NETWORK-BASED SUPPORT VECTOR MACHINES FOR CLASSIFICATION OF MICROARRAY GENE EXPRESSION DATA

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

YANNI ZHU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

August, 2009

Acknowledgments

Without all the wonderful people who have helped and inspired me in the past five years, my thesis would not have become a reality. I would like to acknowledge all of them for their support and encouragement during my study in Biostatistics at the University of Minnesota.

First of all, I would like to express my earnest gratitude to my advisor Prof. Wei Pan. Prof. Pan advised me on both master thesis and doctoral thesis. His creative ideas, stimulating suggestions, and inspirational encouragement helped me in all the time of research for and writing of this thesis. His perpetual passion on new knowledge, everlasting diligence in teaching and research, and keen sense of responsibility for work are worth learning during my entire lifetime.

Prof. William Thomas, Prof. Baolin Wu, and Prof. Xiaotong Shen deserve a special thanks as my thesis committee members. I appreciate for their time to read my thesis and provide me with invaluable advice at various stages.

I benefited greatly from the course offered at the Division of Biostatistics and the School of Statistics. I would like to express my sincere gratitude to all the faculty members. In particular, I would like to thank all my supervisors for whom I was a research or teaching assistant in the past five years. It was my pleasure to work for Prof. Wei Pan, Prof. William Thomas, Prof. Thomas Nevins, Prof. Lynn Eberly, Prof. Judith Garrard, Prof. Baolin Wu, Prof. Xianghua Luo, and Prof. James Neaton. The associated research and teaching experience not only furthered my understanding of but

also broadened my perspective on the practical aspects of what was learned in class.

I also want to thank Dr. Haoyu Yu at the Supercomputer Institute, who generously contributed her expertise and time to answer my questions and requests for computing resources.

My deepest gratitude goes to my parents for their unflagging love, care, and support throughout my life. They gave me momentum that made me persist in my study. They have been always by my side every step of the way. I greatly appreciate my best counselor and mentor Dr. Yong Wang for his inspirational support. Without his encouragement, I would not have begun my study in Biostatistics. His advice helped me pass through my difficult times.

Last but not least, thanks be to all that have ever truly cared about me.

Abstract

The importance of network-based approach to identifying biological markers for diagnostic classification and prognostic assessment in the context of microarray has been increasingly recognized. Standard methods treat all genes independently and identically *a priori* and ignore the biological observation that genes function together in biological processes. For binary classification, we are motivated to improve predictive accuracy and gene selection by developing novel network-based classification tools that explicitly incorporate interrelationships of genes as described by gene networks.

We propose three network-based support vector machines (SVM) by suitably forming the penalty term. The neighboring-gene (NG) penalty groups pairwise gene neighbors and sums up the L_∞-norm of each group over the entire network, leading to NG-SVM. NG-SVM tends to select pairs of neighboring genes. The disease-gene-centric (DGC) penalty is constructed on groups defined on an upper-lower hierarchy imposed on the undirected network. DGC-SVM aims to detect collectives of genes clustering together and around some key disease genes. The truncated L_∞-norm (TL_∞) penalty intends to correct bias induced by penalization through a threshold parameter $C > 0$ built into the L_∞-norm as used in NG-SVM and DGC-SVM. Simulation studies and real data applications demonstrate that the proposed methods are able to capture more disease genes and less noise genes than the existing popular methods, standard SVM and L_1-SVM. We conclude that the proposed methods have the potential to be effective classification tools for microarrays and other high-dimensional data.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The past two decades have witnessed rapid advances in gene expression profiling with the microarray technology (Brown and Botstein 1999, Hackl *et al.* 2004). Gene expression is the process by which the genetic information contained within DNA is transcribed into messenger RNA (mRNA), and mRNA are then translated into the proteins that perform most of the critical functions of cells. The underlying mechanism of this process is highly complex and tightly regulated. It controls which genes are expressed in a cell and also when to increase or decrease the expression level of particular genes as necessary. The microarray technology allows researchers to monitor expression levels of thousands of genes at one time, and to explore primary disease mechanisms by comparing gene expression profiles for malignant and normal cells. It not only brightens the prospect of deciphering the complexity of disease genesis and progression at the genomic level, but also revolutionizes the diagnostic, therapeutic, and prognostic approaches (Sun *et al.* 2004, Bao and Davidson 2008, Nannini *et al.* 2009, Santos *et al.* 2009). Up to recently, diagnostic classification and prognostic assessment have been based on conventional clinical and pathological risk factors, such as patient age and tumor size, many of which are believed to be secondary manifestation (Chuang *et al.* 2007). The regularity and aberration in the expression patterns of certain genes shed light on their functions and pathological importance (Frolov *et al.* 2003). Studies that seek to identify gene markers to refine diagnostic classification and improve prognostic prediction in the context of gene expression data have enriched the literature (Yang 2007, Wang *et al.* 2005, Xiong *et al.* 2001). Since a typical microarray data set records expression levels

of thousands of genes from only tens of samples, there is high noise level inherent in the high-dimensional data ("large p, small N"), which poses big challenges to statisticians, such as the challenge of variable selection and feature extraction to generate sparse model for the ease of interpretation (Fan and Li 2006).

Biological observations reveal that genes interact with each other through their RNA and protein expression products (Lee *et al.* 2002, Vermeirssen *et al.* 2007). For example, the rate at which transcription factor genes are transcribed into RNA molecules may govern the transcriptional rate of their regulatory target genes, which as a result become either up- or down- regulated. A gene network is a collection of effective interactions, describing the multiple ways through which one gene affects all the others to which it is connected. A gene network reveals genetic dynamics underlying the aggregate function that the network maintains.

High-throughput genomic advances have generated various databases providing gene network information, such as the Biomolecular Interaction Network Database (BIND) (Alfarano *et al.* 2005), the Human Protein Reference Database (HPRD) (Peri *et al.* 2004), and the Kyoto Encyclopedia of Genes and Genomes (KEGG) (Kanehisa 2004). In recent years, genetic studies have uncovered hundreds of genes with variants that predispose to common diseases, such as cancer (Dong *et al.* 2009), Parkinson's disease (Patra *et al.* 2009), and diabetes (Holmkvist *et al.* 2008). For example, gene *TP53* is among the most well-known ones, which, as a tumor suppressor, is central to many anti-cancer mechanisms. Gene *TP53* encodes tumor protein *p53*, the so-called "the guardian

of the genome", which mediates the cellular response to DNA damage and is involved in other important biological processes, e.g., cell cycle. Among its other functions, *p53* activates other genes to fix the damage if *p53* determines that the DNA can be repaired. Otherwise, *p53* prevents the cell from dividing and signals its death. Most mutations that deactivate *TP53* destroy protein *p53*'s ability to regulate other genes properly and thus leads to increasing risk of tumor development (Borresen 2003, Soussi and Béroud 2003). Hence, not just a single gene, but a subnetwork of *TP53* and its interacting partners, are involved in the disease progression.

Since its invention (Cortes and Vapnik 1995, Vapnik 1995), support vector machine (SVM) has been acclaimed as a useful regularization method due to its flexibility in penalty specification, its excellent empirical performance, especially for high dimensional data (Brown *et al.* 2000, Furey *et al.* 2000), and resulting model sparsity if a suitable penalty (e.g. L_1-norm) is employed. For binary classification, STD-SVM has good predictive performance, but fails to conduct variable selection with all its nonzero coefficient estimates. The L_1-SVM (Wang and Shen 2007, Zhu *et al.* 2003) causes some estimates to be exactly zero and produces sparse models for data with $p >> n$, thus overcoming the shortcoming of STD-SVM in variable selection. Zou and Yuan (2008) developed the concept of grouped variable selection to further improve the model sparsity compared with the L_1-SVM. By use of an F_∞-norm SVM, features derived from the same categorial predictor are included or excluded simultaneously, achieving factor-wise variable selection. Their grouping scheme was based on non-overlapping groups. Zhao *et al.*

(to appear) introduced the composite absolute penalties (CAP) family, which realizes not only grouped selection for non-overlapping groups but also hierarchical selection for over-lapping groups. CAP generalized the concept of grouped variable selection. With the availability of the various gene networks and the accumulating knowledge on genes linked to diseases, we are motivated to incorporate the prior biological into building classifiers with the hope to improve predictive accuracy and gene selection by extending the idea of grouping to gene networks. Because of the ever-increasing popularity of penalization methods for high-dimensional data, we propose three network-based penalties to be used with the hinge loss respectively, leading to three network-based SVMs. While maintaining some desirable properties of support vector machine (SVM) with the hinge loss function, the network-based penalties directly integrate a gene network to realize more effective variable selection, as compared with generic methods, such as the standard L_2-norm SVM (STD-SVM) and L_1-penalized SVM (L1-SVM).

In Chapter 2, we propose a neighboring-gene SVM (NG-SVM) that treats any two neighboring genes in a network as one group. The empirical results show that NG-SVM enjoys advantages in gene selection and predictive performance compared with the popular STD-SVM and L_1-SVM. However, a potential problem of NG-SVM resides in its tendency to select isolated genes or gene pairs, i.e., genes largely disconnected to the rest in the network, which is not desirable given that some disease genes cluster together and form subnetworks. Chapter 2 has been published as Zhu *et al.* (2009). In Chapter 3, we introduce two disease-gene-centric SVMs (DGC-SVM) by exploring

two different ways of forming gene groups, the pathway grouping and the partial tree grouping. The DGC-SVMs aim to detect collectives of genes connected with or clustering around disease genes that are known to be linked to the disease. Chapter 3 has been accepted for publication. Chapter 4 develops a truncated L_∞-norm SVM (TL_∞-SVM) that is designed to correct the bias from penalization. A threshold parameter $C > 0$ (a second tuning parameter) is built into the L_∞-norm. The performance of TL_∞-SVM, NG-SVM, L_1-SVM, and STD-SVM is compared. Chapter 5 provides a short summary and discussion on some open questions and future work.

Chapter 2

Support Vector Machine with a Neighboring-gene Penalty

2.1 Introduction

In recent years, researchers have realized that gene markers identified from microarrays drawn from different studies on the same disease across similar cohorts lack consistency (Ein-Dor *et al.* 2005, Ein-Dor *et al.* 2006). A possibly more effective means to resolve this problem is to employ a network-based approach, that is, to identify markers as gene subnetworks, defined as groups of functionally related genes based on a gene network, instead of treating individual genes as completely independent and identical a priori as in most existing approaches (Chuang *et al.* 2007). A novel network-based approach proposed recently (Chuang *et al.* 2007, Liu *et al.* 2007) can be summarized as follows: (1) randomly searching subnetworks and assigning a score to each subnetwork that characterizes the subnetwork-wise gene expression level; (2) identifying significant subnetworks that can well discriminate the clinical outcome; (3) constructing a classifier based on the significant subnetworks with a conventional statistical tool, such as logistic regression. Essentially such a network-based approach aggregates gene expression data at the subnetwork level, and then identifies and utilizes some significant subnetworks. It has been shown that such a network-based approach not only improves predictive performance and reproducibility, but also sheds biological insights into molecular mechanisms underlying the clinical outcome. However, the above method is largely heuristic without a formal statistical framework; more importantly, it involves a random search over subnetworks, leading to possibly different results from different runs with no guarantee of the optimality of the final result.

SVM (Cortes and Vapnik 1995, Vapnik 1995) is one of the most popular supervised learning techniques with wide-ranging applications. Previous studies have demonstrated its superior performance in gene expression data analysis, especially its ability to handle high dimensional data (Brown *et al.* 2000, Furey *et al.* 2000). Nevertheless, with categorical predictors, both the STD-SVM and the L_1-SVM may have some shortcomings. Zou and Yuan (2008) applied the concept of grouped variable selection and developed an F_∞-norm penalized SVM to realize simultaneous selection/elimination of all the features derived from the same categorical factor (or a group of variables). Their numerical examples showed that the F_∞-norm SVM outperformed the L_1-SVM in factor-wise variable selection. In this chapter, we extend the idea of variable grouping to gene networks: rather than grouping all the features created from the same categorical factor, we treat any two neighboring genes in a network as one group. The NG penalty is constructed as the sum of the L_∞-norm being applied to the groups of neighboring-gene pairs. With the hinge loss penalized by such an NG penalty as our objective function, we obtain our NG-SVM.

This chapter is organized as follows. We begin with a brief review of the SVM, and then introduce our proposed NG-SVM. We evaluate its performance by simulation studies in both low dimensional and high dimensional data settings as well as two real data applications. The last section concludes this chapter with a brief summary.

2.2 Methods

2.2.1 Existing methods

Suppose we have training data $\{(x_i, y_i)\}_{i=1}^N$ with $x_i \in \mathbb{R}^p$ and $y_i \in \{1, -1\}$. Define a hyperplane $\{x : f(x) = x^T\beta + \beta_0 = 0\}$. The classification rule induced by $f(x)$ is $sign[\hat{f}(x)]$. SVM searches for such a hyperplane $\hat{f}(x) = x^T\hat{\beta} + \hat{\beta}_0$ that maximizes the margin between the training data points for class 1 and class -1:

$$\begin{aligned} &\max_{\beta,\beta_0} \frac{1}{\|\beta\|_2} \\ \text{subject to} \quad &y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i, \quad \forall i \\ &\xi_i \geq 0, \quad \sum_{i=1}^N \xi_i \leq C \end{aligned} \tag{2.1}$$

where ξ_i are slack variables, and C is a tuning parameter to be determined. The STD-SVM has an equivalent *hinge loss + penalty* formulation as an optimization problem (Wahba *et al.* 2000, Hastie *et al.* 2000, Zou and Yuan 2008):

$$\min_{\beta_0,\beta} \left\{ \sum_{i=1}^N \left[1 - y_i\left(x_i^T\beta + \beta_0\right)\right]_+ + \lambda\|\beta\|_2^2 \right\} \tag{2.2}$$

where the subscript "+" denotes the positive part, i.e., $z_+ = max\{z, 0\}$, $\|\beta\|_2^2 = \sum_{k=1}^p |\beta_k|^2$, and λ is the tuning parameter. The solution to (2.1) is the same as that to (2.2).

The above STD-SVM forces all nonzero coefficient estimates, which leads to the problem of its inability to conduct variable selection. The L_1-SVM (Zhu *et al.* 2003)

was proposed to accomplish the goal of variable selection. It can be formulated as

$$\min_{\beta_0,\beta}\left\{\sum_{i=1}^{N}\left[1-y_i\left(x_i^T\beta+\beta_0\right)\right]_+ + \lambda\|\beta\|_1\right\} \tag{2.3}$$

where $\|\beta\|_1 = \sum_{k=1}^{p}|\beta_k|$. The L_1-SVM wins over the STD-SVM when the true model is sparse, while the STD-SVM is preferred if there are not many redundant noise features (Friedman *et al.* 2004).

Zou and Yuan (2008) pointed out the shortcoming of the L_1-norm penalty: even though it encourages parsimonious models, it fails to guarantee successful models in cases of categorical predictors due to the fact that each dummy variable is selected independently. They applied the concept of grouped variable selection and proposed an F_∞-norm SVM to realize simultaneous selection/elimination of features derived from the same factor so as to accomplish automatic factor-wise variable selection. Suppose we have G factors $F_1,\ldots,F_G$. From each factor F_g, we generate a feature vector $x_{(g)} = (x_1^{(g)},\cdots,x_j^{(g)},\cdots,x_{n_g}^{(g)})^T$. Correspondingly we have the coefficient vector $\beta_{(g)} = (\beta_1^{(g)},\cdots,\beta_j^{(g)},\cdots,\beta_{n_g}^{(g)})^T$. Therefore,

$$f(x) = x^T\beta+\beta_0 = \sum_{g=1}^{G}x_{(g)}^T\beta_{(g)}+\beta_0 \tag{2.4}$$

Define the F_∞-norm of F_g as

$$\|F_g\|_\infty = \|\beta_{(g)}\|_\infty = \max_{j\in\{1,\cdots,n_g\}}\{|\beta_j^{(g)}|\} \tag{2.5}$$

The F_∞-norm SVM is formulated as

$$\min_{\beta_0,\beta}\left\{\sum_{i=1}^{N}\left[1-y_i\left(\sum_{g=1}^{G}x_{i,(g)}^T\beta_{(g)}+\beta_0\right)\right]_+ + \lambda\sum_{g=1}^{G}\|\beta_{(g)}\|_\infty\right\} \tag{2.6}$$

The most noteworthy property of the F_∞-norm SVM is its guarantee of sparsity at the factor level. Due to the singularity property of the infinity norm: $\|\beta_{(g)}\|_\infty$ is not differentiable at $\beta_{(g)} = 0$, $\beta_{(g)}$ will be exactly zero if the regularization parameter λ is properly chosen (Zou and Yuan 2008). Therefore, the F_∞-norm SVM automatically eliminates factors that are completely irrelevant to the response, and thus achieves the goal of factor-wise selection. The empirical evidence shows that the F_∞-norm SVM often outperforms both the L_1-SVM and the STD-SVM.

2.2.2 Neighboring-gene support vector machine

Biological observations reveal that neighboring genes in a network tend to function together in biological processes. To incorporate this prior information, a network-based SVM for binary classification is proposed to facilitate generating models that extract more biological insight from gene expression data. The penalty term that characterizes the network structure can be specified by implanting the F_∞-norm into the context of known functional interrelationships among genes by considering each pair of the functionally related genes as one group. We name this network-based SVM as neighboring-gene SVM (NG-SVM).

Consider a gene network with S denoting the set of all edges, i.e., the pair of connected genes.

$$S = \{(j_1, j_2) : \text{gene } j_1 \text{ and gene } j_2 \text{ are connected}\}$$

Define w_k as some weight for gene k. For example, $w_k = \sqrt{d_k}$ where d_k is the number of

direct neighbors of gene k, or $w_k = d_k$, or simply $w_k = 1$ for all genes. We propose an NG penalty in the form of

$$\sum_{(j_1,j_2)\in S} \max\left\{\frac{|\beta_{j_1}|}{w_{j_1}}, \frac{|\beta_{j_2}|}{w_{j_2}}\right\} \tag{2.7}$$

Thus the network-based SVM solves the optimization problem as follows.

$$\min_{\beta_0,\beta}\left\{\sum_{i=1}^{N}\left[1-y_i\left(x_i^T\beta+\beta_0\right)\right]_+ + \lambda \sum_{(j_1,j_2)\in S} \max\left(\frac{|\beta_{j_1}|}{w_{j_1}}, \frac{|\beta_{j_2}|}{w_{j_2}}\right)\right\} \tag{2.8}$$

Four properties of the penalty term are noteworthy. First, the regularization is performed at the level of grouped genes with each group containing two neighboring genes in the network. In the case of penalized linear regression, it has been proven that this penalty achieves the goal of eliminating both β_{j_1} and β_{j_2} simultaneously if $(j_1, j_2) \in S$ (Pan *et al.* to appear). The automatic selection of grouped features is due to the singularity of function $\max\{|a|, |b|\}$ (Zou and Yuan 2008). This formulation satisfies our assumption that neighboring genes tend to (or not to) contribute to the same biological process at the same time. Second, the choice of the weight depends on the goal of shrinkage and influences the predictive performance. Consider a network comprised of several subnetworks, each with one regulator and ten target genes. Because of the singularity of function $\max(|a|, |b|)$ at $a = b$, the weighted penalty in the context of penalized regression, encourages $|\beta_{j_1}|/w_{j_1} = |\beta_{j_2}|/w_{j_2}$ (Pan *et al.* to appear). Here we examine three weight functions in particular: $w_k = 1$, $w_k = \sqrt{d_k}$, and $w_k = d_k$, where gene k has d_k direct neighbors. The new method encourages $|\beta_{j_1}| = |\beta_{j_2}|$ if $w_k = 1$, $\frac{|\beta_{j_1}|}{\sqrt{d_{j_1}}} = \frac{|\beta_{j_2}|}{\sqrt{d_{j_2}}}$ if $w_k = \sqrt{d_k}$, and $\frac{|\beta_{j_1}|}{d_{j_1}} = \frac{|\beta_{j_2}|}{d_{j_2}}$ if $w_k = d_k$. Therefore, larger weights (from $w_k = 1$,

$w_k = \sqrt{d_k}$, to $w_k = d_k$) favor genes with more direct neighbors to have larger coefficient estimates; in other words, larger weights relax the shrinkage effect for those regulators, which are known to be biologically more important. Due to this property, the choice of a large weight, as a simple strategy, enables us to alleviate the bias in the coefficient estimates from the penalization method and possibly improve the predictive performance. Our default weight is $w_k = \sqrt{d_k}$. The weight, considered as another tuning parameter, can be determined from cross-validation or an independent validation data set, though we do not consider it here. Third, the penalty term, under certain conditions, tends to encourage a grouping effect, where highly correlated predictors tend to have similar coefficient estimates (Li and Li 2008, Zou and Hastie 2005, Wang *et al.* 2006, Pan *et al.* to appear). Fourth, the penalty is linear, which allows the solution to be found by the linear programming (LP) technique that is computationally convenient (Fang and Puthenpura 1993).

As usual, the fitted classifier is $\hat{f}(x) = x^T\hat{\beta} + \hat{\beta}_0$, and the classification rule is $sign(\hat{f}(x))$. We employ LP to obtain the solutions to (2.8) by

$$\min_{\beta_0^+,\beta_0^-,\beta^+,\beta^-} \left(\sum_{i=1}^{N} \xi_i + \lambda \sum_{(j_1,j_2)\in S} M_{(j_1,j_2)} \right) \tag{2.9}$$

subject to

$$\begin{aligned} & y_i \left(\beta_0^+ - \beta_0^- + x_i^T(\beta^+ - \beta^-)\right) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i \\ & \frac{\beta_j^+}{w_j} + \frac{\beta_j^-}{w_j} \leq M_{(j_1,j_2)}, \quad j = j_1, j_2 \qquad \forall (j_1,j_2) \in S \\ & \beta_j^+ \geq 0, \quad \beta_j^- \geq 0, \quad j = j_1, j_2 \qquad \forall (j_1,j_2) \in S \end{aligned} \tag{2.10}$$

where

$$\xi_i = \left[1 - y_i \left(\sum_{i=1}^{N} x_i^T \beta + \beta_0\right)\right]_+, \qquad i = 1, 2, \ldots, N \tag{2.11}$$

$$M_{(j_1,j_2)} = \max\left\{\frac{|\beta_{j_1}|}{w_{j_1}}, \frac{|\beta_{j_2}|}{w_{j_2}}\right\} \tag{2.12}$$

and $\beta_j = \beta_j^+ - \beta_j^-$, in which β_j^+ and β_j^- denote the positive and negative parts of β_j. The calculation of the new method can be easily implemented by the R package `lpsolve`, so is the computation of the L_1-SVM. The R package `e1071` (with linear kernel) is used to obtain the solution to the STD-SVM.

