) in parenthesis.

W-CBC	CBC	BDD	Linear-SVM	Quad-SVM	Gaussian-SVM	GAM	1–1-NN	1–2-NN
7.30 (5.34)	7.74 (5.38)	12.2 (6.72)	11.94 (7.36)	7.86 (5.15)	8.02 (5.24)	8.56 (4.81)	7.48 (5.32)	7.52 (4.97)