2.3 Simulation

2.3.1 Low dimensional data setting

We conducted several simulation studies to numerically evaluate the performance of the NG-SVM along with the STD-SVM and L_1-SVM. The simulation setups were similar to those in (Li and Li 2008). We started from a simple network consisting of 5 subnetworks, each having a regulator gene t ($t = 1, \ldots, 5$) that regulated 10 target genes, leading to a total of 55 genes ($p = 55$). We assumed that two out of the five subnetworks were informative; that is, the coefficients of 22 genes were nonzero and thus informative to the outcome, while the remaining 33 noise genes had no effect on the outcome. We generated a simulated data set by the following steps:

- Generate the expression level of regulator gene t, $X_t \sim N(0, 1)$, $t = 1, \ldots, 5$, independently.

- Assume that the expression level of regulator gene t and each of its regulated genes follow a bivariate normal distribution with correlation 0.7. Thus, the expression level of each target gene regulated by gene t, $X_l^{(t)} \sim N(0.7X_t, 0.51)$, $l = 1, \ldots, 10$ and $t = 1, \ldots, 5$.

- Generate the outcome Y from a logistic regression model:

  $\text{Logit}\,(Pr(Y = 1|X)) = X^T\beta + \beta_0$, $\beta_0 = 2$, where X is the vector of the expression levels of all the genes, and coefficient vector $\beta = (\beta_1^{(1)}, \ldots, \beta_{10}^{(1)}, \ldots, \beta_1^{(5)}, \ldots, \beta_{10}^{(5)})$.

Four sets of true coefficients, β's, were specified to reflect four scenarios:

(i) $\beta = (5, \underbrace{\frac{5}{\sqrt{10}}, \cdots, \frac{5}{\sqrt{10}}}_{10}, -5, \underbrace{\frac{-5}{\sqrt{10}}, \cdots, \frac{-5}{\sqrt{10}}}_{10}, 0, \cdots, 0)$.

The effect of one informative subnetwork was the same as the other in magnitude but with an opposite direction.

(ii) $\beta = (5, \underbrace{\frac{5}{\sqrt{10}}, \cdots, \frac{5}{\sqrt{10}}}_{10}, 3, \underbrace{\frac{3}{\sqrt{10}}, \cdots, \frac{3}{\sqrt{10}}}_{10}, 0, \cdots, 0)$.

Both informative subnetworks had positive effects but in different magnitudes.

(iii) $\beta = (5, \underbrace{\frac{5}{\sqrt{10}}, \cdots, \frac{5}{\sqrt{10}}}_{7}, \frac{-5}{\sqrt{10}}, \frac{-5}{\sqrt{10}}, \frac{-5}{\sqrt{10}}, 3, \underbrace{\frac{3}{\sqrt{10}}, \cdots, \frac{3}{\sqrt{10}}}_{7}, \frac{-3}{\sqrt{10}}, \frac{-3}{\sqrt{10}}, \frac{-3}{\sqrt{10}}\, 0, \cdots, 0)$.

Target genes in the same informative subnetworks had both positive and negative effects.

(iv) $\beta =$

$(5, \underbrace{\frac{5}{\sqrt{10}}, \cdots, \frac{5}{\sqrt{10}}}_{6}, \underbrace{\frac{-5}{\sqrt{10}}, \cdots, \frac{-5}{\sqrt{10}}}_{4}, -3, \underbrace{\frac{-3}{\sqrt{10}}, \cdots, \frac{-3}{\sqrt{10}}}_{6}, \underbrace{\frac{3}{\sqrt{10}}, \cdots, \frac{3}{\sqrt{10}}}_{4}, 0, \cdots, 0)$.

It was similar to but more extreme than scenario 3.

Five methods, STD-SVM, L_1-SVM, and NG-SVM with $w_k = 1$, $w_k = \sqrt{d_k}$, and $w_k = d_k$, were compared based on the results averaged over 100 runs under each of the above four scenarios. For each run, 100 observations were simulated as training data to build a classifier (with any given λ), another 100 for tuning the regularization parameter λ, and the last 10,000 as test data. Each predictor was normalized to have mean 0 and standard deviation 1. Given any value of λ, we obtained the coefficient estimates from the training set, then applied the classifier to the tuning set to find the misclassification error rate (the number of misclassifications divided by the sample size). We searched for $\hat{\lambda}$, from a wide range of prespecified values, which produced the smallest error rate. The classifier corresponding to $\hat{\lambda}$ was identified as the fitted classifier $\hat{f}$. Then we applied $\hat{f}$ to the test set and calculated the test error rate. Table 2.1 reports the mean misclassification error rate of the test set and its standard error (SE in parentheses), the standard deviation of the misclassification error rate divided by the square root of the number of runs, for each method over 100 runs under each scenario. To evaluate each method's ability to select informative genes, we examined the false negatives, defined as the number of informative genes whose coefficients were estimated to be zero. In addition, we also considered a smaller sample size: we repeated the entire process with 50 training data points, 50 tuning data points, and again 10,000 test data points.

According to our simulation setups, the correct weight function should be $w = \sqrt{d}$. However, we find that the NG-SVM with $w = d$ overwhelmingly beat all other methods in all the setups. It consistently made the most accurate classifications and missed no

Table 2.1: Low dimensional data setting: simulation results averaged over 100 runs ($p = 55$; 22 informative and 33 noise genes).

		Test Error (SE)		# False Negative (SE)		Model Size (SE)	
Scenario	Method	$n = 50$	$n = 100$	$n = 50$	$n = 100$	$n = 50$	$n = 100$
1	STD	0.122 (0.002)	0.096 (0.001)	0.0 (0.0)	0.0 (0.0)	55.0 (0.0)	55.0 (0.0)
	L_1	0.134 (0.003)	0.094 (0.002)	13.1 (0.2)	10.9 (0.2)	12.3 (0.3)	15.3 (0.3)
	NG ($w = 1$)	0.156 (0.003)	0.105 (0.002)	9.3 (0.2)	2.4 (0.1)	17.0 (0.3)	24.3 (0.4)
	NG ($w = \sqrt{d}$)	0.111 (0.003)	0.068 (0.002)	1.0 (0.1)	0.1 (0.03)	24.7 (0.4)	25.1 (0.4)
	NG ($w = d$)	0.081 (0.002)	0.059 (0.002)	0.0 (0.0)	0.0 (0.0)	28.6 (0.4)	28.2 (0.4)
2	STD	0.121 (0.002)	0.099 (0.001)	0.0 (0.0)	0.0 (0.0)	55.0 (0.0)	55.0 (0.0)
	L_1	0.133 (0.003)	0.096 (0.001)	13.6 (0.2)	11.1 (0.2)	11.4 (0.3)	15.1 (0.3)
	NG ($w = 1$)	0.156 (0.003)	0.105 (0.002)	9.6 (0.2)	3.9 (0.2)	16.3 (0.3)	24.7 (0.4)
	NG ($w = \sqrt{d}$)	0.121 (0.003)	0.075 (0.002)	3.0 (0.2)	0.3 (0.1)	22.3 (0.4)	25.2 (0.4)
	NG ($w = d$)	0.083 (0.002)	0.064 (0.002)	0.0 (0.0)	0.0 (0.0)	28.6 (0.4)	29.0 (0.4)
3	STD	0.162 (0.002)	0.138 (0.001)	0.0 (0.0)	0.0 (0.0)	55.0 (0.0)	55.0 (0.0)
	L_1	0.166 (0.003)	0.131 (0.001)	13.9 (0.2)	11.0 (0.2)	11.2 (0.3)	16.6 (0.3)
	NG ($w = 1$)	0.177 (0.003)	0.140 (0.002)	12.4 (0.2)	7.7 (0.2)	13.5 (0.3)	19.9 (0.4)
	NG ($w = \sqrt{d}$)	0.164 (0.003)	0.127 (0.002)	4.4 (0.2)	1.2 (0.1)	21.5 (0.4)	26.3 (0.4)
	NG ($w = d$)	0.137 (0.003)	0.114 (0.001)	0.4 (0.1)	0.1 (0.03)	29.8 (0.4)	33.2 (0.4)
4	STD	0.189 (0.002)	0.157 (0.002)	0.0 (0.0)	0.0 (0.0)	55.0 (0.0)	55.0 (0.0)
	L_1	0.186 (0.002)	0.155 (0.002)	14.2 (0.2)	10.5 (0.2)	11.5 (0.3)	18.1 (0.3)
	NG ($w = 1$)	0.198 (0.003)	0.160 (0.002)	13.8 (0.2)	8.6 (0.2)	11.8 (0.3)	20.9 (0.4)
	NG ($w = \sqrt{d}$)	0.190 (0.003)	0.147 (0.002)	7.2 (0.2)	1.8 (0.1)	18.8 (0.4)	30.1 (0.4)
	NG ($w = d$)	0.163 (0.002)	0.139 (0.002)	0.2 (0.04)	0.03 (0.02)	32.2 (0.4)	34.8 (0.4)

Table 2.2: Low dimensional data setting: coefficient estimates of selected informative genes from 100 runs ($p = 55$ and $n = 100$).

		L_1		NG ($w = 1$)		NG ($w = \sqrt{d}$)		NG ($w = d$)	
Scenario	β	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	$\beta_1 = 5$	0.53	0.29	0.04	0.04	0.27	0.26	0.67	0.35
	$\beta_1^{(1)} = \frac{5}{\sqrt{10}}$	0.11	0.17	0.14	0.15	0.10	0.10	0.07	0.08
	$\beta_2 = -5$	-0.55	0.30	-0.04	0.05	-0.28	0.32	-0.68	0.35
	$\beta_1^{(2)} = \frac{-5}{\sqrt{10}}$	-0.08	0.15	-0.18	0.15	-0.11	0.09	-0.08	0.08
2	$\beta_1 = 5$	0.76	0.33	0.09	0.06	0.34	0.16	0.91	0.40
	$\beta_1^{(1)} = \frac{5}{\sqrt{10}}$	0.09	0.14	0.20	0.14	0.14	0.11	0.09	0.08
	$\beta_2 = 3$	0.29	0.23	0.01	0.03	0.15	0.10	0.48	0.23
	$\beta_1^{(2)} = \frac{3}{\sqrt{10}}$	0.08	0.12	0.11	0.13	0.07	0.08	0.04	0.04
3	$\beta_1 = 5$	0.51	0.39	0.03	0.07	0.41	0.70	0.95	0.34
	$\beta_1^{(1)} = \frac{5}{\sqrt{10}}$	0.22	0.21	0.24	0.19	0.20	0.17	0.13	0.11
	$\beta_8^{(1)} = \frac{-5}{\sqrt{10}}$	-0.01	0.07	-0.01	0.11	-0.03	0.21	-0.04	0.12
	$\beta_2 = 3$	0.26	0.27	0.01	0.04	0.15	0.30	0.52	0.27
	$\beta_1^{(2)} = \frac{3}{\sqrt{10}}$	0.09	0.13	0.13	0.16	0.12	0.16	0.07	0.11
	$\beta_8^{(2)} = \frac{-3}{\sqrt{10}}$	0.001	0.07	0.004	0.06	0.01	0.05	-0.01	0.07
4	$\beta_1 = 5$	0.40	0.38	0.03	0.06	0.48	0.80	0.97	0.43
	$\beta_1^{(1)} = \frac{5}{\sqrt{10}}$	0.27	0.26	0.32	0.25	0.30	0.23	0.20	0.20
	$\beta_7^{(1)} = \frac{-5}{\sqrt{10}}$	-0.04	0.12	-0.02	0.14	-0.11	0.24	-0.09	0.16
	$\beta_2 = -3$	-0.23	0.29	-0.004	0.01	-0.21	0.45	-0.56	0.30
	$\beta_1^{(2)} = \frac{-3}{\sqrt{10}}$	-0.15	0.20	-0.16	0.19	-0.17	0.19	-0.09	0.13
	$\beta_7^{(2)} = \frac{3}{\sqrt{10}}$	0.03	0.08	-0.002	0.10	0.05	0.18	0.06	0.15

Table 2.3: High dimensional data setting: simulation results averaged over 100 runs ($p = 550$ or $1,100$; 22 informative and either 528 or 1,078 noise genes).

	Test Error (SE)		# False Negative (SE)		Model Size (SE)	
Method	$p = 550$	$p = 1,100$	$p = 550$	$p = 1,100$	$p = 550$	$p = 1,100$
STD	0.305 (0.003)	0.354 (0.002)	0.0 (0.0)	0.0 (0.0)	550 (0.0)	1,100 (0.0)
L_1	0.218 (0.004)	0.235 (0.004)	16.6 (0.2)	17.1 (0.2)	16.1 (0.4)	19.2 (0.4)
NG ($w = 1$)	0.232 (0.003)	0.255 (0.004)	14.9 (0.2)	15.6 (0.2)	20.7 (0.4)	22.6 (0.5)
NG ($w = \sqrt{d}$)	0.202 (0.004)	0.221 (0.004)	5.7 (0.2)	6.7 (0.2)	32.6 (0.6)	34.6 (0.6)
NG ($w = d$)	0.170 (0.003)	0.180 (0.004)	0.7 (0.1)	1.3 (0.1)	82.6 (0.8)	98.9 (0.9)

informative genes. The NG-SVM with $w = \sqrt{d}$ performed the second best: in most cases, it improved the classification accuracy over STD-SVM and L_1-SVM; and under all the settings, it produced models that identified more informative genes than the L_1-SVM. In contrast, $w = 1$ did not bring much gains over the STD-SVM or the L_1-SVM. The L_1-SVM led to models that were too sparse, missing about 14 and 11 informative genes for $n = 50$ and $n = 100$ respectively. The superior performance and the larger model size of the large weight ($w = d$) compared with its counterparts ($w = 1$ and $w = \sqrt{d}$) is presumably due to its relaxation of the shrinkage effect. The penalization methods shrink the $\hat{\beta}$ toward zero by imposing the constraints (the penalty term) and therefore introduces bias to $\hat{\beta}$. By grouping neighboring genes, the NG-SVM encourages the pairwise weighted absolute coefficients to be equal. Therefore, a larger weight leads to larger $|\hat{\beta}|$ for regulator genes. By choosing a larger weight, we may overcome over-

shrinkage, alleviate biases, and achieve better classification accuracy to some extent at the expense of model sparsity. As shown by Table 2.2, $w = d$ produced the largest $|\hat{\beta}|$ for regulators than its two counterparts. The L_1-SVM estimates were treated as a yardstick for comparison as to provide an idea of the extent of shrinkage by each weight function. For example, $w = 1$ and $w = \sqrt{d}$ overly shrank all the regulators under all scenarios as compared with the L_1-SVM estimates. Note that the binary outcome Y was generated from a logistic regression model while $\hat{\beta}$ was estimated from a linear model, hence $E(\hat{\beta})$ may be different from β even for an unbiased estimator $\hat{\beta}$ of the linear model.

2.3.2 High dimensional data setting

Next, we evaluated the performance of the NG-SVM for high-dimensional data with large p. We used the setup of 50 observations for training, 50 for tuning, and 10,000 for test data. We assumed that (1) the network was composed of either 50 or 100 subnetworks, each having one gene regulating 10 target genes; (2) the first 2 subnetworks were informative resulting in 22 informative genes; (3) the rest of the genes had no effect on the outcome, leading to 528 noise genes when $p = 550$ and 1,078 noise genes when $p = 1,100$; and (4) the true β was specified as scenario 3 in Section 2.3.1. Table 2.3 shows the simulation results averaged over 100 runs.

Again, we see the gains from using a large weight ($w = d$). It prevailed over all the other methods in making accurate classifications and selecting informative genes. The $w = \sqrt{d}$ ranked the second. However, $w = d$ generated models much larger than those from other methods except STD-SVM. In this case, the performance of $w = 1$ is no

better than L_1-SVM possibly due to over shrinkage of the effects of the regulator genes.

2.4 Applications to microarray data

To evaluate its performance in the real world, we applied the NG-SVM to two microarray gene expression data sets related to the Parkinson's disease (PD) (Gene Expression Omnibus: GSE6613 [http://www.lncbi.nlm.nih.gov/geo/]) and breast cancer metastasis (BC) (Wang *et al.* 2005, Chuang *et al.* 2007) respectively.

2.4.1 Parkinson's disease

The data set includes the Parkinson's disease status and the expression levels of 22,283 genes from 105 patients (50 cases and 55 controls) (Scherzer *et al.* 2007). We used the same network structure as (Li and Li 2008). The network combines 33 Kyoto Encyclopedia of Genes and Genomes (KEGG) regulatory pathways and contains a total of 1,523 genes and 6,865 edges. The data were randomly split into training (40 observations), tuning (20 observations), and test (45 observations) sets. The expression level of each gene was normalized to have mean 0 and standard deviation 1 across samples. The tuning parameter was identified from the tuning set and the performance of the method was evaluated on the test set by the mean misclassification error rate and its standard error averaged over 10 runs. Five methods were compared: STD-SVM, L_1-SVM, NG-SVM with $w = 1$, $w = \sqrt{d}$, and $w = d$. To obtain a final model based on the NG-SVM with $w = \sqrt{d}$, we combined, for each run, the previous tuning and test data as the new tuning

set leading to a sample size as large as 65 observations, on which the misclassification error rates were calculated for wide-ranging values of the tuning parameter. Then after 10 runs, we had an averaged misclassification error rate corresponding to each tuning parameter value. The value that generated the minimal averaged error rate was the one we selected to fit the final model to all the data. Note that the misclassification error rate from the final model was likely to be biased due to the double use of the data for training/tuning and test; the main purpose of fitting the final model was to see the selected genes at the end.

First, we focused on the 1,070 genes that appeared in the network with the largest variations of expression levels (i.e., SD of expression levels across the 105 samples $\geq$ 15). According to the KEGG pathway of Parkinson's disease (KEGG: Parkinson's disease `[http://www.genome.ad.jp/kegg/pathway/hsa/hsa05020.html]`), 20 genes play a role in the disease progression, five of which (*UBE1, PARK2, UBB, SEPT5*, and *SNCAIP*) belong to the 1,070 genes. In addition to the misclassification error rate, we added two additional criteria for method comparison: the number of disease genes identified, and the number of genes identified. Table 2.4 shows that STD-SVM made the most accurate classification, even though the difference with other methods was perhaps non-significant. The $w = d$ ranked the second in predictive performance while produced a model including 70.6 genes on average. In this case, the $w = \sqrt{d}$ gained advantage: it selected more disease genes by a relatively sparse model with a misclassification error non-significantly larger than STD-SVM. From the 1,070 genes, with the final model the

Table 2.4: PD data with 1,070 genes in the network. Missclassification error rate, number of selected disease genes, number of selected genes, and their standard errors (SE in parentheses) obtained by averaging over 10 runs. Five disease genes were *UBE1, PARK2, UBB, SEPT5*, and *SNCAIP*.

Method	Error	# Disease Genes	# Genes
STD	0.424 (0.016)	5.0 (0.0)	1,070.0 (0.0)
L_1	0.464 (0.021)	0.1 (0.1)	19.2 (1.4)
NG ($w = 1$)	0.476 (0.015)	0.1 (0.1)	24.9 (1.6)
NG ($w = \sqrt{d}$)	0.480 (0.026)	0.2 (0.1)	30.6 (1.7)
NG ($w = d$)	0.451 (0.028)	0.0 (0.0)	70.6 (2.6)
Final Model	-	1.0	75.0

NG-SVM identified 75 genes including one disease gene.

Next, to better integrate the biological observation of the KEGG pathway and the known network structure of Li and Li (2008), we restricted our analysis to the first- and second-order-neighbors of the 8 disease genes on the Parkinson's disease KEGG pathway whose expression levels and network structure are available. The first-order-neighbor subnetwork (PD-1nb-net) was composed of the 8 disease genes and their 8 direct neighbors. The second-order-neighbor subnetwork (PD-2nb-net) comprised the PD-1nb-net as well as the direct neighbors of the 8 direct neighbors of the disease genes, leading to a total of 26 genes. Figure 2.1 displays the two subnetworks. We conducted the analysis in the same way as described above. The only difference resided in that

Table 2.5: PD-1nb-net/PD-2nb-net. Misclassification error rate, number of selected disease genes, number of selected genes, and their standard errors (SE in parentheses) obtained by averaging over 10 runs. Eight disease genes were *UBE1, PARK2, UBB, SEPT5, SNCAIP, GPR37, TH*, and *SNCA*.

Network	Method	Error	# Disease Genes	# Genes
PD-1nb-net	STD	0.476 (0.023)	8.0 (0.0)	16.0 (0.0)
	L_1	0.471 (0.017)	2.8 (0.4)	6.1 (0.6)
	NG ($w = 1$)	0.462 (0.016)	3.4 (0.4)	7.3 (0.6)
	NG ($w = \sqrt{d}$)	0.462 (0.014)	3.6 (0.4)	8.4 (0.6)
	NG ($w = d$)	0.482 (0.015)	3.0 (0.4)	7.5 (0.6)
	Final Model	-	8.0	16.0
PD-2nb-net	STD	0.444 (0.016)	8.0 (0.0)	26.0 (0.0)
	L_1	0.449 (0.017)	3.1 (0.4)	10.9 (0.8)
	NG ($w = 1$)	0.464 (0.022)	5.3 (0.4)	13.2 (0.8)
	NG ($w = \sqrt{d}$)	0.447 (0.023)	6.1 (0.4)	13.7 (0.8)
	NG ($w = d$)	0.433 (0.016)	6.2 (0.4)	20.0 (0.7)
	Final Model	-	8.0	26.0

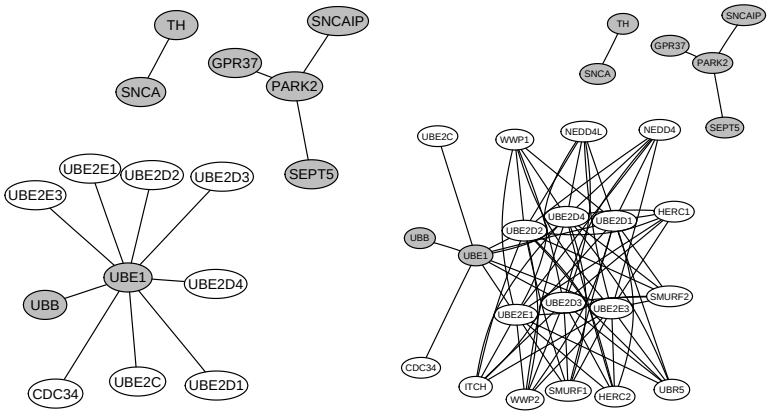

Figure 2.1: PD subnetworks. Left: PD-1nb-net, including 8 Parkinson disease genes (gray) and their 8 direct neighbors (white). Right: PD-2nb-net, including 8 Parkinson disease genes (gray), their 8 direct and 10 second-order neighbors (white).

this time only genes appearing in the PD-1nb-net and PD-2nb-net were included in the analysis. Table 2.5 shows the results.

We see the gains from employing the NG-SVM when narrowing down our focus on the PD-1nb-net and PD-2nb-net. For the PD-1nb-net, $w = 1$ and $w = \sqrt{d}$ performed equally well. They had the smallest misclassification error rate and identified one more disease gene through a model slightly larger than the one obtained from L_1-SVM. The NG-SVM with $w = d$ won over in the case of PD-2nb-net with the best accuracy and most selected disease genes. The $w = \sqrt{d}$ ranked the second in terms of the prediction accuracy while detecting 3 more disease genes by a model with 3 more genes than that of the L_1-SVM. This means that the NG-SVM was able to identify more clinically relevant

genes while keeping the same number of noise genes in the model as L_1-SVM. In both subnetworks, the final models included all the genes.

2.4.2 Breast cancer metastasis

The breast cancer metastasis data set (Wang *et al.* 2005, Chuang *et al.* 2007) contains expression levels of 8,141 genes for 286 patients, 106 of whom were detected to develop metastasis within a 5-year follow-up after surgery. *TP53*, *BRCA1*, and *BRCA2* are three human genes that belong to the class of tumor suppressor genes, which are known to prevent uncontrolled cell proliferation, and to play a critical role in repairing the chromosomal damage. Certain mutations of these genes lead to increasing risk of breast cancer. We explored the protein-protein interaction (PPI) network previously used by (Chuang *et al.* 2007). The PPI network comprises 57,235 interactions among 11,203 proteins, obtained by assembling various sources of experimental data and curation of the literature (Chuang *et al.* 2007). We confined our analysis to the direct or first-order neighbors (BC-1nb-net) of the three cancer genes, and the subnetwork composed of two parts (BC-2nb-net): the direct neighbors of *TP53*, and the second-order neighbors of *BRCA1* and *BRCA2*. We fit the final model and compared the four methods in terms of misclassification error rate, cancer genes selection, and model sparsity. The cancer genes are the 227 known or putative cancer genes with estimated mutation frequencies in cancer samples (Chuang *et al.* 2007). A total of 294 genes that fell into the BC-1nb-net had observed expression levels, among which were 40 cancer genes and 7 cancer genes (*ABL1, JAK2, p53, PTEN, p14ARF, PTCH*, and *RB*) with mutation frequencies larger

Table 2.6: BC-1nb-net/BC-2nb-net: 294/1,718 genes in total including 40/107 cancer genes, and 7/14 cancer genes with mutation frequencies larger than 0.10. Misclassification error rate, number of selected cancer genes with mutation frequencies larger than 0.10 (CA-LMF), number of selected cancer genes (CA), number of selected genes, and their standard errors (SE in parentheses) obtained by averaging over 10 runs.

Network	Method	Error	# CA-LMF	# CA	# Genes
BC-1nb-net	STD	0.371 (0.014)	7.0 (0.0)	40.0 (0.0)	294.0 (0.0)
	L_1	0.357 (0.014)	0.3 (0.2)	4.6 (0.6)	32.3 (1.7)
	NG ($w = 1$)	0.360 (0.014)	0.4 (0.2)	3.6 (0.6)	25.0 (1.5)
	NG ($w = \sqrt{d}$)	0.366 (0.012)	0.6 (0.2)	4.7 (0.6)	27.2 (1.6)
	NG ($w = d$)	0.399 (0.012)	1.2 (0.3)	7.8 (0.8)	40.2 (1.9)
	Final Model	-	1.0	4.0	14.0
BC-2nb-net	STD	0.351 (0.014)	14.0 (0.0)	107.0 (0.0)	1,718.0 (0.0)
	L_1	0.360 (0.006)	0.0 (0.0)	2.4 (0.5)	42.9 (2.0)
	NG ($w = 1$)	0.374 (0.011)	0.1 (0.1)	1.9 (0.4)	51.4 (2.2)
	NG ($w = \sqrt{d}$)	0.360 (0.007)	0.2 (0.1)	2.5 (0.5)	41.7 (2.0)
	NG ($w = d$)	0.385 (0.021)	0.3 (0.2)	0.7 (0.3)	34.2 (1.8)
	Final Model	-	1.0	2.0	23.0

than 0.10. The BC-2nb-net was composed of 2,070 genes, 1,718 of them with observed expression levels, including 107 cancer genes. Besides the 7 included in BC-1nb-net, 7 additional cancer genes (*ACH, APC, EGFR, KIT, NICD, RAS*, and *CTNNB1*) that had mutation frequencies larger than 0.10 belonged to BC-2nb-net.

For BC-1nb-net, $w = d$ had the advantage in selecting cancer genes and those with large mutant frequencies (Table 2.6). The NG-SVM with $w = \sqrt{d}$ detected more clinically relevant genes by a sparser model while reaching a comparable misclassification error rate to that of L_1-SVM. Even though the final model was parsimonious, it included 4 cancer genes, one of which had a large mutation frequency. For BC-2nb-net, the NG-SVM with $w = \sqrt{d}$ detected more cancer genes with equally accurate predictions while maintaining a sparse model compared with L_1-SVM. The final model included only 23 genes out of 1,718, two of which were cancer genes with one having a large mutation frequency.

2.5 Conclusion

The advancement in the microarray technology has enriched the tool kit of researchers to decipher the complexity of disease mechanisms at the genomic level. Studies have been widely conducted to identify genetic markers to better the diagnostic classification and prognostic assessment, largely by ignoring biological knowledge on gene functions and treating individual genes equally and independently a priori. The downside of such an endeavor has been realized; for example, gene markers identified across similar pa-

tient cohorts for the same disease in such a way often lack consistency. As a viable alternative, the network-based approach has been gaining popularity. In addition to improving predictive performance and gene selection, the network-based approach extracts more biological insights from high-throughput gene expression data. In this chapter, we have proposed a network-based SVM, NG-SVM, with a penalty term incorporating gene network information, as a practically useful classification tool for microarray data. Our simulation studies and two real data applications indicate that the proposed method is able to better identify clinically relevant genes and make accurate predictions.

Chapter 3

Support Vector Machine with a Disease-gene-centric Penalty

3.1 Introduction

With the availability of various repositories of gene networks and the accumulating knowledge on genes linked to diseases, one question naturally arises: how to integrate the two sources of prior information into a model to detect genes involved in disease-related biological processes. A network-based approach is based on such a coherent view and makes use of the network information in building statistical models. Employing a network-based perspective not only sheds insight within the network modules (Benson *et al.* 2006, Calvano *et al.* 2005, Chuang *et al.* 2007, Liu *et al.* 2007) but also permits identification of disease genes that have only weak effects. Such genes often play a central role in discriminative subnetworks by interconnecting groups of genes involved in various biological processes. Chuang *et al.* (2007) pointed out that several well-known cancer genes, such as *TP53*, *KRAS*, and *HRAS*, were ignored by gene-expression-alone analysis but successfully detected by using network information. However, their network-based approach involves a random search over subnetworks, leading to possibly instable and suboptimal final results.

Since its invention (Cortes and Vapnik 1995, Vapnik 1995), the support vector machine (SVM) has been acclaimed as a useful regularization method due to its excellent empirical performance, especially for high dimensional data (Brown *et al.* 2000, Furey *et al.* 2000), its possible extensions to accommodate various penalty functions, and resulting model sparsity if a suitable penalty (e.g. L_1-norm) is employed. For binary classification, the standard L_2-norm SVM (STD-SVM) has good predictive performance, but is inca-

pable of performing variable selection. The L_1-SVM (Wang and Shen 2007, Zhu *et al.* 2003) produces sparse models for data with $p >> N$. Zou and Yuan (2008) developed a grouped variable selection scheme for factors by the use of an F_∞-norm SVM such that all features derived from the same factor (i.e. categorical predictor) are included or excluded simultaneously. Note that their grouping scheme was based on non-overlapping groups. Zhao *et al.* (to appear) generalized grouped variable selection and introduced the composite absolute penalties (CAP) family. CAP achieves both grouped selection for non-overlapping groups and hierarchical selection for overlapping groups. Extending the idea of grouping to gene networks, Zhu *et al.* (2009) proposed a neighboring-gene SVM (NG-SVM), treating any two neighboring genes in a network as one group, and explicitly incorporating the network information into building classifiers. Both the simulation studies and real data applications showed that NG-SVM enjoyed advantages in gene selection and predictive performance compared with the popular STD-SVM and L_1-SVM. However, a potential problem of NG-SVM resides in its tendency of selecting isolated genes or gene pairs, i.e., genes largely disconnected to the rest in the network, which is not desirable given that some disease genes cluster together and form subnetworks.

In this chapter, we incorporate the information of both a gene network and some crucial disease genes into the SVM framework by exploiting two ways of grouping genes for penalty construction. By considering an undirected network to be anchored on certain crucial disease gene(s), i.e., genes known to be central to a disease, a hierarchical structure is imposed on the network (with the anchoring crucial genes at the top) to facilitate the

definition of various gene groups. By summing up an L_∞-norm over each group, we obtain the penalty for DGC-SVM. Ideally, by DGC-SVM, identification of one gene triggers the inclusion of disease genes along the connected paths towards the top crucial gene(s). In particular, we intend to capture disease genes, even if their direct effects on the outcome are weak, which are important in regulating functional activities of other genes along the pathways or within the subnetworks involved in the disease.

The later sections are organized as follows. First, we discuss an orientation scheme that converts an undirected gene network into an upper-lower hierarchy on which groups of genes are defined. Next, we introduce a second network-based SVM, the disease-gene-centric SVM. Then we evaluate the performance of the proposed method through simulation studies and real data applications. We conclude this chapter with a brief discussion of the pros and cons of this new method.

3.2 Methods

3.2.1 Orienting an undirected network

Given an undirected network G, we convert it into a directed acyclic graph (DAG) $\tilde{G}$. Suppose that G originates from only one disease gene g and consists of a total of p genes. Genes (including g) in network G are indexed by $\{1, 2, \ldots, p\}$. We have the expression levels of the p genes and binary outcomes for N samples, $\{(x_i, y_i)\}_{i=1}^N$ with $x_i \in \mathbb{R}^p$ and $y_i \in \{1, -1\}$. The expression of each gene is normalized to have mean 0 and standard deviation 1 across samples. We define a directed edge by an ordered pair of ends (a, b)

indicating that a is upstream to b, or equivalently, b is downstream to a. Since genetic interrelationships occur only between pairs of distinct genes, network G contains no loop, defined as a directed edge with identical ends. In addition, no two directed edges adjoin the same pair of genes. Gene g is the top (center) gene of network G. The distance between two genes a and b is the minimum number of directed edges traversed from a to b. Genes closer to the network origin, gene g, are said to be at an upper level than those farther apart. Genes with the same distance from the origin are at the same level. For example, the distance between gene g and any of its direct neighbors is 1. The distance between any two genes at the same level is 0. Thus, DAG $\tilde{G}$ is defined from the undirected network G. $\tilde{G}$ assigns directions from upper-level to lower-level genes but ignores edges connecting genes at the same level. Upper-level genes are called nodes whereas genes with no downstream genes are named as leaves. DAG $\tilde{G}$ captures the upper-lower interrelationships but ignoring the lateral ones.

In presence of multiple center genes $g_1 \ldots g_L$, DAG $\tilde{G}$ can be defined as follows: (1) Derive DAGs, $\tilde{G_1} \ldots \tilde{G_L}$, each corresponding to one center gene in $g_1 \ldots g_L$; (2) $\tilde{G} = \cup_{l=1}^{L} \tilde{G_l}$ if $\tilde{G_1} \ldots \tilde{G_L}$ share no common nodes; (3) if the DAGs have common nodes, pick up any of them, align all the associated DAGs at the level where that common node is seated, treat that node in each associated DAG as being located at the same level (named as $level_v$), and merge the associated DAGs by recognizing only the upper-lower interrelationships but ignoring the lateral ones. Then, identify the common nodes of the merged DAG and the remaining untouched DAGs, and repeat step (3) until no common

nodes exist. Note that each node in the merged DAG has the same downstream genes no matter which center the node is derived from. The above process may result in different DAGs if the combination of the associated DAGs occurs at different common nodes, introducing certain arbitrariness.

3.2.2 Pathway grouping

To achieve our goal of detecting collectives of genes involved in disease along pathways or within subnetworks, we propose a penalty on suitably defined groups of genes. We experiment two ways of grouping: pathway (PW) grouping and partial tree (PT) grouping. We first describe the PW grouping. It forms groups along linear paths as an attempt to encourage linear pathway selection.

A path in $\tilde{G}$ is a connected sequence of directed edges and the length of the path is the number of directed edges traversed. Note that a path connects genes from upper- to lower-levels without any two consecutive genes from the same level. Since a path can be determined by the sequence of the nodes along the path, a path is simply specified by its node sequence. We define a single node as a trivial path. Define a complete path of leaf k in $\tilde{G}$, $\mathcal{E}_k$ $(k = 1, \ldots, K)$, as

$$\mathcal{E}_k = \{j : \text{ Gene } j \text{ appears on the path from top gene } g \text{ down to leaf } k\}.$$

Suppose $\mathcal{E}_k$ contains a total of n_k genes, including leaf k and gene g. Then we have n_k groups, $\mathcal{G}_t^{(k)}$ $(t = 1, \ldots, n_k)$, by grouping the genes in $\mathcal{E}_k$ under the "lower nested within upper" rule, that is, node/leaf at a lower level must appear in all the groups that contain

any node at its upper-level. For example, in the network displayed in Figure 3.1, if gene 1 is considered to be at the top, then genes $1, \ldots, 12$ are nodes and genes $13, \ldots, 26$ are leaves. The complete path of leaf gene 16, $\mathcal{E}_{16}$, is $\{1, 2, 6, 16\}$. Groups derived from $\mathcal{E}_{16}$ or leaf gene 16 are $\{2, 6, 16\}$, $\{6, 16\}$, $\{16\}$, and $\mathcal{E}_{16}$ itself. Note that multiple distinct complete paths may exist between leaf k and gene g, for example, $\{g, a, c, k\}$ and $\{g, b, c, k\}$. In this case, group $\{c, k\}$ and group $\{k\}$ are defined twice respectively. When forming groups, we count each distinct group only once. Therefore, groups formed from $\{g, a, c, k\}$ and $\{g, b, c, k\}$ include 6 groups: $\{g, a, c, k\}$, $\{g, b, c, k\}$, $\{a, c, k\}$, $\{b, c, k\}$, $\{c, k\}$, and $\{k\}$. Thus, we impose a grouping structure $\mathcal{G}$ containing distinct groups of $\tilde{G}$, that is, every group in $\mathcal{G}$ appears only once:

$$\mathcal{G} = (\mathcal{G}_1^{(1)}, \ldots, \mathcal{G}_{n_1}^{(1)}, \ldots, \mathcal{G}_1^{(K)}, \ldots, \mathcal{G}_{n_K}^{(K)}),$$

while a gene may appear in multiple groups, which is permitted in our formulation and computation.

Corresponding to $\mathcal{G}$, we construct our penalty as

$$\left(\sum_{k=1}^{K} \sum_{t=1}^{n_k} \|\beta_{\mathcal{G}_t^{(k)}}\|_\infty \right). \tag{3.1}$$

The hinge loss penalized by (3.1) leads to our proposed DGC-SVM with PW grouping (DGC-SVM-PW), which is developed as an attempt to encourage selecting genes along the pathway (pathway selection):

$$\min_{\beta_0, \beta} \left\{ \sum_{i=1}^{N} \left[1 - y_i \left(x_i^T \beta + \beta_0\right)\right]_+ + \lambda \left(\sum_{k=1}^{K} \sum_{t=1}^{n_k} \|\beta_{\mathcal{G}_t^{(k)}}\|_\infty \right) \right\}, \tag{3.2}$$

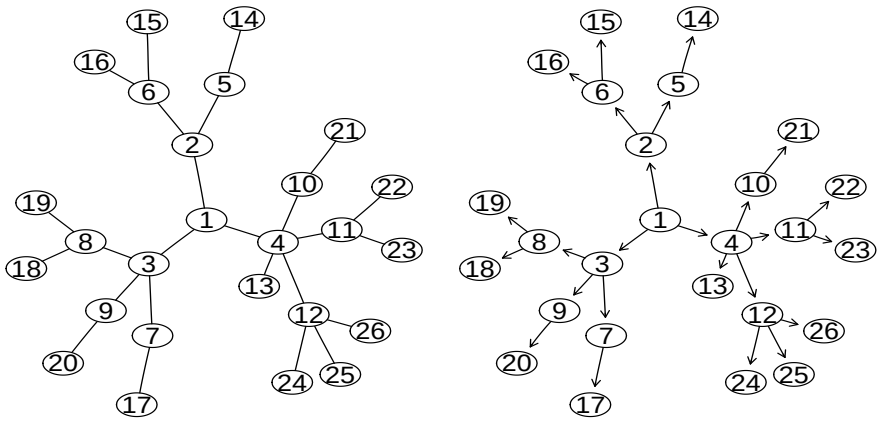

Figure 3.1: Left: simple network originating from gene 1. Right: DAG derived from the simple network.

where the subscript "+" denotes the positive part, i.e., $z_+ = max\{z, 0\}$, λ is a tuning parameter, and $\|\beta_{\mathcal{G}_t^{(k)}}\|_\infty = \max_{s\in\mathcal{G}_t^{(k)}}\{|\beta_s|/w_s\}$ with w_s as a weight function for gene s. For example, w_s can be $\sqrt{d_s}$ with d_s as the number of direct neighbors of gene s, or d_s, or simply 1 for all genes. The solution to (3.2) can be obtained through linear programming:

$$\min_{\beta_0^+,\beta_0^-,\beta^+,\beta^-}\left(\sum_{i=1}^{N}\xi_i + \lambda\sum_{k=1}^{K}\sum_{t=1}^{n_k} M_{\mathcal{G}_t^{(k)}}\right) \tag{3.3}$$

subject to

$$\begin{aligned}
&y_i\left(\beta_0^+ - \beta_0^- + x_i^T(\beta^+ - \beta^-)\right) \geq 1 - \xi_i,\\
&\xi_i = \left[1 - y_i\left(x_i^T\beta + \beta_0\right)\right]_+ \geq 0, \quad \forall i,\\
&\frac{\beta_s^+}{w_s} + \frac{\beta_s^-}{w_s} \leq M_{\mathcal{G}_t^{(k)}}, \quad \beta_s^+ \geq 0, \quad \beta_s^- \geq 0,\\
&s \in \mathcal{G}_t^{(k)}, \quad k = 1,\ldots,K, \quad t = 1,\ldots,n_k.
\end{aligned} \tag{3.4}$$

In the above parametrization in (3.4), $M_{\mathcal{G}_t^{(k)}} = \max_{s \in \mathcal{G}_t^{(k)}} \{|\beta_s|/w_s\}$, and $\beta_s = \beta_s^+ - \beta_s^-$, in which β_s^+ and β_s^- denote the positive and negative parts of β_s. Note that, by our construction, some genes fall within multiple groups $\mathcal{G}_t^{(k)}$, which is permitted by linear programming.

3.2.3 Partial tree grouping

The PT grouping is devised to achieve hierarchical selection, that is, selecting a lower-level gene ensures the selection of its upper-level gene(s). In addition, selecting any gene in the DAG guarantees the inclusion of at least one center gene, which is desirable in view of the biological importance of any center gene. The DGC-SVM with PT grouping (DGC-SVM-PT) groups each node/leaf with all its downstream genes. Since a leaf has no downstream genes, the group derived from the leaf contains only one element, the leaf itself. For the above $\tilde{G}$, we have p groups in total, K of which contain only single elements derived from K leaves, and the rest $p - K$ of which are formed as

$$\mathcal{G}_q = \{\text{node } q \text{ and all its downstream genes}, q = 1, \ldots, p - K\}.$$

For example, the simple network in Figure 3.1 derives 26 groups, including $\mathcal{G}_1$ with all the 26 genes as well as 14 single-leaf groups. Here we impose a grouping structure as $\mathcal{G} = (\mathcal{G}_1, \ldots, \mathcal{G}_p)$. The formulation of DGC-SVM-PT is the same as its PW grouping counterpart in (3.2)-(3.4).

The DGC-SVM-PT is a direct application of the CAP family of Zhao *et al.* (to appear) in the context of SVM. It has the hierarchical property that if any node/leaf

at a lower-level is included in the model, nodes at its upper-level will also be included. This property is important to our goal of capturing disease genes along pathways or within subnetworks, which offers the possibility to detect genes that have weak direct effect on the outcome but are critical in regulating multiple biological processes through connecting various functional groups of disease-relevant genes. In addition, this property guarantees the identification of a center gene of the network if any gene in the network is selected.

3.2.4 Choice of weight

DGC-SVM involves a weight function w. The choice of the weight depends on the goal of shrinkage, and governs variable selection and predictive performance.

A main motivation behind the proposed penalties is the *grouping* effect of the L_∞-norm. Because of the singularity of the penalty $\max(|a|, |b|)$ at $|a| = |b|$, by Fan and Li (2001), the penalty encourages the shrinkage towards $|a| = |b|$, which can be achieved when the penalization parameter λ is large enough. For linear regression, this so-called grouping effect has been theoretically established by Bondell and Reich (2008), and Pan *et al.* (to appear) for two-gene groups, and by Wu *et al.* (to appear) for a more general case with more than two genes in a group. Now consider network G and its grouping structure $\mathcal{G}$ derived from $\tilde{G}$. For simplicity, we assume that $\mathcal{G}$ contains only two-gene and one-gene groups. For these two-gene groups, the weighted penalty encourages $|\beta_{j_1}|/w_{j_1} = |\beta_{j_2}|/w_{j_2}$ where β_{j_1} and β_{j_2} belong to the same group. Here we examine three weight functions specifically: $w_s = 1$, $w_s = \sqrt{d_s}$, and $w_s = d_s$, where d_s is the degree

of gene s, i.e., the number of direct neighbors of gene s. The new method encourages $|\beta_{j_1}| = |\beta_{j_2}|$ if $w_s = 1$, $\frac{|\beta_{j_1}|}{\sqrt{d_{j_1}}} = \frac{|\beta_{j_2}|}{\sqrt{d_{j_2}}}$ if $w_s = \sqrt{d_s}$, and $\frac{|\beta_{j_1}|}{d_{j_1}} = \frac{|\beta_{j_2}|}{d_{j_2}}$ if $w_s = d_s$. The same reasoning also applies to groups with more than two genes. Therefore, larger weights (from $w_s = 1$, $w_s = \sqrt{d_s}$, to $w_s = d_s$) favor genes with more direct neighbors to have larger coefficient estimates; in other words, larger weights relax the shrinkage effect for those "hub" genes that are connected to many genes and are known to be biologically more important. Because of this property, the choice of a large weight, as a simple strategy, enables us to alleviate the bias in the coefficient estimates from penalization and possibly improve predictive performance. The weight can be considered as a tuning parameter and estimated by cross-validation or an independent tuning data set although we will not pursue it further here. Since the proposed penalty is linear, linear programming is applicable to solve the associated optimization problem. We implemented the proposed method through a linear programming routine `lpsolve` in R.

3.3 Simulation

We numerically evaluated the new methods, DGC-SVM-PW and DGC-SVM-PT, in two simulation studies over a simple network and a more complex one. The DAG for a simple network is essentially a hierarchical tree where any two genes are connected by a unique path. In contrast, there exist multiple paths adjoining the same pair of genes in the complicated network. The grouping structure in either case is unique. We compare the performance of DGC-SVMs with that of STD-SVM, L_1-SVM, and NG-SVM. All

methods were implemented by the R package `lpsolve` except STD-SVM, which was obtained by the R package `e1071` (with linear kernel).

3.3.1 A simple network

We applied the DGC-SVM to the simple network depicted in Figure 3.1, where any two genes in its DAG are connected by a unique path. The simulation data sets were generated following the set-ups of Li and Li (2008):

- Generate the expression level of center gene 1, $X_1 \sim N(0, 1)$.
- Assume node s and each of its downstream genes follow a bivariate normal distribution with means 0 and unit variances with correlation 0.7. Thus, the expression level of each downstream gene is distributed as $N(0.7X_s, 0.51)$.
- Generate outcome Y from a logistic regression model:
  $\text{Logit}\left(Pr(Y = 1|X)\right) = X^T\beta + \beta_0$, $\beta_0 = 2$, where X is a vector of the expression levels of all the genes, and β is the corresponding coefficient vector.

We considered three sets of informative genes. The effect of each informative gene on the outcome was equal to that of its upstream node divided by the square-root of the upstream node's degree. All the other genes were noninformative, which had no effect on the outcome. Three sets of true coefficients $\beta = (\beta_1, \beta_2, \ldots, \beta_{26})$ were specified in three scenarios:

(i) PT setting: one of the tree branches of the hierarchical tree or DAG (gene 1, 2, 5, 6, 14, 15, and 16) was informative.

$$\beta = (5, \beta_1/\sqrt{3}, 0, 0, \beta_2/\sqrt{3}, \beta_2/\sqrt{3}, \underbrace{0, \cdots, 0}_{7}, \beta_5/\sqrt{2},\ \beta_6/\sqrt{3}, \beta_6/\sqrt{3}, \underbrace{0, \cdots, 0}_{10}).$$

(ii) PW setting: pathway $\{1, 3, 7, 17\}$ was informative.

$$\beta = (5, 0, \beta_1/\sqrt{3}, 0, 0, 0, \beta_3/\sqrt{4}, \underbrace{0, \cdots, 0}_{9}, \beta_7/\sqrt{2}, \underbrace{0, \cdots, 0}_{9}).$$

(iii) PW setting: pathway $\{1, 2, 5, 14\}$ was informative.

$$\beta = (5, \beta_1/\sqrt{3}, 0, 0, \beta_2/\sqrt{3}, \underbrace{0, \cdots, 0}_{8}, \beta_5/\sqrt{2}, \underbrace{0, \cdots, 0}_{12}).$$

In each scenario, we simulated 50, 50 and 10,000 observations for each training, tuning and test data set. For each tuning parameter value, we obtained a classifier from the training data, applied it to the tuning data, and identified $\hat{\lambda}$ that yielded the minimal misclassification error rate over the tuning set. Then we used the classifier corresponding to $\hat{\lambda}$ to compute the misclassification error rate on the test data. The entire process was repeated 100 times (i.e., 100 independent runs). The means of the test misclassification error rates, false negatives (the number of informative genes with zero coefficient estimates), model sizes (the number of genes with nonzero coefficient estimates), and their corresponding standard errors ($sd/\sqrt{run}$) are reported in Table 3.1.

Evidently, DGC-SVM-PT generated models as sparse as that obtained from L_1-SVM, and gave the most accurate prediction among all the other methods. In addition, the center gene, gene 1, was detected in each run by DGC-SVM-PT. NG-SVM and DGC-SVM-PW yielded fewer false negatives due to the larger models produced by each method. The weight function $w = d$ improved the classification accuracy, slightly shrank the model size, and kept almost the same false negatives for NG-SVM and DGC-SVM-PW compared with the other two weight functions. In contrast, $w = 1$ worked better

Table 3.1: Simple network: simulation results averaged over 100 runs ($p = 26$; 7, 4, and 4 informative genes in the three scenarios respectively).

Scenario	Method	Test Error (SE)	# False Negative (SE)	Model Size (SE)
1	STD	0.129 (0.003)	0.00 (0.00)	26.00 (0.00)
	L_1	0.122 (0.003)	2.81 (0.13)	7.19 (0.23)
	NG ($w = 1$)	0.145 (0.004)	0.26 (0.05)	14.73 (0.25)
	NG ($w = \sqrt{d}$)	0.123 (0.004)	0.10 (0.03)	15.32 (0.25)
	NG ($w = d$)	0.108 (0.003)	0.12 (0.03)	14.47 (0.25)
	PW ($w = 1$)	0.152 (0.003)	0.84 (0.09)	15.57 (0.25)
	PW ($w = \sqrt{d}$)	0.136 (0.003)	0.93 (0.09)	16.90 (0.24)
	PW ($w = d$)	0.126 (0.003)	1.60 (0.11)	14.44 (0.25)
	PT ($w = 1$)	0.107 (0.003)	1.94 (0.12)	9.42 (0.25)
	PT ($w = \sqrt{d}$)	0.107 (0.004)	2.51 (0.13)	8.65 (0.24)
	PT ($w = d$)	0.110 (0.004)	3.08 (0.13)	7.39 (0.23)
2	STD	0.147 (0.003)	0.00 (0.00)	26.00 (0.00)
	L_1	0.118 (0.004)	0.99 (0.09)	6.56 (0.22)
	NG ($w = 1$)	0.162 (0.004)	0.16 (0.04)	17.83 (0.24)
	NG ($w = \sqrt{d}$)	0.138 (0.003)	0.01 (0.01)	17.33 (0.24)
	NG ($w = d$)	0.125 (0.003)	0.04 (0.02)	16.09 (0.25)
	PW ($w = 1$)	0.174 (0.003)	0.32 (0.05)	18.93 (0.23)
	PW ($w = \sqrt{d}$)	0.163 (0.003)	0.42 (0.06)	16.31 (0.25)
	PW ($w = d$)	0.144 (0.003)	0.43 (0.06)	14.94 (0.25)
	PT ($w = 1$)	0.111 (0.003)	0.72 (0.08)	7.74 (0.23)
	PT ($w = \sqrt{d}$)	0.109 (0.003)	1.05 (0.09)	6.64 (0.22)
	PT ($w = d$)	0.113 (0.003)	1.32 (0.09)	6.07 (0.22)
3	STD	0.143 (0.002)	0.00 (0.00)	26.00 (0.00)
	L_1	0.120 (0.003)	0.96 (0.09)	6.67 (0.22)
	NG ($w = 1$)	0.149 (0.003)	0.09 (0.03)	16.04 (0.25)
	NG ($w = \sqrt{d}$)	0.127 (0.003)	0.02 (0.01)	15.50 (0.25)
	NG ($w = d$)	0.117 (0.003)	0.06 (0.02)	13.25 (0.25)
	PW ($w = 1$)	0.165 (0.002)	0.33 (0.06)	17.98 (0.24)
	PW ($w = \sqrt{d}$)	0.147 (0.003)	0.34 (0.06)	16.96 (0.24)
	PW ($w = d$)	0.141 (0.003)	0.41 (0.06)	15.39 (0.25)
	PT ($w = 1$)	0.106 (0.003)	0.59 (0.07)	8.07 (0.24)
	PT ($w = \sqrt{d}$)	0.110 (0.003)	1.10 (0.09)	5.84 (0.21)
	PT ($w = d$)	0.113 (0.003)	1.28 (0.09)	5.71 (0.21)

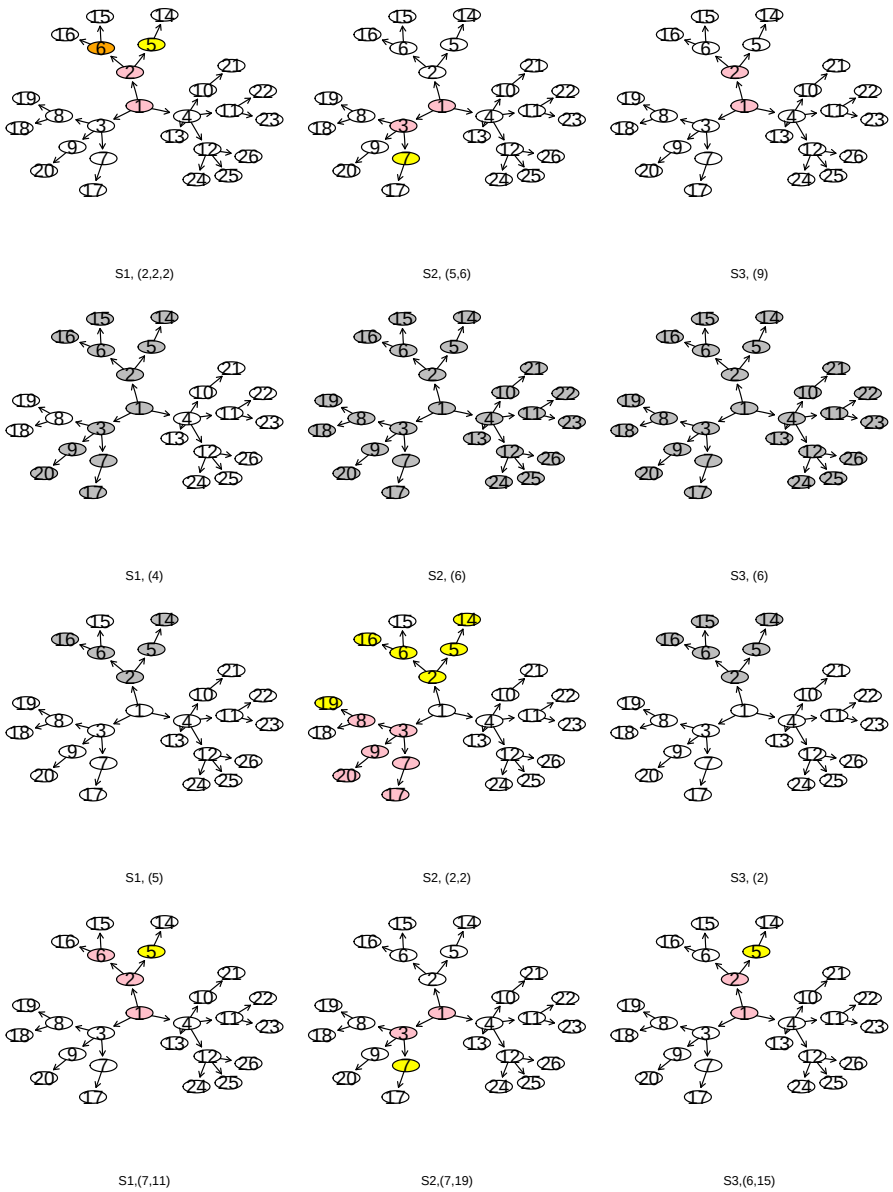

Figure 3.2: Most frequently recovered pathways. Rows from top to bottom: L_1-SVM, NG-SVM, DGC-SVM-PW, and DGC-SVM-PT. Columns from left to right: scenario 1 (S1) $\{1, 2, 5, 6, 14, 15, 16\}$, scenario 2 (S2) $\{1, 3, 7, 17\}$, and scenario 3 (S3) $\{1, 2, 5, 14\}$. Frequencies of the recovered pathways are included in parentheses.

for DGC-SVM-PT. It reduced the false negatives while produced models of comparable predictive performance to that with $w = \sqrt{d}$ or $w = 1$. Therefore, DGC-SVM-PT with $w = 1$ was the winner. In addition, it also improved reproducibility. The most frequently-recovered pathways from each method (L_1-SVM, NG-SVM $w = \sqrt{d}$, DGC-SVM-PW $w = \sqrt{d}$, and DGC-SVM-PT $w = \sqrt{d}$) are displayed in Figure 3.2. Both DGC-SVM-PT and L_1-SVM missed the leaves under each scenario. However, both identified the majority parts of the true pathways. Compared with all the other methods, DGC-SVM-PT detected the same pathway with a much higher frequency. Therefore, the identified pathways by this method were more reproducible.

3.3.2 A complicated network

Next, we explored the complicated network originating from gene 1 as displayed in Figure 3.3. For this network, there exists one pair of genes connected by more than one path. Therefore, the DAG derived from the complicated network does not form a tree. For example, gene 32 has both gene 23 and gene 3 at its upstream. In addition, genes at the same level are connected, such as genes 22 and 23, and genes 33 and 34. By definition, genes with no downstream genes are considered as leaves. Even though gene 22 is connected with gene 23, gene 22 is considered as a leaf because gene 23 is at the same level as gene 22. Likewise, genes 33 and 34 are both treated as leaves. Therefore, unlike the simple network, the DAG defined by the complicated network contains directed edges characterizing upper-lower relationships but ignores undirected edges describing lateral connections. Genes 1, 2, and 3 are assumed to be disease genes that affect the outcome

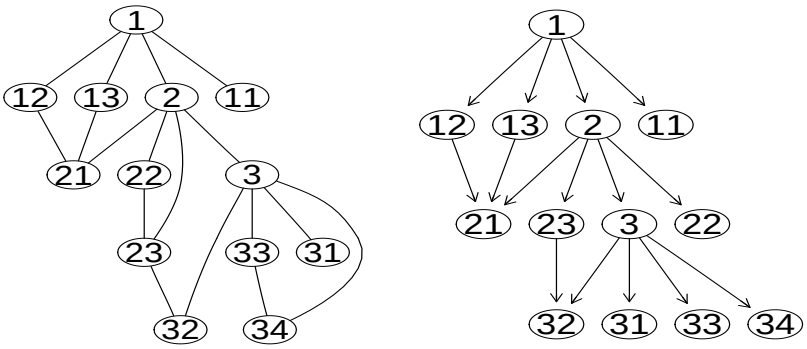

Figure 3.3: Left: a complicated network originating from gene 1. Right: DAG defined from the network.

weakly but importantly. Our goal is to identify disease genes, inlcuding those that play a critical role in mediating other genes in multiple biological processes even if their direct effects on the outcome are weak.

The simulated data were generated similarly as for the simple network. Here we considered two scenarios: (1) genes 1, 2, and 3 had weak effects ($\beta_1 = \beta_2 = \beta_3 = 0.1$), and leaf gene 32 had strong effect ($\beta_{32} = 10$); (2) the three disease genes had the same effect as in scenario (1) whereas leaf gene 34 had strong effect ($\beta_{34} = 10$).

Table 3.2 suggests that L_1-SVM generated sparse models yielding the most accurate prediction under both the scenarios. However, on average, it missed about 2.5 out of 4 informative genes and identified only around 0.5 out of 3 disease genes. In contrast, NG-SVM yielded models that contained every gene. Even though DGC-SVMs led to larger test errors than L_1-SVM, they detected most of the informative and disease genes. In particular, DGC-SVM-PT detected all the informative and disease genes, where the

Table 3.2: Complicated network: simulation results averaged over 100 runs ($p = 13$; 4 informative genes in each scenario).

Scenario	Method	Test Error (SE)	# False Negative (SE)	Model Size (SE)	# Disease Genes (SE)
1	STD	0.131 (0.002)	0.00 (0.00)	13.00 (0.00)	3.00 (0.00)
	L_1	0.089 (0.003)	2.59 (0.10)	3.04 (0.15)	0.41 (0.06)
	NG ($w = 1$)	0.214 (0.005)	0.00 (0.00)	13.00 (0.00)	3.00 (0.00)
	NG ($w = \sqrt{d}$)	0.210 (0.004)	0.00 (0.00)	13.00 (0.00)	3.00 (0.00)
	NG ($w = d$)	0.216 (0.005)	0.00 (0.00)	13.00 (0.00)	3.00 (0.00)
	PW ($w = 1$)	0.109 (0.003)	1.10 (0.09)	9.58 (0.16)	1.90 (0.08)
	PW ($w = \sqrt{d}$)	0.115 (0.003)	0.86 (0.08)	9.70 (0.16)	2.14 (0.08)
	PW ($w = d$)	0.117 (0.003)	0.46 (0.06)	10.65 (0.14)	2.54 (0.06)
	PT ($w = 1$)	0.146 (0.004)	0.00 (0.00)	10.79 (0.14)	3.00 (0.00)
	PT ($w = \sqrt{d}$)	0.145 (0.004)	0.00 (0.00)	10.86 (0.13)	3.00 (0.00)
	PT ($w = d$)	0.145 (0.004)	0.00 (0.00)	10.62 (0.14)	3.00 (0.00)
2	STD	0.137 (0.003)	0.00 (0.00)	13.00 (0.00)	3.00 (0.00)
	L_1	0.095 (0.003)	2.46 (0.10)	3.75 (0.16)	0.54 (0.07)
	NG ($w = 1$)	0.217 (0.005)	0.00 (0.00)	13.00 (0.00)	3.00 (0.00)
	NG ($w = \sqrt{d}$)	0.213 (0.004)	0.00 (0.00)	13.00 (0.00)	3.00 (0.00)
	NG ($w = d$)	0.215 (0.004)	0.00 (0.00)	13.00 (0.00)	3.00 (0.00)
	PW ($w = 1$)	0.101 (0.004)	2.12 (0.10)	5.52 (0.18)	0.88 (0.08)
	PW ($w = \sqrt{d}$)	0.099 (0.003)	1.56 (0.10)	6.79 (0.18)	1.44 (0.09)
	PW ($w = d$)	0.107 (0.004)	1.16 (0.09)	8.03 (0.18)	1.84 (0.08)
	PT ($w = 1$)	0.151 (0.004)	0.00 (0.00)	9.62 (0.16)	3.00 (0.00)
	PT ($w = \sqrt{d}$)	0.150 (0.004)	0.00 (0.00)	9.97 (0.15)	3.00 (0.00)
	PT ($w = d$)	0.152 (0.005)	0.00 (0.00)	9.98 (0.15)	3.00 (0.00)

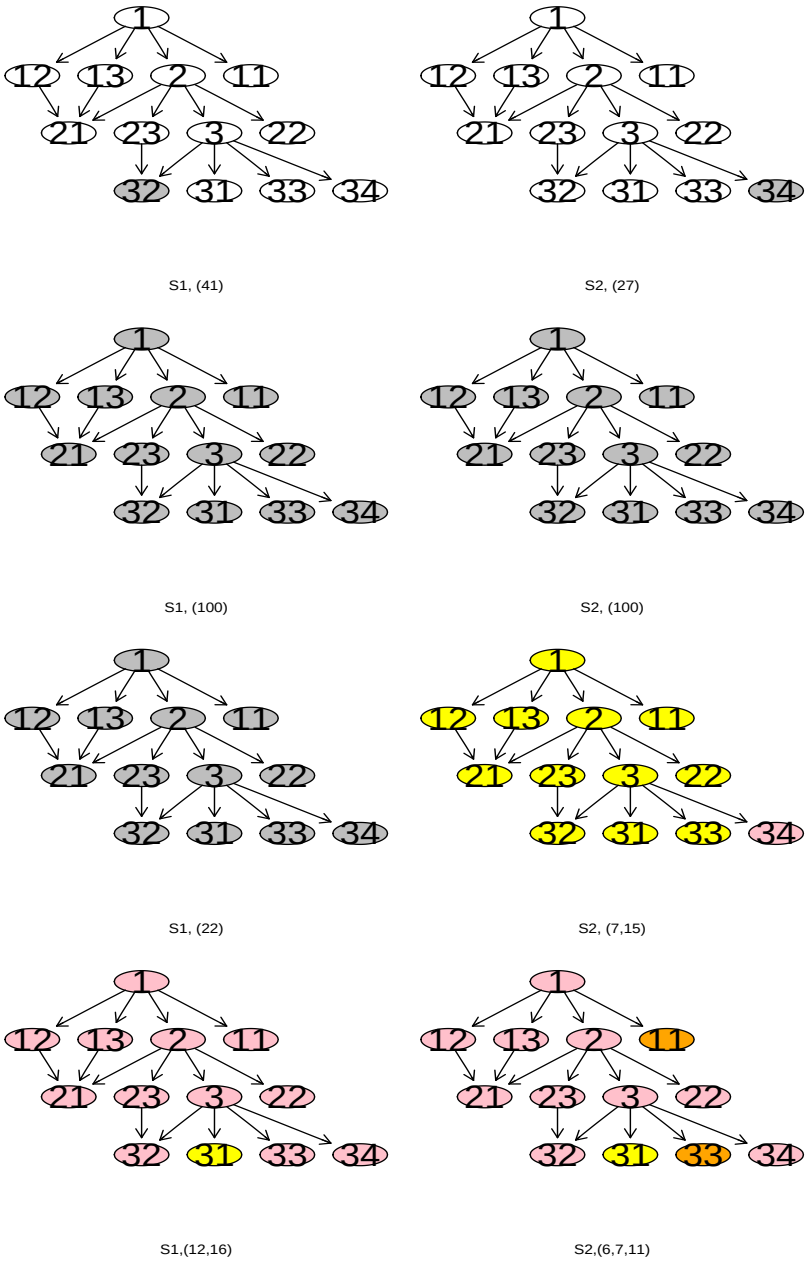

Figure 3.4: Most frequently recovered pathways. Rows from top to bottom: L_1-SVM, NG-SVM, DGC-SVM-PW, and DGC-SVM-PT. Columns from left to right: scenario 1 (S1) $\{1, 2, 3, 32\}$, and scenario 2 (S2) $\{1, 2, 3, 34\}$. Frequencies of the recovered pathways are included in parentheses.

results are insensitive to the choice of weight. Figure 3.4 displays the pathways most frequently identified by each method ($w = \sqrt{d}$ when a weight was involved) except the standard SVM in each scenario. It is evident that L_1-SVM captured the gene that exerted the largest influence on the response with the highest frequency (genes 32 and 34 in scenarios 1 and 2 respectively). NG-SVM involved all genes and hence that did not distinguish the informative from noise genes. After a close examination, we find that the DGC-SVM-PT selected all the informative genes as well as all the three disease genes in each run even though the three disease genes affected the response weakly. Although this method generated larger models with less accurate prediction compared with L_1-SVM, it exactly achieved our goal of identifying all the disease genes along a pathway, including those with only weak effects.

3.4 Applications to microarray data

To evaluate their performance in the real world, we applied the proposed DGC-SVMs to two microarray data sets related to breast cancer metastasis (Chuang *et al.* 2007, Wang *et al.* 2005) and Parkinson's disease (Gene Expression Omnibus: GSE6613 [`http://www.lncbi.nlm.nih.gov/geo/`]) respectively.

3.4.1 Breast cancer metastasis

The breast cancer metastasis (BC) data set contains expression levels of 8,141 genes from 286 patients, 106 of whom developed metastasis within a 5-year follow-up after surgery.

The data set includes three tumor suppressor genes, *TP53*, *BRCA1*, and *BRCA2*, which are known for preventing uncontrolled cell proliferation, and for playing a critical role in repairing the chromosomal damage. The malfunction of these genes leads to an increased risk of breast cancer (Costa *et al.* 2008, Nowacka-Zawisza *et al.* 2008). Together with the expression data set, the protein-protein interaction (PPI) network previously used by Chuang *et al.* (2007) was adopted as our prior biological information, which was obtained by assembling a pooled data set comprising 57,235 interactions among 11,203 proteins and curation of the literature. We considered a subnetwork consisting of the direct neighbors of the three tumor suppressor genes, denoted as BC-1nb-net, where expression levels of a total of 294 genes that belong to BC-1nb-net were available. In our analysis, due to their prominent roles, *TP53*, *BRCA1*, and *BRCA2* were considered as center genes of BC-1nb-net.

For evaluation, we randomly split the data into training, tuning, and testing set with 95, 95 and 96 observations respectively. The expression level of each gene was normalized to have mean 0 and standard deviation 1 across samples. Given any value from a prespecified set of wide-ranging values for the tuning parameter λ, we computed the classifier $\hat{f}_\lambda$ on the training set and applied $\hat{f}_\lambda$ to tuning set. The value of λ yielding the minimal misclassification error rate on the tuning data was chosen as $\hat{\lambda}$. Then we applied the classifier $\hat{f}_{\hat{\lambda}}$ on the test set for evaluation. We compared the performance of DGC-SVM with that of STD-SVM, L_1-SVM, and NG-SVM in terms of misclassification error rate, selection of mutant genes (genes with available mutation frequencies), and

model sparsity, averaged over 50 runs. Since genes with large mutation frequencies are more likely to malfunction, disturbing the aggregate activity of the network, a method that can detect more such mutant genes may be considered to perform better. The mutation frequencies of 227 genes are available (Chuang *et al.* 2007). Among the 294 genes in BC-1nb-net, 40 genes had mutation frequencies (denoted as cancer genes), 7 of which (*ABL1*, *JAK2*, *p53*, *PTEN*, *p14ARF*, *PTCH*, and *RB*) had a mutation frequency larger than 0.10 (denoted as cancer genes with large mutation frequency (CA-LMF)).

As indicated in Table 3.3, all the methods had similar predictive accuracies even though DGC-SVM-PT with $w = 1$ performed slightly better. However, an improvement in detecting clinically relevant genes was seen evidently by incorporating the prior network information. Compared with L_1-SVM, NG-SVM with $w = d$ detected almost twice as many frequently mutant cancer genes and more than 1.5 times cancer genes with models less than 1.5 times larger. DGC-SVM-PW with $w = d$ generated a more sparse model with 3 genes less than L1-SVM while detected the same number of cancer genes and almost twice as many frequently mutant cancer genes as L1-SVM. The advantage of DGC-SVM-PT with $w = d$ in gene selection was evident. It captured almost 3.5 times as many frequently mutant cancer genes and almost twice cancer genes by a model only 1.2 times larger than L1-SVM. The proposed DGC-SVM prevailed the other methods in identifying clinically important genes.

Table 3.3: BC-1nb-net: 294 genes including 40 cancer genes (CA) and 7 cancer genes with mutation frequencies larger than 0.10 (CA-LMF); misclassification error rate, number of selected CA; number of selected CA-LMF, number of selected genes, and their standard errors (SE in parentheses) obtained by averaging over 50 runs.

Method	Error (SE)	# CA-LMF	# CA	# Genes
STD	0.396 (0.006)	7.00 (0.00)	40.00 (0.00)	294.00 (0.00)
L_1	0.384 (0.007)	0.46 (0.09)	3.92 (0.27)	26.48 (0.69)
NG ($w = 1$)	0.385 (0.008)	0.58 (0.10)	3.82 (0.26)	29.38 (0.73)
NG ($w = \sqrt{d}$)	0.388 (0.006)	0.64 (0.11)	4.90 (0.29)	31.82 (0.75)
NG ($w = d$)	0.404 (0.006)	0.86 (0.12)	6.42 (0.33)	36.88 (0.80)
PW ($w = 1$)	0.380 (0.007)	0.78 (0.12)	4.86 (0.29)	42.32 (0.85)
PW ($w = \sqrt{d}$)	0.382 (0.006)	0.62 (0.11)	4.48 (0.28)	30.50 (0.74)
PW ($w = d$)	0.396 (0.006)	0.86 (0.12)	3.96 (0.27)	23.80 (0.66)
PT ($w = 1$)	0.373 (0.006)	1.76 (0.16)	14.90 (0.43)	88.18 (1.11)
PT ($w = \sqrt{d}$)	0.395 (0.006)	1.84 (0.16)	12.52 (0.41)	63.88 (1.00)
PT ($w = d$)	0.396 (0.005)	1.58 (0.16)	7.20 (0.34)	31.76 (0.75)

3.4.2 Parkinson's disease

The Parkinson's disease (PD) data set includes disease status and expression levels of 22,283 genes from 105 patients with 50 cases and 55 controls (Scherzer *et al.* 2007). The gene network information was obtained from two sources: (1) the network in Li and Li (2008), which combines 33 KEGG regulatory pathways and contains 1,523 genes and 6,865 edges; (2) the Parkinson's disease KEGG pathway (PD-KEGG, [http://www.genome.ad.jp/kegg/pathway/hsa/hsa05020.html]), which uncovers the interactions of 27 PD disease genes as of November 2008. A total of 12 out of the 27 PD disease genes are contained in the network of Li and Li (2008): *UBB*, *UBE1*, *CASP9*, *CASP3*, *APAF1*, *CYCS*, *PARK2*, *GPR37*, *SEPT5*, *SNCAIP*, *SNCA*, and *TH*. We focused our analysis on a subnetwork expanded from the 12 disease genes with the network of Li and Li (2008), denoted as PD-net (Figure 3.5), which consists of four components: (1) the 6th-order-neighbor-subnetwork of *UBB* (A direct neighbor of *UBB* is defined as a 1st-order-neighbor; a direct neighbor of a 1st-order-neighbor of *UBB* as a 2nd-order-neighbor; and so on.); (2) the 3rd-order-neighbor-subnetwork of *CASP9*; (3) the isolated four-gene-subnetwork including *PARK2*, *GPR37*, *SEPT5*, and *SNCAIP*; and (4) the isolated two-gene-subnetwork including *SNCA* and *TH*. A total of 181 genes belong to PD-net. Note that *PARK2*/*GPR37*/*SEPT5*/*SNCAIP* and *SNCA*/*TH* form two islands respectively. The DAG of PD-net (Figure 3.6) was obtained by merging the DAG of *UBB* and that of *CASP9* at the common node *SMAD7*. Note the two islands on the top right position in Figure 3.6.

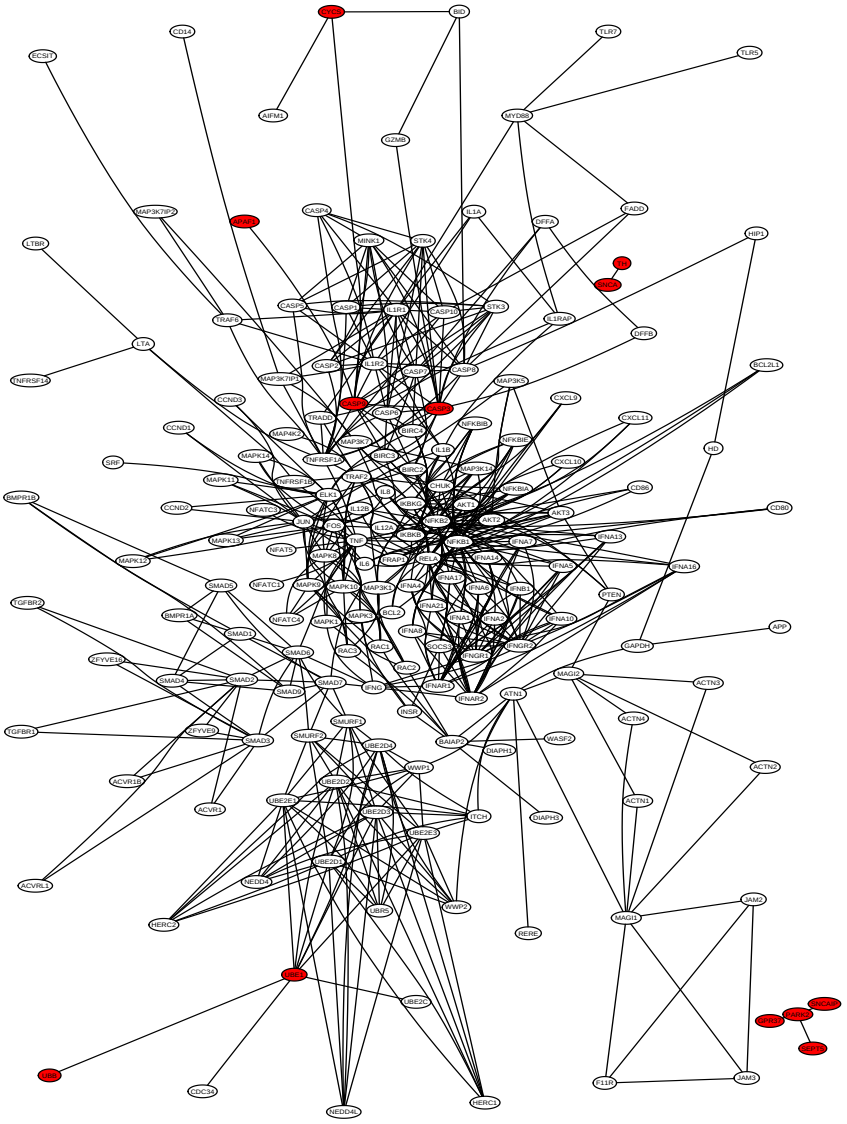

Figure 3.5: PD-net: subnetworks derived from the 12 PD disease genes (*UBB*, *UBE1*, *CASP9*, *CASP3*, *APAF1*, *CYCS*, *PARK2*, *GPR37*, *SEPT5*, *SNCAIP*, *SNCA*, and *TH*); 181 genes in total. PD genes are in red.

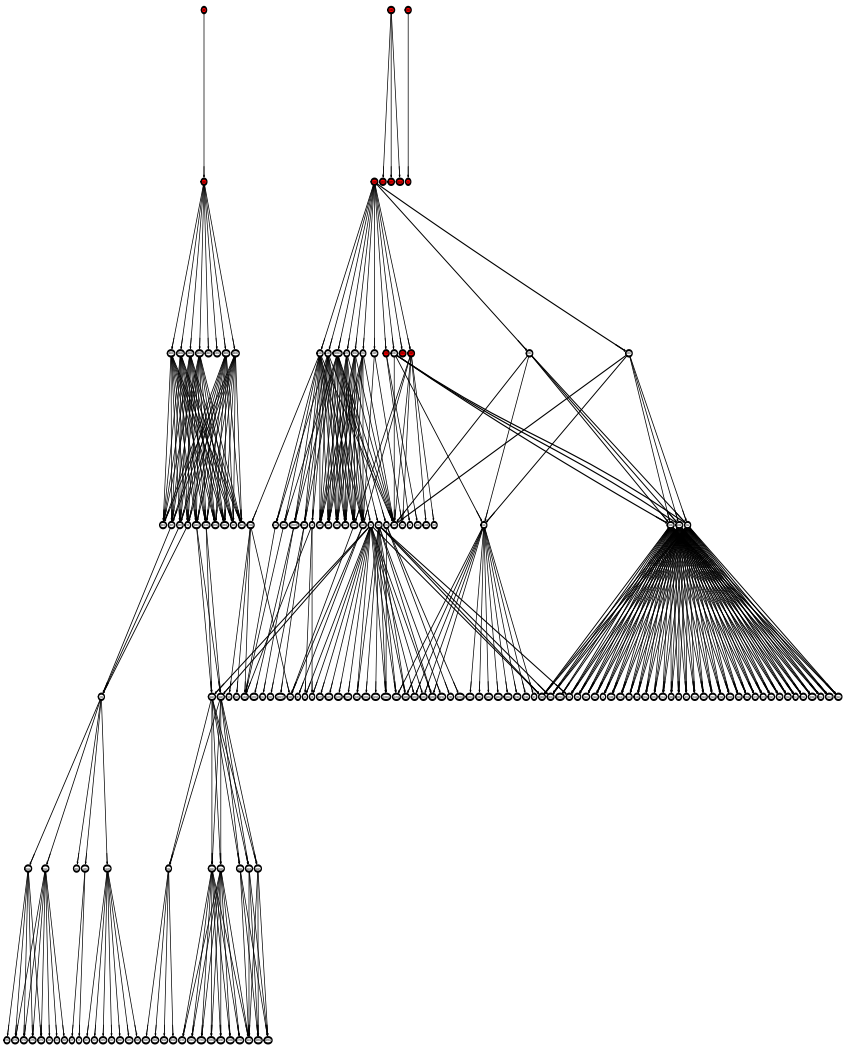

Figure 3.6: DAG transformed from PD-net: 12 PD disease genes (*UBB*, *UBE1*, *CASP9*, *CASP3*, *APAF1*, *CYCS*, *PARK2*, *GPR37*, *SEPT5*, *SNCAIP*, *SNCA*, and *TH*); 181 genes in total. PD genes are in red.

The data set was randomly split into training, tuning, and test sets with 40, 20, and 45 observations respectively. The expression level of each gene was normalized to have mean 0 and standard deviation 1 across samples. The performance of each method was evaluated on the test set by the misclassification error rate, the selection of PD genes, and model sparsity averaged over 50 runs. Again five methods were compared: STD-SVM, L_1-SVM, NG-SVM, DGC-SVM-PW ($w = 1$, $w = \sqrt{d}$, or $w = d$), and DGC-SVM-PT ($w = 1$, $w = \sqrt{d}$, or $w = d$). To obtain the final model for each method, we used a new tuning data set by combining, in each run, the previous tuning and test data, leading to a sample size as large as 65. Here $\hat{\lambda}$ was identified on the new tuning set from a wide range of prespecified values. Over 50 runs, the misclassification error rates were averaged corresponding to each tuning parameter value. The value $\hat{\lambda}$ leading to the minimal average error was used to fit the final model to all the data (105 observations). Note that the misclassification error from the final model was likely to be biased due to reuse of the data for training/tuning and test; the purpose of fitting the final model was to yield a set of selected genes.

As indicated in Table 3.4, the methods that incorporate the prior gene network information improved gene selection while maintaining predictive accuracy comparable to that of the STD-SVM and L_1-SVM. L_1-SVM generated models with an average of 16.86 genes including 1.50 PD genes. NG-SVM with $w = 1$ detected more than twice as many PD genes with models having 1.28 less genes than L_1-SVM. The improvement in gene selection of DGC-SVM-PW with $w = 1$ was more significant. It produced models

Table 3.4: Parkinson's disease: misclassification error rate, number of selected disease genes, number of selected genes, and their standard errors (SE in parentheses) obtained by averaging over 50 runs.

Method	Error (SE)	# Disease Genes	# Genes
STD	0.448 (0.011)	12.00 (0.00)	181.00 (0.00)
L_1	0.490 (0.008)	1.50 (0.16)	16.86 (0.55)
Final	-	5	70
NG ($w = 1$)	0.500 (0.011)	3.32 (0.22)	15.58 (0.53)
Final	-	8	83
NG ($w = \sqrt{d}$)	0.501 (0.009)	0.66 (0.11)	12.00 (0.47)
Final	-	12	102
NG ($w = d$)	0.493 (0.008)	0.00 (0.00)	14.10 (0.51)
Final	-	5	133
PW ($w = 1$)	0.481 (0.010)	4.80 (0.24)	13.10 (0.49)
Final	-	9	97
PW ($w = \sqrt{d}$)	0.493 (0.009)	3.26 (0.22)	13.22 (0.50)
Final	-	9	103
PW ($w = d$)	0.493 (0.007)	1.52 (0.16)	14.70 (0.52)
Final	-	7	97
PT ($w = 1$)	0.407 (0.011)	4.56 (0.24)	42.34 (0.81)
Final	-	6	108
PT ($w = \sqrt{d}$)	0.445 (0.013)	2.40 (0.20)	45.04 (0.82)
Final	-	6	113
PT ($w = d$)	0.456 (0.011)	2.42 (0.20)	55.56 (0.88)
Final	-	5	117

with 3.76 less genes while detected more than three times as many PD genes as L_1-SVM. Both the methods had a predictive accuracy at a comparable level to that of L_1-SVM. DGC-SVM-PT with $w = 1$ improved both prediction accuracy and gene selection. It reduced the misclassification error by 17% and also selected three times as many PD genes with a model size not so much larger than that of L_1-SVM. Overall, DGC-SVM-PT with $w = 1$ outperformed the other methods.

The selected genes of the final model obtained from each method (L_1-SVM, NG-SVM with $w = 1$, DGC-SVM-PW with $w = 1$, and DGC-SVM-PT with $w = 1$) are displayed in the networks in Figure 3.7–3.10. L_1-SVM generated the sparsest final model and missed the most PD genes. NG-SVM neglected around a half of the nodes and a half of the leaves. According to DGC-SVM-PT, nodes located at higher level are more likely to have nonzero estimates. The majority of nodes were detected and also both center genes of the two islands were included by this method. DGC-SVM-PW identified the most PD genes among the four final models even though it neglected most part of the subnetwork derived from *UBB*.

3.5 Discussion and conclusion

The availability of various repositories of gene networks and the accumulating knowledge on genes central to diseases make it possible to use these two sources of prior biological information to construct classifiers. Employing such a network-based perspective not only sheds insight on deciphering the complexity within the network modules but also

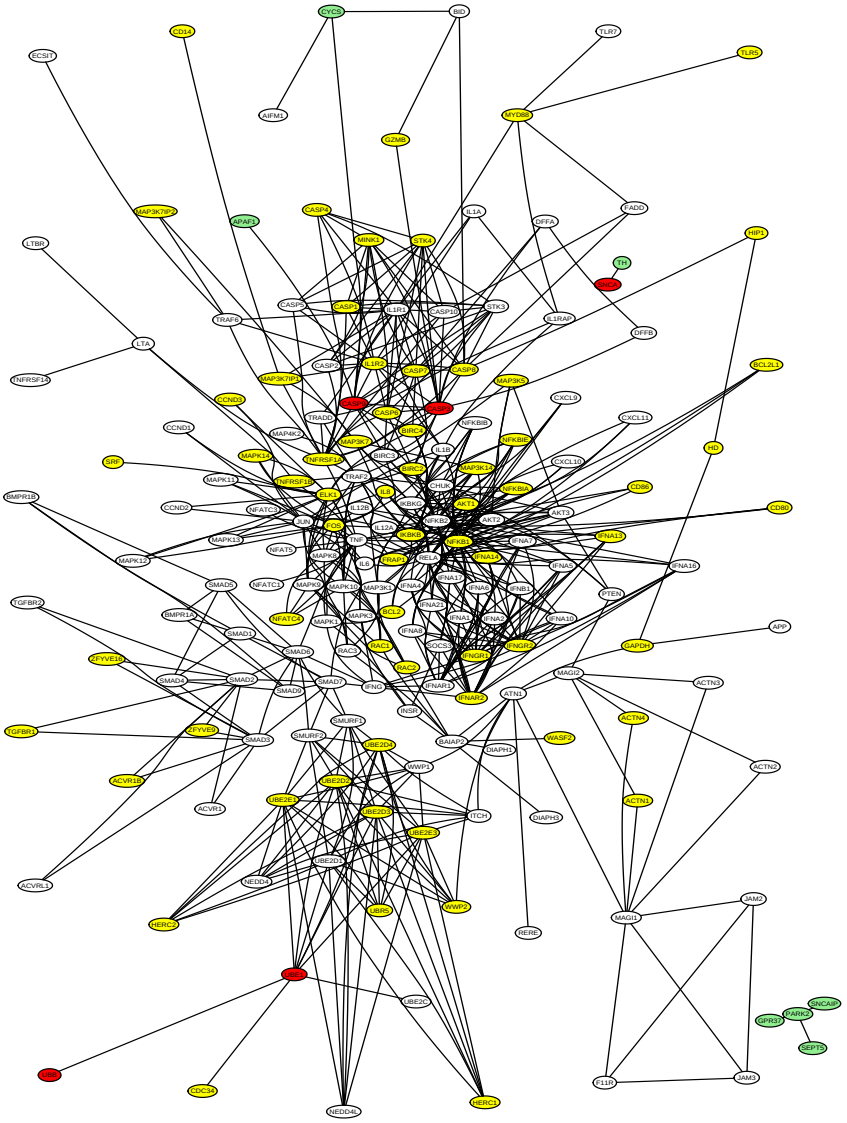

Figure 3.7: L1-SVM final model: selected PD genes (*UBB*, *UBE1*, *CASP9*, *CASP3*, and *SNCA*) in red; missed PD genes (*APAF1*, *CYCS*, *PARK2*, *GPR37*, *SEPT5*, *SNCAIP*, and *TH*) in green; other selected genes in yellow; unselected genes in white; 181 genes in total.

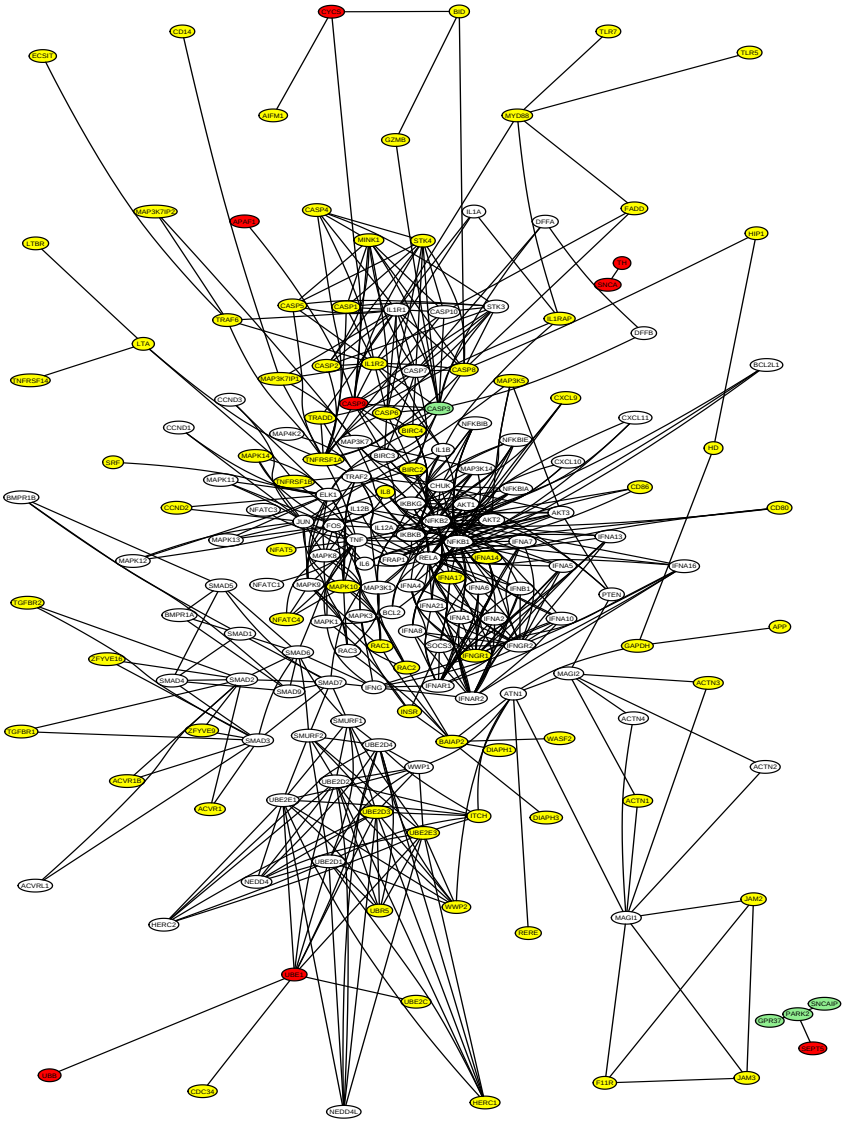

Figure 3.8: NG-SVM with $w = 1$ final model: selected PD genes (*UBB*, *UBE1*, *CASP9*, *APAF1*, *CYCS*, *SEPT5*, *SNCA*, and *TH*) in red; missed PD genes (*CASP3*, *PARK2*, *GPR37*, and *SNCAIP*) in green; other selected genes in yellow; unselected genes in white; 181 genes in total.

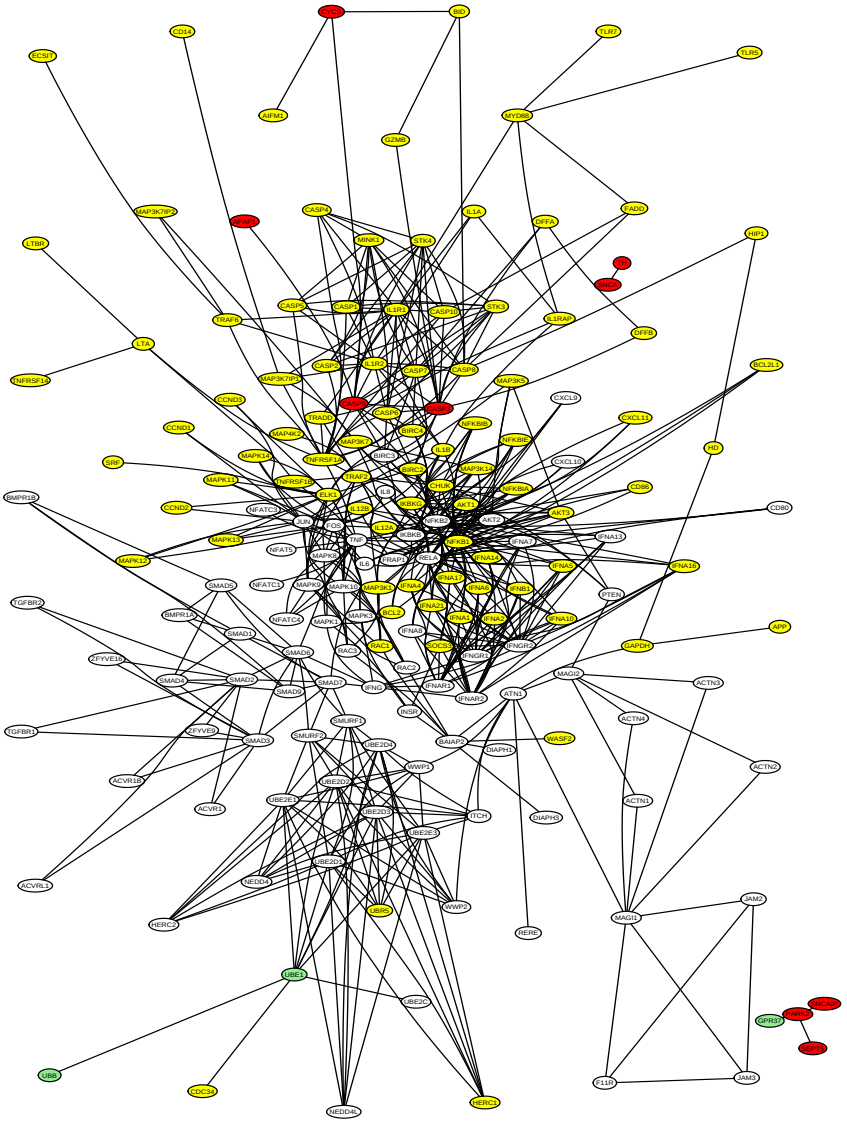

Figure 3.9: DGC-SVM-PW with $w = 1$ final model: selected PD genes (*CASP9*, *CASP3*, *SNCA*, *CYCS*, *APAF1*, *TH*, *PARK2*, *SEPT5*, and *SNCAIP*) in red; missed PD genes (*UBB*, *UBE1*, and *GPR37*,) in green; other selected genes in yellow; unselected genes in white; 181 genes in total.

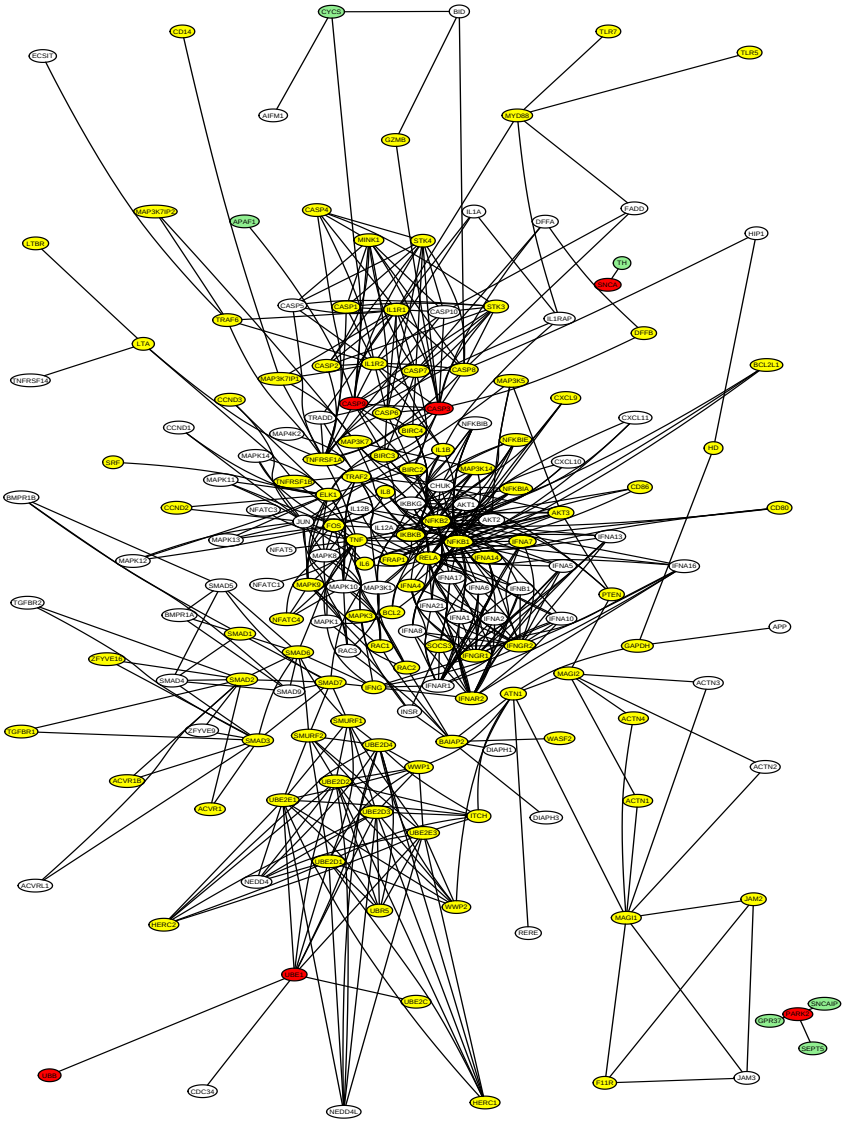

Figure 3.10: DGC-SVM-PT with $w = 1$ final model: selected PD genes (*UBB*, *UBE1*, *PARK2*, *CASP3*, *SNCA*, and *CASP9*) in red; missed PD genes (*APAF1*, *CYCS*, *GPR37*, *SEPT5*, *SNCAIP*, and *TH*) in green; other selected genes in yellow; unselected genes in white; 181 genes in total.

offers the possibility of detecting genes that play a critical role in mediating multiple biological processes but have weak direct effects on the outcome by themselves. Such genes are often ignored by expression analysis without the network information as pointed out by Chuang *et al.* (2007).

In this chapter, we proposes a penalty that encourages gene selection along pathways or within subnetworks. The identification of any gene leads to the search for disease genes along gene pathways or within subnetworks all the way toward genes central to the disease. The penalty enhances investigation of relationships within collectives of genes that contain both the selected genes and the central genes. By converting the undirected gene network into DAG, we imposed an upper-lower hierarchy on the gene network depending on each gene's distance to the assumed network center, typically gene(s) central to the disease. The DAG facilitated the definition of gene groups according to one of two different ways of grouping, one based on grouping genes along linear pathways, and a second on grouping genes with all their downstream genes. The penalty term was constructed from the L_∞-norm being applied to each group. Combined with the hinge loss, we obtained our proposed method: DGC-SVM-PW and DGC-SVM-PT. The former attempts to realize selection along linear pathways in the network. The latter ensures the selection of an upper-level gene if any of its downstream genes is selected, thus realizing hierarchical selection. Due to this property, genes that have more downstream genes and are closer to the center are very likely to be detected. In addition, selection of any gene guarantees the inclusion of at least one central gene. According to the simulation

studies, DGC-SVM detected more disease genes even if they potentially affected the outcome only weakly. Two real data applications showed that the new method prevailed STD-SVM and L_1-SVM in capturing clinically relevant genes while making either more accurate or comparable prediction on the outcome. We conclude that DGC-SVM has the potential to be an effective classification tool for microarray data.

Even though its strength in improving gene selection has been established, DGC-SVM has some weaknesses. First, specification of the center of a network is somewhat arbitrary. As a matter of fact, different specifications result in different grouping structures and thus different penalties. Second, how to define DAG in presence of multiple center genes requires further investigation. We suggested an approach that is admittedly somewhat arbitrary. Third, DGC-SVM-PW is computationally intensive for a large network since a large number of groups are generated. Fourth, use of the weight function may improve gene selection at the expense of reduced predictive accuracy or vice versa; it is not guaranteed to better both at the same time. A further investigation is necessary to develop a simpler and more effective way to incorporate a prior gene network.

Chapter 4

Support Vector Machine with a Truncated L_∞-norm Penalty

4.1 Introduction

To integrate the prior biological observation that genes work together, we have proposed two network-based SVMs, NG-SVM in Chapter 2, and DGC-SVM in Chapter 3. Both incorporate the prior network information by imposing on the hinge loss a penalty constructed on appropriately grouped genes. NG-SVM groups pairs of neighboring genes on the gene network. DGC-SVM defines an upper-lower hierarchy on the undirected network, and groups genes either along the pathways or with its lower-level genes. By summing up an L_∞-norm over each group, we obtain the penalty for NG-SVM and DGC-SVM. According to these two methods, genes from the same group are more likely to be selected simultaneously and have similar estimates. In particular, by DGC-SVM, identification of any gene triggers the search for disease genes along its connected gene pathways or within subnetworks. The penalty boosts investigation of relationships within collectives of genes that contain both the selected genes and the central genes. Both NG-SVM and DGC-SVM have been demonstrated to be effective classification tools for microarray samples.

Even though the proposed penalization methods have been shown to improve gene selection with predictive performance comparable to STD-SVM and L_1-SVM, they produce biased estimates, for large coefficients in particular, because their penalty terms continuously shrink all the coefficients uniformly toward 0. To alleviate over-shrinkage, especially for large coefficients, we built a weight function in the penalty in NG-SVM and DGC-SVM. By exploring various weight functions, we allow the data to reveal the

underlying mystery, and thus guide us towards the model with an appropriate weight function that best counteracts the shrinkage effect. Our proposed weight favors genes with more neighboring genes to have larger estimates, and thus correct for, to some extent, the bias induced by penalization.

In this chapter, we target over-shrinkage from a different perspective. We define a threshold parameter or a second tuning parameter ($C > 0$) in the penalty, which controls what coefficients to shrink: only coefficients smaller than C are penalized while those beyond C are freed from penalization. Since the threshold parameter is built in with the L_∞-norm, we name the novel penalty as truncated L_∞-norm penalty and the corresponding SVM as truncated L_∞-norm SVM (TL_∞-SVM). In this chapter, we only discuss TL_∞-SVM that is defined on groups of neighboring genes. Similarly, TL_∞-SVM can also be extended to DGC-SVM but we will not discuss it in details here. It appears that TL_∞-SVM improves variable selection. However, since the TL_∞ penalty is no longer convex, it imposes a great computational challenge. An efficient algorithm is developed to solve the nonconvex minimization problem by using difference convex programming, which can be easily implemented.

4.2 Methods

4.2.1 Existing methods

Suppose on training data $\{(x_i, y_i)\}_{i=1}^N$ with $x_i \in \mathbb{R}^p$ and $y_i \in \{1, -1\}$, we define a hyperplane $\{x : f(x) = x^T\beta + \beta_0 = 0\}$. The classification rule induced by $f(x)$ is

$sign[\hat{f}(x)]$. STD-SVM searches for $\hat{f}(x) = x^T\hat{\beta} + \hat{\beta}_0$ that maximizes the margin between the training data points for class 1 and class -1. This optimization problem can be formulated as:

$$\min_{\beta_0,\beta} \left\{ \sum_{i=1}^{N} \left[1 - y_i \left(x_i^T\beta + \beta_0\right)\right]_+ + \lambda\|\beta\|_2^2 \right\}$$

where $[z]_+ = max\{z, 0\}$, $\|\beta\|_2^2 = \sum_{k=1}^{p} |\beta_k|^2$, and $\lambda > 0$ is a tuning parameter. With its all nonzero coefficient estimates, STD-SVM fails to accomplish variable selection. The L_1-SVM (Zhu *et al.* 2003) was proposed to generate sparse model by forcing certain coefficient estimates to be exactly 0.

$$\min_{\beta_0,\beta} \left\{ \sum_{i=1}^{N} \left[1 - y_i \left(x_i^T\beta + \beta_0\right)\right]_+ + \lambda\|\beta\|_1 \right\}$$

where $\|\beta\|_1 = \sum_{k=1}^{p} |\beta_k|$.

Biological observations reveal that genes function together in multiple biological processes. The interrelationships between genes are described as gene networks or gene pathways. By combining pairwise neighboring genes into one group, we proposed the neighboring-gene SVM (NG-SVM) (Zhu *et al.* 2009). Consider a gene network with S denoting all edges, i.e., the pairwise connection of neighboring genes.

$$S = \{(j_1, j_2) : \text{gene } j_1 \text{ and gene } j_2 \text{ are connected}\}$$

Define w_k as some weight for gene k. For example, $w_k = \sqrt{d_k}$ where d_k is the number of direct neighbors of gene k, or $w_k = d_k$, or simply $w_k = 1$ for all genes. Here we only focus on $w_k = \sqrt{d_k}$. The NG-SVM solves the optimization problem as follows.

$$\min_{\beta_0,\beta} \left\{ \sum_{i=1}^{N} \left[1 - y_i \left(x_i^T\beta + \beta_0\right)\right]_+ + \lambda \sum_{(j_1,j_2)\in S} \max\left(\frac{|\beta_{j_1}|}{w_{j_1}}, \frac{|\beta_{j_2}|}{w_{j_2}}\right) \right\} \tag{4.1}$$

NG-SVM encourages $|\beta_{j_1}| = |\beta_{j_2}|$ if $w_k = 1$, $\frac{|\beta_{j_1}|}{\sqrt{d_{j_1}}} = \frac{|\beta_{j_2}|}{\sqrt{d_{j_2}}}$ if $w_k = \sqrt{d_k}$, and $\frac{|\beta_{j_1}|}{d_{j_1}} = \frac{|\beta_{j_2}|}{d_{j_2}}$ if $w_k = d_k$. Larger weights (from $w_k = 1$, $w_k = \sqrt{d_k}$, to $w_k = d_k$) favor genes with more direct neighbors, which are supposed to be biologically more important, to have larger coefficient estimates. Thus larger weight relaxes the shrinkage effect on large coefficients. Thus choice of weight provides a straightforward strategy to counteract the bias in the coefficient estimates from the penalization method and possibly improve the predictive performance.

4.2.2 Truncated L_∞-norm support vector machine

To correct the bias from penalization, we devise a TL_∞ penalty.

$$\begin{aligned} TL_\infty(|\beta|;\lambda,C) &= \sum_{(j_1,j_2)\in S} TL_\infty(|\beta_{j_1}|,|\beta_{j_2}|;\lambda,C) \\ &= \lambda \sum_{(j_1,j_2)\in S} \min\left[\max\left(\frac{|\beta_{j_1}|}{\sqrt{d_{j_1}}},\frac{|\beta_{j_2}|}{\sqrt{d_{j_2}}}\right),C\right] \end{aligned} \tag{4.2}$$

where $C > 0$ is a threshold parameter (a second tuning parameter). Three properties of the TL_∞ penalty are noteworthy.

(i) It corrects the bias by choosing which coefficients to be penalized. In each group the larger weighted coefficient is compared with C. The weighted coefficients beyond the threshold C are not penalized. By introducing C, we attempt to release large coefficients from penalization so as to avoid the bias in the coefficient estimates. Note that as $C \to \infty$ (4.2) reduces to the penalty term in (4.1). Therefore, NG-SVM is a special case of TL_∞-SVM.

(ii) In essence, the proposed TL_∞-SVM searches for the classifier by minimizing the number of nonzero edgewise larger weighted coefficients over the network. This is evident when (4.2) is rewritten as

$$TL_\infty(|\beta|;\lambda,C) = \lambda C \sum_{(j_1,j_2)\in S} I(M_{(j_1,j_2)} \geq C) + \lambda \sum_{(j_1,j_2)\in S} M_{(j_1,j_2)} I(M_{(j_1,j_2)} < C) \tag{4.3}$$

where $M_{(j_1,j_2)} = \max\left(\frac{|\beta_{j_1}|}{\sqrt{d_{j_1}}}, \frac{|\beta_{j_2}|}{\sqrt{d_{j_2}}}\right)$. Suppose $C \to 0$, $\lambda \to \infty$, and $\lambda C \to 1$, the first term in (4.3) approaches the number of nonzero $M_{(j_1,j_2)}$ and the second term goes to 0.

(iii) The TL_∞ penalty is non-convex piecewise linear (Figure 4.1 (a)). On the one hand, non-convexity poses a challenging minimization problem; on the other hand, piecewise linearity brings computational advantage. Note that the TL_∞ penalty (4.2) can be decomposed into the difference of two convex functions (Figure 4.1 (c) and (d)),

$$\begin{aligned} TL_\infty(|\beta|;\lambda,C) &= \lambda \sum_{(j_1,j_2)\in S} \max\left(\frac{|\beta_{j_1}|}{\sqrt{d_{j_1}}}, \frac{|\beta_{j_2}|}{\sqrt{d_{j_2}}}\right) \\ &\quad - \lambda \sum_{(j_1,j_2)\in S} \max\left(\frac{|\beta_{j_1}|}{\sqrt{d_{j_1}}}, \frac{|\beta_{j_2}|}{\sqrt{d_{j_2}}}, C\right) \end{aligned} \tag{4.4}$$

which facilitates us to develop an efficient algorithm for computation. This decomposition largely determines the convergence rate, stability, robustness, and globality of sought solutions (Liu *et al.* 2005). Next section discusses computation in more details.

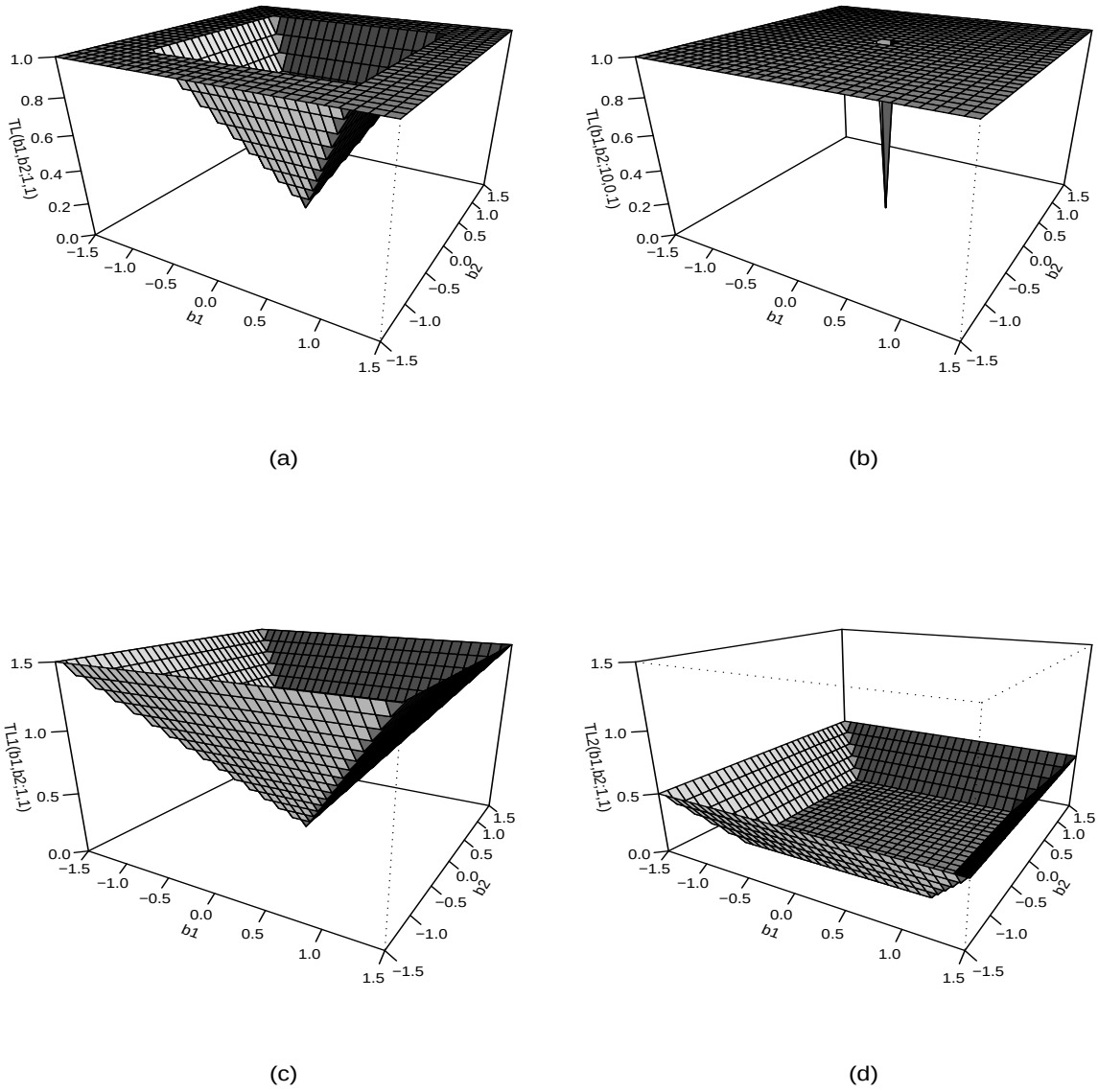

Figure 4.1: The truncated L_∞-norm penalty $TL_\infty(|b|;\lambda,C)$: (a) with $\lambda = 1$ and $C = 1$, (b) with $\lambda = 10$ and $C = 0.1$; $TL1_\infty$ in (a) is decomposed into the difference of two convex functions TL_∞ in (c) and $TL2_\infty$ in (d).

4.2.3 Computation

The non-convex minimization for TL_∞-SVM is a challenging problem. As far as we know, non-convex minimization lacks for a general approach to compute the global optimum. DC (difference of convex functions) programming (An and Tao, 1997) is a major advance in non-convex programming and optimization. If the objective function can be decomposed into a difference of two convex functions, DC programming seeks the final solution by iteratively constructing a sequence of convex approximations (Liu *et al.* 2005, ANOTHER DC LITERATURE). In general, it produces local optima and quite often global solutions, and proves to be more robust and more efficient than standard methods, especially in the large scale setting (An and Tao 2009).

We develop our DC algorithm as follows. By exploring property (iii) of the TL_∞ penalty, we decompose the objective function

$$\begin{aligned} S(\beta_0, \beta) &= \sum_{i=1}^{N} \left[1 - y_i \left(x_i^T \beta + \beta_0\right)\right]_+ \\ &\quad + \lambda \sum_{(j_1, j_2) \in S} \min \left[\max \left(\frac{|\beta_{j_1}|}{\sqrt{d_{j_1}}}, \frac{|\beta_{j_2}|}{\sqrt{d_{j_2}}} \right), C \right] \end{aligned} \tag{4.5}$$

into $S_1(\beta_0, \beta) - S_2(\beta)$, where

$$S_1(\beta_0, \beta) = \sum_{i=1}^{N} \left[1 - y_i \left(x_i^T \beta + \beta_0\right)\right]_+ + \lambda \sum_{(j_1, j_2) \in S} \max \left(\frac{|\beta_{j_1}|}{\sqrt{d_{j_1}}}, \frac{|\beta_{j_2}|}{\sqrt{d_{j_2}}} \right)$$

and

$$S_2(\beta) = \lambda \sum_{(j_1, j_2) \in S} \max \left(\frac{|\beta_{j_1}|}{\sqrt{d_{j_1}}}, \frac{|\beta_{j_2}|}{\sqrt{d_{j_2}}}, C \right)$$

At iteration m ($m \geq 1$), DC programming replaces $S_2(\beta)$ with its affine minorization $S_2(\hat{\beta}^{(m-1)}) + \langle |\beta| - |\hat{\beta}^{(m-1)}|, \nabla S_2(\hat{\beta}^{(m-1)}) \rangle$ to give birth to the upper approximation of

$S^{(m)}(\beta_0, \beta)$,

$$S^{(m)}(\beta_0, \beta) = S_1(\beta_0, \beta) - (S_2(\hat{\beta}^{(m-1)}) + \langle |\beta| - |\hat{\beta}^{(m-1)}|, \nabla S_2(\hat{\beta}^{(m-1)}) \rangle) \tag{4.6}$$

where $\langle \cdot, \cdot \rangle$ is the inner product, and $\nabla S_2(\hat{\beta}^{(m-1)})$ is a gradient vector of $S_2(\beta)$ evaluated at the solution at iteration $m-1$, $|\hat{\beta}^{(m-1)}|$, with kth element equal to

$$\lambda \sum_{(j_1, j_2) \in S} \frac{1}{\sqrt{d_k}} I\left(\frac{|\hat{\beta}_k^{(m-1)}|}{\sqrt{d_k}} = \max\left(\frac{|\hat{\beta}_{j_1}^{(m-1)}|}{\sqrt{d_{j_1}}}, \frac{|\hat{\beta}_{j_2}^{(m-1)}|}{\sqrt{d_{j_2}}}, C \right) \right) \tag{4.7}$$

Note that $S^{(m)}(\beta_0, \beta)$ is a convex upper bound of $S(\beta_0, \beta)$ by convexity of $S_2(\beta)$, and $S_2(\hat{\beta}^{(m-1)}) - \langle |\hat{\beta}^{(m-1)}|, \nabla S_2(\hat{\beta}^{(m-1)}) \rangle$ is a constant. This iterative process amounts to minimizing

$$\begin{aligned} & S_1(\beta_0, \beta) - \langle |\beta|, \nabla S_2(\hat{\beta}^{(m-1)}) \rangle \\ = & \sum_{i=1}^{N} \left[1 - y_i \left(x_i^T \beta + \beta_0 \right) \right]_+ + \lambda \sum_{(j_1, j_2) \in S} \max\left(\frac{|\beta_{j_1}|}{\sqrt{d_{j_1}}}, \frac{|\beta_{j_2}|}{\sqrt{d_{j_2}}} \right) \\ & - \lambda \sum_{k=1}^{p} \left[\frac{|\beta_k|}{\sqrt{d_k}} \sum_{(j_1, j_2) \in S} I\left(\frac{|\hat{\beta}_k^{(m-1)}|}{\sqrt{d_k}} = \max\left(\frac{|\hat{\beta}_{j_1}^{(m-1)}|}{\sqrt{d_{j_1}}}, \frac{|\hat{\beta}_{j_2}^{(m-1)}|}{\sqrt{d_{j_2}}}, C \right) \right) \right] \end{aligned} \tag{4.8}$$

with respect to β to yield $\hat{\beta}^{(m)}$ at iteration m. Note that $S_1(\beta_0, \beta)$ is exactly the formulation of NG-SVM.

By inserting the threshold parameter C, we restrain coefficient that is larger than both its paired weighted coefficient and C from penalization. This is evident if we rewrite

the penalty term in (4.8) as

$$\begin{aligned}
&\lambda\sum_{k=1}^{p}\left[\frac{|\beta_k|}{\sqrt{d_k}}\sum_{(j_1,j_2)\in S} I\left(\frac{|\hat{\beta}_k^{(m-1)}|}{\sqrt{d_k}} = \max\left(\frac{|\hat{\beta}_{j_1}^{(m-1)}|}{\sqrt{d_{j_1}}}, \frac{|\hat{\beta}_{j_2}^{(m-1)}|}{\sqrt{d_{j_2}}}, 0\right)\right)\right] \\
&-\lambda\sum_{k=1}^{p}\left[\frac{|\beta_k|}{\sqrt{d_k}}\sum_{(j_1,j_2)\in S} I\left(\frac{|\hat{\beta}_k^{(m-1)}|}{\sqrt{d_k}} = \max\left(\frac{|\hat{\beta}_{j_1}^{(m-1)}|}{\sqrt{d_{j_1}}}, \frac{|\hat{\beta}_{j_2}^{(m-1)}|}{\sqrt{d_{j_2}}}, C\right)\right)\right] \\
= \quad &\lambda\sum_{k=1}^{p}\left[\frac{|\beta_k|}{\sqrt{d_k}}\sum_{(j_1,j_2)\in S} I\left(C > \frac{|\hat{\beta}_k^{(m-1)}|}{\sqrt{d_k}} = \max\left(\frac{|\hat{\beta}_{j_1}^{(m-1)}|}{\sqrt{d_{j_1}}}, \frac{|\hat{\beta}_{j_2}^{(m-1)}|}{\sqrt{d_{j_2}}}\right)\right)\right]
\end{aligned} \tag{4.9}$$

Therefore, the TL_∞ penalty adaptively corrects the bias from the grouped L_∞ penalty (NG-SVM) by releasing large components but shrinking small ones of the estimate from the previous step. This self-correcting process continues until the termination criterion is met.

The linearity of the penalty term allows the use of linear programming (LP) technique to solve (4.8). Thus, the computation is very convenient.

$$\begin{aligned}
&\min_{\beta_0^+,\beta_0^-,\beta^+,\beta^-} \sum_{i=1}^{N}\xi_i + \lambda\sum_{(j_1,j_2)\in S} M_{(j_1,j_2)} \\
&-\lambda\sum_{k=1}^{p}\left[\frac{\beta_k^+ + \beta_k^-}{\sqrt{d_k}}\sum_{(j_1,j_2)\in S} I\left(\frac{|\hat{\beta}_k^{(m-1)}|}{\sqrt{d_k}} = \max\left(\frac{|\hat{\beta}_{j_1}^{(m-1)}|}{\sqrt{d_{j_1}}}, \frac{|\hat{\beta}_{j_2}^{(m-1)}|}{\sqrt{d_{j_2}}}, C\right)\right)\right]
\end{aligned} \tag{4.10}$$

subject to

$$\begin{aligned}
&y_i\left(\beta_0^+ - \beta_0^- + x_i^T(\beta^+ - \beta^-)\right) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i \\
&\frac{\beta_j^+}{\sqrt{d_j}} + \frac{\beta_j^-}{\sqrt{d_j}} \leq M_{(j_1,j_2)}, \quad j = j_1, j_2 \qquad \forall (j_1,j_2) \in S \\
&\beta_j^+ \geq 0, \quad \beta_j^- \geq 0, \quad j = j_1, j_2 \qquad \forall (j_1,j_2) \in S
\end{aligned} \tag{4.11}$$

where

$$\xi_i = \left[1 - y_i \left(\sum_{i=1}^{N} x_i^T \beta + \beta_0\right)\right]_+, \qquad i = 1, 2, \ldots, N \tag{4.12}$$

$$M_{(j_1,j_2)} = \max\left\{\frac{|\beta_{j_1}|}{\sqrt{d_{j_1}}}, \frac{|\beta_{j_2}|}{\sqrt{d_{j_2}}}\right\} \tag{4.13}$$

and $\beta_j = \beta_j^+ - \beta_j^-$, in which β_j^+ and β_j^- denote the positive and negative parts of β_j. This minimization can be easily implemented by the R package `lpsolve`, so is the computation of the L_1-SVM and NG-SVM. The R package `e1071` (with linear kernel) is used to obtain the solution to the STD-SVM.

The detailed algorithm for TL_∞-SVM is summarized as

Step 1 (Initialization). Compute the initial value, $(\beta_0^{(0)}, \beta^{(0)})$, and specify the termination criterion, for example, $\delta = 10^{-6}$. The default $(\beta_0^{(0)}, \beta^{(0)})$ is the NG-SVM estimate.

Step 2 (Iteration). Solve (4.10) by the R package `lpsolve` at iteration m.

Step 3 (Termination). Terminate the process if $S^{(m-1)}(\beta_0^{(m-1)}, \beta^{(m-1)}) - S^{(m)}(\beta_0^{(m)}, \beta^{(m)}) \leq \delta$. Then $(\beta_0^{(m^*)}, \beta^{(m^*)})$ is the solution where m^* is the smallest m that terminates the iteration.

Due to the linearity of the penalty term and the use of LP technique, this algorithm converges rapidly. According to our empirical experience, it usually takes 4 or 5 iterations at the most for the estimate to converge.

4.3 Results

The performance of TL_∞-SVM was compared with that of the existing popular methods, STD-SVM and L_1-SVM, as well as its special case NG-SVM in terms of predictive accuracy and gene selection. L_1-SVM, NG-SVM, and TL_∞-SVM were implemented by R package `lpsolve`, while STD-SVM by `e1071`.

4.3.1 Simulation

Suppose the network was composed of 5 subnetworks, each with a regulator gene ($t = 1, \ldots, 5$) that regulated 10 target genes, leading to a total of 55 ($p = 55$) genes. Two out of the five subnetworks were assumed to be informative; that is, 22 genes determined the outcome while the rest 33 genes had no effect. The simulated data were generated by the following steps:

- Generate the expression level of regulator gene t, $X_t \sim N(0, 1)$, $t = 1, \ldots, 5$, independently.

- Assume that the expression level of regulator gene t and each of its regulated genes follow a bivariate normal distribution with correlation 0.7. Thus, the expression level of each target gene regulated by gene t, $X_l^{(t)} \sim N(0.7X_t, 0.51)$, $l = 1, \ldots, 10$ and $t = 1, \ldots, 5$.

- Generate a classifier from a linear model: $f(X) = sign\left(X^T\beta + \beta_0\right)$, $\beta_0 = 2$, where X is the vector of the expression levels of all the genes, and coefficient

vector $\beta = (\beta_1^{(1)}, \ldots, \beta_{10}^{(1)}, \ldots, \beta_1^{(5)}, \ldots, \beta_{10}^{(5)})$. Randomly flip 5% of each group to obtain Y.

Four scenarios were specified as:

(i) $\beta = (10, \underbrace{5, \cdots, 5}_{10}, -10, \underbrace{-5, \cdots, -5}_{10}, 0, \cdots, 0)$.
The effect of one informative subnetwork was the same as the other in magnitude but with an opposite direction.

(ii) $\beta = (10, \underbrace{5, \cdots, 5}_{10}, 5, \underbrace{3, \cdots, 3}_{10}, 0, \cdots, 0)$.
Both informative subnetworks had positive effects but in different magnitudes.

(iii) $\beta = (10, \underbrace{5, \cdots, 5}_{7}, -5, -5, -5, 5, \underbrace{3, \cdots, 3}_{7}, -3, -3, -3)$.
Target genes in the same informative subnetworks had both positive and negative effects.

(iv) $\beta = (10, \underbrace{5, \cdots, 5}_{6}, \underbrace{-5, \cdots, -5}_{4}, -5, \underbrace{-3, \cdots, -3}_{6}, \underbrace{3, \cdots, 3}_{4}, 0, \cdots, 0)$.
It was similar to but more extreme than scenario 3.

Under each scenario, we generated 100 observations for training the classifier, an independent tuning data set of equal size to identify $(\hat{\lambda}, \hat{C})$ that gave rise to the minimal misclassification error rate on the tuning data set, and another 10,000 observations for testing the $\hat{f}(X|\hat{\lambda}, \hat{C})$. We repeated this process 100 times (i.e., 100 independent runs). We used the coefficient estimate from NG-SVM as the initial value $(\beta_0^{(0)}, \beta^{(0)})$. The termination criterion was specified as 10^{-6}. We searched for $(\hat{\lambda}, \hat{C})$ over a grid with

as few as 100 equally spaced points up to a 2100-point grid depending on the training and tuning data sets. Usually the prespecified values for C ranged from 0.01 to $\max\{(|\beta_0^{(0)}|, |\beta_1^{(0)}|, \ldots, |\beta_p^{(0)}|)\}$, and the increments varied between 0.01 and 0.1. $\hat{\lambda}$ was examined between 5 to 25 with increments between 0.05 to 0.5. The means of test error (the percent of misclassified predicted outcomes), false negatives (the number of informative genes with zero estimates), false positives (the number of noninformative genes with nonzero estimates), model sizes (the number of genes with nonzero estimates), and their corresponding standard errors ($sd/\sqrt{run}$) are reported in Table 4.1.

We find that TL_∞-SVM improved gene selection at the expense of a slight increase in test error. For the small network ($p = 55$), under each scenario, NG-SVM had the highest predictive accuracy while TL_∞-SVM fell between STD-SVM and L_1-SVM. Obviously L_1-SVM generated such sparse models that they missed at least half of the informative genes. By incorporating the network structure in model formulation, NG-SVM significantly improved false negatives compared with L_1-SVM. Even though the predictive performance of TL_∞-SVM was not as good as NG-SVM, it significantly reduced both false negatives and false positives compared with L_1-SVM and NG-SVM. The results were similar when $p = 550$. NG-SVM generated the smallest test error and TL_∞-SVM the second. A combination of the false negative and false positive findings suggested that TL_∞ made 1.43 less inaccurate estimates than its special case, NG-SVM, averaged over all the four scenarios.

Table 4.1: Simulation Results averaged over 100 runs for $p = 55$ and $p = 550$ (22 informative genes; 33 or 528 noise genes).

	Scenario	Method	Test Error (SE)	# False Negative (SE)	# False Positive (SE)	Model Size (SE)
$p = 55$	1	STD	0.135 (0.002)	0.0 (0.0)	33.0 (0.0)	55.0 (0.0)
		L_1	0.154 (0.002)	12.3 (0.2)	3.6 (0.2)	13.3 (0.3)
		NG	0.122 (0.003)	0.4 (0.1)	2.6 (0.2)	24.2 (0.4)
		TL_∞	0.141 (0.004)	0.7 (0.1)	1.7 (0.1)	23.1 (0.4)
	2	STD	0.133 (0.002)	0.0 (0.0)	33.0 (0.0)	55.0 (0.0)
		L_1	0.155 (0.002)	11.8 (0.2)	4.1 (0.2)	14.3 (0.3)
		NG	0.129 (0.003)	0.9 (0.1)	3.1 (0.2)	24.2 (0.4)
		TL_∞	0.143 (0.004)	2.7 (0.2)	0.8 (0.1)	20.1 (0.4)
	3	STD	0.181 (0.002)	0.0 (0.0)	33.0 (0.0)	55.0 (0.0)
		L_1	0.185 (0.002)	11.6 (0.2)	5.1 (0.2)	15.5 (0.3)
		NG	0.179 (0.002)	3.7 (0.2)	7.2 (0.2)	25.5 (0.4)
		TL_∞	0.199 (0.003)	7.4 (0.2)	1.3 (0.1)	15.9 (0.3)
	4	STD	0.213 (0.003)	0.0 (0.0)	33.0 (0.0)	55.0 (0.0)
		L_1	0.216 (0.002)	11.6 (0.2)	6.2 (0.2)	16.6 (0.3)
		NG	0.206 (0.002)	4.7 (0.2)	11.2 (0.3)	28.5 (0.4)
		TL_∞	0.208 (0.002)	6.0 (0.2)	6.0 (0.2)	22.0 (0.4)
$p = 550$	1	STD	0.268 (0.002)	0.0 (0.0)	528.0 (0.0)	550.0 (0.0)
		L_1	0.179 (0.003)	14.8 (0.2)	7.7 (0.3)	14.9 (0.4)
		NG	0.144 (0.004)	0.3 (0.1)	9.9 (0.3)	31.6 (0.6)
		TL_∞	0.152 (0.004)	1.1 (0.1)	8.4 (0.3)	29.3 (0.5)
	2	STD	0.270 (0.002)	0.0 (0.0)	528.0 (0.0)	550.0 (0.0)
		L_1	0.182 (0.003)	14.8 (0.2)	11.7 (0.3)	19.0 (0.4)
		NG	0.151 (0.004)	1.0 (0.1)	9.1 (0.3)	30.1 (0.5)
		TL_∞	0.156 (0.004)	2.4 (0.2)	5.9 (0.2)	25.5 (0.5)
	3	STD	0.306 (0.002)	0.0 (0.0)	528.0 (0.0)	550.0 (0.0)
		L_1	0.222 (0.003)	15.2 (0.2)	12.5 (0.4)	19.2 (0.4)
		NG	0.213 (0.003)	4.4 (0.2)	13.6 (0.4)	31.2 (0.5)
		TL_∞	0.214 (0.003)	6.0 (0.2)	9.0 (0.3)	25.0 (0.5)
	4	STD	0.336 (0.002)	0.0 (0.0)	28.0 (0.0)	550.0 (0.0)
		L_1	0.254 (0.003)	15.6 (0.2)	14.0 (0.4)	20.4 (0.4)
		NG	0.250 (0.003)	9.1 (0.2)	15.6 (0.4)	28.5 (0.5)
		TL_∞	0.254 (0.003)	10.2 (0.2)	14.3 (0.4)	26.2 (0.5)

4.3.2 Real data application

The performance of the newly proposed method was evaluated on the breast cancer metastasis (BC) expression data (Wang *et al.* 2005, Chuang *et al.* 2007). The expression levels of 8,141 genes from 286 patients are included in the BC data. 106 of the patients developed metastasis within a 5-year follow-up after surgery.

We focused our analysis on two subnetworks derived from the three human genes, *TP53*, *BRCA1*, and *BRCA2*, which are known to play a critical role in metastasis development. These three human genes are vital in repairing the chromosomal damage. Their malfunction results in uncontrolled cell proliferation and thus increase the risk of breast cancer. The subnetworks are defined on the protein-protein interaction (PPI) network (Chuang *et al.* 2007). The PPI network describes 57,235 interactions among 11,203 proteins, obtained by assembling various sources of experimental data and curation of the literature (Chuang *et al.* 2007). *TP53*, *BRCA1*, and *BRCA2* and their direct neighbors (genes immediately connected with a certain gene) consist of the first-order subnetwork (BC-1nb-net). The direct neighbors of *TP53*, and the second-order neighbors of *BRCA1* and *BRCA2* form the second-order subnetwork (BC-2nb-net).

The four methods, STD-SVM, L_1-SVM, NG-SVM, and TL_∞-SVM, were compared in terms of misclassification error rate, cancer genes selection, and model sparsity. The cancer genes (CA) are the 227 known or putative cancer genes with estimated mutation frequencies in cancer samples (Chuang *et al.* 2007). Genes with high mutation frequencies are believed to stimulate disease progression.

Among the 227 known cancer genes, those with mutation frequencies greater than 0.1 were defined as cancer genes with large mutant frequencies (CA-LMF). BC-1nb-net included 294 genes with expression levels available, 40 CA, and 7 CA-LMF (*ABL1, JAK2, p53, PTEN, p14ARF, PTCH*, and *RB*). BC-2nb-net was made up of 1,718 genes with observed expression levels, including 107 CA and 7 additional CA-LMF (*ACH, APC, EGFR, KIT, NICD, RAS*, and *CTNNB1*). We randomly split the sample into 95/95/96 for training/tuning/test. When we applied TL_∞-SVM, $(\hat{\lambda}, \hat{C})$ was identified on a grid of around 250 points.

We find that TL_∞-SVM detected more CA and CA-LMF than the other three methods at the expense of misclassification error and model sparsity (Table 4.2). However, gains in gene selection are worth the sacrifice if the research interest puts more weight on gene selection. For BC-1nb-net, NG-SVM performed better than L_1-SVM by selecting one more CA with a slightly larger model size at a comparable predictive accuracy. TL_∞-SVM doubled the number of selected CA-LMF and selected more than twice the number of CA with a model size twice as big as NG-SVM. The advantage of TL_∞-SVM in gene selection is also evident for BC-2nb-net. It detected more than three times as many as CA-LMF, and more than twice the number of CA with a model slightly less than twice as big as NG-SVM.

Table 4.2: BC-1nb-net/BC-2nb-net: 294/1,718 genes in total including 40/107 cancer genes, and 7/14 cancer genes with mutation frequencies larger than 0.10. Misclassification error rate, number of selected cancer genes with mutation frequencies larger than 0.10 (CA-LMF), number of selected cancer genes (CA), number of selected genes, and their standard errors (SE in parentheses) obtained by averaging over 50 runs.

Network	Method	Error	# CA-LMF	# CA	# Genes
BC-1nb-net	STD	0.396 (0.006)	7.0 (0.0)	40.0 (0.0)	294.0 (0.0)
	L_1	0.384 (0.007)	0.5 (0.1)	3.9 (0.3)	26.5 (0.7)
	NG	0.388 (0.006)	0.6 (0.1)	4.9 (0.3)	31.8 (0.8)
	TL_∞	0.408 (0.007)	1.2 (0.1)	10.6 (0.4)	62.8 (1.0)
BC-2nb-net	STD	0.363 (0.006)	14.0 (0.0)	107.0 (0.0)	1,718.0 (0.0)
	L_1	0.386 (0.006)	0.1 (0.1)	0.8 (0.1)	33.4 (0.8)
	NG	0.393 (0.007)	0.3 (0.1)	1.4 (0.2)	38.7 (0.8)
	TL_∞	0.407 (0.006)	1.0 (0.1)	3.2 (0.2)	76.7 (1.1)

4.4 Discussion and conclusion

In this chapter, we were motivated to correct the bias from the NG penalty (grouped L_∞-norm). To achieve this, we employed a nonconvex linear penalty term by inserting a threshold parameter ($C > 0$; a second tuning parameter) into the grouped L_∞-norm, leading to the TL_∞-SVM. The TL_∞ penalty intends to free the large components of the coefficient vector from penalization and impose shrinkage only on small components. When $C \to \infty$, it essentially reduces to NG-SVM.

Such a nonconvex penalty brought a great computational challenge. As far as we know, there has been no general approach to solve a nonconvex optimization problem. We explored the DC property of the newly proposed penalty, and designed an algorithm through DC programming, a major advance in nonconvex programming and optimization. We proposed to use the NG-SVM estimate as the initial value, $(\beta_0^{(0)}, \beta^{(0)})$, to facilitate the computation of the final solution. On the other hand, the linearity of the penalty made it possible to solve the minimization problem at each iteration, which was very computationally convenient. It usually took only 4 or 5 iterations at the most until convergence based on our empirical experience. This algorithm is able to generate coefficient estimate efficiently.

The simulation study shows that TL_∞-SVM improved gene selection at the expense of a slight increase in test error. It made fewer mistakes in gene selection than its special case NG-SVM. The real world application suggests that the gains of TL_∞-SVM in gene selection are worth the sacrifice in model sparsity and predictive accuracy if the research

interest is on gene selection.

The limitation of TL_∞-SVM mainly resides in three aspects. First, the proposed algorithm based on DC programming does not ensure a global solution. When implementing this algorithm, it would be better to choose a good initial value. We proposed to start from the NG-SVM estimate. The L_1-SVM is also an available option. A better practice is to use multiple initial values, for example, estimates from NG-SVM and L_1-SVM, compare $S^{(m^*)}(\beta_0, \beta)$ obtained from each initial value, and choose the estimate corresponding to the minimum $S^{(m^*)}(\beta_0, \beta)$ as the final solution. Second, the use of two tuning parameters increases the computational cost exponentially compared with NG-SVM. We employed a grid search to tune λ and C. We suggest to choose $C \in (0, \max(|\beta_0^{(0)}|, |\beta_1^{(0)}|, \ldots, |\beta_p^{(0)}|))$. Third, TL_∞-SVM may not correct the bias as substantially as we have expected. After a close examination of the magnitude of the coefficient estimates in the simulation study, we find that TL_∞-SVM and NG-SVM often produced similar coefficient estimates. To validate the effectiveness of the truncated penalty, we applied the truncated L_1-norm, that is, $\sum_{j=1}^{p} \min(|\beta_j|, C)$, and find that the coefficients are increased substantially (from 2 to 39 times as large as those generated from L_1-norm penalty). The ineffectiveness of TL_∞-SVM in bias reduction could be presumably due to its incorporation of the complicated network structure. One gene could appear in multiple groups and the complicated repeated comparisons are involved to decide whether this gene should be released from penalization or not. This problem still needs a convincing explanation. The analysis in this chapter focuses on the default

weight function $w = \sqrt{d}$. More weight functions, such as $w = d$ and $w = d^2$, could be explored. Also, a further refined grid search that covers a larger area could be conducted.

Chapter 5

Discussion and Future Work

5.1 Discussion

This study aims to take a coherent view of genes instead of treating them as independently and identically *a priori*, and to make full use of the accumulating knowledge on functional interrelationships of genes, described as gene networks, to improve predictive accuracy and gene selection. The underlying assumption of this network-based approach is that genes clustering together are more similar than those father apart, and clustering genes are more likely to participate in disease progression at the same time. Because of the ever-increasing popularity in penalization methods for high-dimensional data, and also the flexibility in penalty specialization to achieve our goals, we propose three network-based SVM by embedding the network information in various penalty terms to build binary classifiers.

We extend the concept of grouped variable selection to the context of microarray data, and explore various groupings of genes over available gene networks. In Chapter 2, we consider any two neighboring genes as one group and propose the NG-SVM. NG-SVM tends to select pairs of neighboring genes. In Chapter 3, we devise an orientation scheme to impose an upper-lower hierarchy on the undirected gene network, which facilitates us to define gene groups according to two ways of grouping: pathway grouping (PW) and partial tree grouping (PT). The corresponding penalties lead to DGC-SVM-PW and DGC-SVM-PT. The former encourages gene selection along the connected genetic paths while the latter ensures the inclusion of all genes at the upper level if any lower level gene is selected. The DGC-SVMs exactly implement our assumption and select

collectives of genes clustering around disease genes. To correct the bias from penalization, we propose TL_∞-SVM, which intends to free large components of the coefficient vector from penalization and exert shrinkage only on small components. Empirical experiences demonstrate that the three network-based SVMs detected more clinically important genes than the generic methods, STD-SVM and L_1-SVM.

5.2 Future work

Even though the proposed methods have been shown to improve gene selection, the limitations are mainly in three aspects. Improvements could be performed in the future:

(i) Due to the current limited knowledge on gene interactions, we have to exclude genes with expression levels but without available network information out of the study. A more complete network could be used in the future when interrelationships of more and more genes are unraveled.

(ii) When defining the hierarchy for constructing DGC-SVM penalty in presence of multiple center genes, we propose an approach that is admittedly arbitrary. The arbitrariness involved in specifying center genes and merging multiple DAGs result in different grouping structures and thus different penalties. A further investigation is necessary to develop a simpler and more effective way to incorporate a prior gene network.

(iii) DGC-SVM-PW is computationally intensive when a large number of groups are

derived from a large gene network. In addition, the proposed algorithm for solving TL_∞-SVM does not ensure a global optima, which may not lead to good estimates and thus influence its performance. More computationally efficient and effective algorithms could be explored in the future.

Chapter 6

Bibliography

Alfarano, C., Andrade, C.E., Anthony, K., *et al.* (2005). The Biomolecular Interaction Network Database and related tools 2005 update. *Nucleic Acids Res Database Issue*, **33**, D418–D424.

An, L.T.H., and Tao, P.D. (1997). Solving a class of linearly constrained indefinite quadratic problems by D.C. algorithms. *Journal of Global Optimization*, **11**, 253-285.

An, L.T.H., and Tao, P.D. (2009). DC programming and DCA for nonconvex optimization: theory, algorithms and applications. *3rd International Conference on Approximation Methods and Numerical Modelling in Environment and Natural Resources*, France, June 2009.

Bao, T., and Davidson, N.E. (2008). Gene expression profiling of breast cancer. *Ad-*

vances in Surgery, **42**, 249–260.

Benson, M., Carlsson, L., Guillot, G., *et al.* (2006). A network-based analysis of allergen-challenged CD4+ T cells from patients with allergic rhinitis. *Genes and Immunity*, **7**, 514–521.

Bondell, H.D., and Reich, B.J. (2008). Simultaneous regression shrinkage, variable selection, and supervised clustering of predictors with OSCAR. *Biometrics*, **64**, 115–123.

Børresen-Dale, A.L. (2003). TP53 and breast cancer. *Human Mutation*, **21**, 292–300.

Brown, M.P.S., Grundy, W.N., Lin, D., *et al.* (2000). Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc Natl Acad Sci U S A*, **97**, 262–267.

Brown, P.O., and Botstein, D. (1999). Exploring the new world of the genome with DNA microarrays. *Nature Genetics Supplement*, **21**, 33–37.

Calvano, S.E., Xiao, W.Z., Richards, D.R., *et al.* (2005). A network-based analysis of systematic inflammation in humans. *Nature*, **437**, 1032–1037.

Chuang, H.Y., Lee, E.J., Liu, Y.T., *et al.* (2007). Network-based classification of breast cancer metastasis. *Molecular Systems Biology*, **3**, doi: 10.1038/msb4100180.

Cortes, C., and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, **20**, 273–297.

Costa, S., Pinto, D., Pereira, D., *et al.* (2008). Importance of TP53 codon 72 and intron 3 duplication 16bp polymorphisms in prediction of susceptibility on breast cancer. *BMC cancer*, **8**, 1-7.

Dong, W., Tu, S., Xie, J., *et al.* (2009). Frequent promoter hypermethylation and transcriptional downregulation of BTG4 gene in gastric cancer. *Biochemical and Biophysical Research Communications*, **387**, 132–138.

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

Ein-Dor, L., Zuk, O., and Domany, E. (2006). Thousands of samples are needed to generate a robust gene list for predicting outcome in cancer. *Proc Natl Acad Sci U S A*, **103**, 5923–5928.

Fan, J., and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, **96**, 1348-1360.

Fan, J., and Li, R. (2006). Statistical challenges with high dimensionality: feature selection in knowledge discovery. *Proceedings of the International Congress of Mathematicians*, Spain, Madrid.

Fan, S., and Puthenpura, S. (1993). *Linear optimization and extentions: Theory and Algorithms*, Prentice Hall, New Jersey.

Frolov, A.E., Godwin, A.K., and Favorova, O.O. (2003). Differential gene expression

analysis by DNA microarray technology and its application in molecular oncology. *Mol Biol*, **37**, 486–494.

Furey, T., Cristianini, N., Duffy, N., *et al.* (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, **16**, 906–914.

Hastie, T., Tibshirani, R., and Friedman, J.H. (2001). *The Elements of Statistical Learning*. New York: Springer 2001.

Hackl, H., Cabo, F.S., Sturn, A., *et al.* (2004). Analysis of DNA microarray data. *Current Topics in Medicinal Chemistry*, **4**, 1355–1368.

Holmkvist, J., Almgren, P., Lyssenko, V., *et al.* (2008). Common Variants in Maturity-Onset Diabetes of the Young Genes and Future Risk of Type 2 Diabetes. *Diabetes*, **57 (6)**, 1738–1744.

Kanehisa, M., Goto, S., Kawashima, S., *et al.* (2004). The KEGG resource for deciphering the genome. *Nucleic Acids Res Database Issue*, **32**, D277–D280.

Lee, T.I., Rinaldi, N.J., Robert, F., *et al.* (2002). Transcriptional Regulatory Networks in Saccharomyces cerevisiae. *Science*, **298**, 799–804.

Li, C., and Li, H. (2008). Network-constrained regularization and variable selection for analysis of genomic data. *Bioinformatics*, **24**, 1175–1182.

Liu, M., Liberzon, A., Kong, S.W., *et al.* (2007). Network-based analysis of affected biological processes in type 2 diabetes models. *PLoS Genet*, **3**, 958–972.

Liu, S., Shen, X., and Wong, W. (2005). Computational developments of ψ-learning. *Proceedings of The Fifth SIAM International Conference on Data Mining*, Newport, California, April, 2005.

Nannini, M., Pantaleo, M.A., Maleddu, A., *et al.* (2009). Gene expression profiling in colorectal cancer using microarray technologies: results and perspectives. *Cancer Treatment Reviews*, **35(3)**, 201–209.

Nowacka-Zawisza, M., Brys, M., Romanowicz-Makowska, H. (2008). Dinucleotide repeat polymorphisms of RAD51, BRCA1, BRCA2 gene regions in breast cancer. *Pathology International*, **58 (5)**, 275–281.

Pan, W., Xie, B., and Shen, X. (2009). Incorporating predictor network in penalized regression with application to microarray data. *Biometrics*, to appear.
Available at `http: http://www.biostat.umn.edu./rrs.php`. as Research Report 2009-001, Division of Biostatistics, University of Minnesota.

Patra, B., Parsian, Z., Racette, B.A., *et al.* (2009). LRRK2 gene G2019S mutation and SNPs [haplotypes] in subtypes of Parkinson's disease. *Parkinsonism and Related Disorders*, **15(3)**, 175–180.

Peri, S., Navarro, J.D., Kristiansen, T.Z., *et al.* (2004). Human Protein Reference Database as a discovery resource for proteomics. *Nucleic Acids Res Database Issue*,

32, D497–D501.

Santos, E.S., Blaya, M., and Raez, L.E. (2009). Gene expression profiling and non-small-cell lung cancer: where are we now? *Clinical Lung Cancer*, **10(3)**, 168–173.

Scherzer, C.R., Eklund, A.C., Morse, L.J., *et al.* (2007). Molecular markers of early Parkinson's disease based on gene expression in blood. *Proc Natl Acad Sci U S A*, **104**, 955–960.

Soussi, T., and Béroud, C. (2003). Significance of TP53 mutations in human cancer: a critical analysis of mutations at CpG dinucleotides. *Human Mutation*, **21**, 192–200.

Sun, Z., Yang, P., Aubry, M.C., *et al.* Can gene expression profiling predict survival for pateints with squamous cell carcinoma of the lung? *Molecular Cancer*, **3:35**.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory.* New York: Springer.

Vermeirssen, V., Barrasa, M.I., Hidalgo, C.A., *et al.* (2007). Transcription factor modularity in a gene-centered C. elegans core neuronal proteinCDNA interaction network. *Genome Research*, **17**, 1061-1071.

Wahba, G., Lin, Y., and Zhang, H. (2000). GACV for support vector machines. In *Advances in Large Margin Classifiers.* Edited by Smola, A., Bartlett, P., Scholkopf, B., and Schuurmans, D. Cambridge, MA: MIT Press 2000, 297–311.

Wang, L., and Shen, X. (2007). On L_1-Norm multiclass support vector machines: Methodology and theory. *Journal of the American Statistical Association*, **102**,

583–594.

Wang, L., Zhu, J., and Zou, H. (2006). The doubly regularized support vector machine. *Stat Sin*, **16**, 589–615.

Wang, Y., Klijin, J.G., Zhang, Y., *et al.* (2005). Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *The Lancet*, **365**, 671–679.

Wu, S., Shen, X., and Geyer, C. (2008). Adaptive regularization through entire solution surface. *Biometrika*, to appear.

Xiong, M.M., Li, W.J., Zhao, J.Y., *et al.* (2001). Feature (gene) selection in gene expression-based tumor classification. *Mol Genet Metab*, **73**, 239–247.

Yang, T.Y. (2007). The simple classification of multiple cancer types using a small number of significant genes. *Mol Diagn Ther*, **11**, 265–275.

Zhao, P., Rocha, G., and Yu, B. The Composite Absolute Penalties Family for Grouped and Hierarchical Variable Selection *The Annals of Statistics*, to appear.

Zou, H., and Hastie, T. (2005). Regularization and variable selection via the elastic net. *J R Statist Soc B*, **67**, 301–320.

Zou, H., and Yuan, M. (2008). The F_∞-norm Support Vector Machine. *Stat Sin*, **18**, 379–398.

Zhu, J., Rosset, S., Hastie, T., and Tibshirani, R. (2003). 1-norm support vector machines. *The Annual Conference on Neural Information Processing Systems*, MIT Press, 16.

Zhu, Y., Shen, X., and Pan, W. (2009). Network-based support vector machine for classification of microarray samples. *BMC Bioinformatics*, **10**, S21.

Gene Expression Omnibus: GSE6613. [http://www.lncbi.nlm.nih.gov/geo/].

KEGG: Parkinson's disease. [http://www.genome.ad.jp/kegg/pathway/hsa/hsa05020.html].