Learning with Kernels and Graphs to Understand Cancer DNA Copy Number Variations

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Ze Tian

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

RUI KUANG

July, 2012

Acknowledgements

There are many people that have earned my gratitude for their contribution to my time in graduate school.

First and foremost, I would like to express my sincere gratitude to my advisor Dr. Rui Kuang. It has been an honor to be his Ph.D. student. I appreciate all his contributions of time, ideas and funding to make my Ph.D. experience productive and stimulating. His guidance helped me a lot in my research and writing of this thesis. This thesis would not have been possible without his support and encouragement.

The members of the Computational Biology Group have contributed immensely to my personal and professional time at University of Minnesota. I would like to acknowledge two former group members Taehyun Hwang and Changjin Hong, who I have had the pleasure to work with in several research projects. I would also like to thank former and current group members Huanan Zhang, Wei Zhang, Maoqiang Xie, Baryun Hwang and Nicholas Johnson who supported me in various ways.

My sincere thanks also go to Dr. Dengyong Zhou and Dr. Xiaodong Fan, who I have worked with during my summer internship. Their supervision and support truly helped the progress and smoothness of my internship program.

For this thesis I would like to thank my committee members Dr. Vipin Kumar, Dr. Chad Myers and Dr. Baolin Wu for their time, interest, insightful questions and helpful comments.

Last, I gratefully acknowledge the BICB traineeship program which funded me for the first two years of my Ph.D. program.

Abstract

DNA copy number variations (CNVs) are biological indicators that characterize cancer genomes. Predicting the prognosis of cancer from CNVs and identifying cancer-causing CNVs is a challenging problem due to the high dimensionality of the CNV features and the heterogeneity of patients. In this thesis, our objective is to build robust predictive models based on CNV data using machine learning techniques for accurate cancer diagnosis and prognosis, as well as for the identification of cancer-causing CNVs.

We proposed several machine learning models towards these objectives: 1. We developed a hypergraph-based semi-supervised learning algorithm *HyperPrior* for cancer outcome prediction from CNV data and gene expression data. It incorporates biological prior knowledge such as the spacial information in arrayCGH datasets to get consistent weighting on correlated genomic features, thus to improve the accuracy of the model in sample classification. In addition, the algorithm can also be used for biomarker or cancer-causing CNV detection; 2. We developed an alignment-based kernel method for integrating CNV data from multiple platforms. By integrating datasets generated from different probe sets, the new kernel could improve the cancer outcome prediction by the SVM classifier. Furthermore, we also designed a multiple alignment approach based on our alignment kernel to identify shared CNVs among cancer samples, which served as candidates of cancer-causing CNVs for further analysis; 3. We proposed an algorithm to learn a low-rank graph to represent the similarities between data points. This low-rank graph could capture the global cluster structures and improve the performance of label propagation. The whole approach can be applied to arrayCGH datasets as well as other types of datasets for better sample classification results; 4. We proposed a latent feature model that couples sparse sample group selection with fused lasso. Clinical information was used to define the group structure on patient samples. By sparse group selection, the model was able to identify group-specific CNVs instead of common CNVs from arrayCGH datasets.

We used both simulations and several publicly available genomic datasets to evaluate our models. The results suggest that these models are promising in achieving better cancer prognosis prediction and identification of cancer-causing CNVs.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

There are normally two copies of each gene in the double-stranded DNAs of human genome. However, duplications and deletions of DNA sequences can lead to a different number of copies of the genes in the DNA. If a DNA region is deleted in one or both strands, there will be a fewer number of copies of the genes and on the contrary if a DNA region is duplicated, there will be a larger number of copies of the genes. This variability in human genomes is often referred as DNA copy number variation (CNV). Recent research reveals that CNV is a major cause of structural variation in the human genome. For example, CNV can affect the gene expression levels, alter the organization of chromatin and influence the regulation of nearby genes [1]. It has been confirmed in many recent studies that chromosomal aberrations of DNA copy numbers, rearrangement and structures have association with cancer and other diseases [2]. Among these aberrations, DNA copy number variations (CNVs) are believed to play an important role in tumorigenesis [3, 4]. For example, Alzheimer disease, autism, Crohn disease, HIV susceptibility, mental retardation, pancreatitis and Parkinson disease are all associated with CNVs [5]. Since CNVs play important roles in human disease, it is vital to identify disease-associated CNVs which are crucial for understanding the molecular mechanisms underlying cancer and might be used for improving the diagnosis and prognosis of patients.

Currently DNA copy numbers can be measured by three different techniques in biology experiments, which are array comparative genomic hybridization (arrayCGH), single nucleotide polymorphism (SNP) genotyping and next-generation sequencing. Comparative genomic hybridization (CGH) compares the copy number of a differentially labeled case sample with a normal reference DNA. ArrayCGH technology based on DNA microarray can currently allow genome-wide identification of CNV regions by CGH measuring at a number of sampled locations at different resolutions [6]. ArrayCGH data provide important information of candidate cancer loci for the classification of patients and discovery of molecular mechanisms of cancers [7]. Since most publicly available DNA copy number datasets were generated by the first technique, we just focused our study on arrayCGH datasets in this thesis. However, the proposed models in this thesis can also be applied on datasets generated with the other two techniques.

1.1 Challenges and Objectives

Many methods were proposed to study the associations between phenotypes and genotypes from various genomic data. For example, [8] proposed a graph-based semi-supervised learning algorithm to identify discriminative disease markers from gene expression data. However, for most genomic datasets, the number of features is much larger than the number of samples, which causes the "curse of dimensionality" in machine learning problems. Specifically, the results generated by general methods from a small number of samples might be unstable and unreliable. For example, in the study of breast cancer, [9] and [10] identified two sets of marker genes related to the metastasis of breast cancer using large scale gene expression profiles produced in two different microarray experiments. However, there are only three genes in common between the two sets of the marker genes. Furthermore, there are three other problems in arrayCGH data analysis:

1. The features in arrayCGH datasets are not independent of each other. Actually, the features have their biological meanings and are correlated. It is challenging to integrate their biological prior information in a general machine learning task such as classification and feature selection.

2. ArrayCGH datasets were generated from different platforms, which means they

might use different probe sets to measure the copy numbers. Since the probes were located at different positions, it is a challenging problem to integrate arrayCGH samples from different platforms.

3. There are discrepancies introduced by the heterogeneity in the patient samples. These discrepancies might be useful to distinguish disease-associated CNVs from common CNVs. However, it is challenging to incorporate this information with general CNV identification methods.

In this thesis, we proposed several graph and kernel based machine learning methods for the analysis of arrayCGH data. The goal of our methods is to build robust and reliable models for cancer outcome prediction and disease-associated CNV identification. The algorithms developed in the thesis provided general tools for CNV analysis and the results from the experiments in the thesis provided candidates of biomarkers for further in depth analysis by oncologists.

1.2 Contribution

To address the challenges described in section 1.1, we proposed four different models in this thesis.

First, to incorporate biological prior knowledge into predictive models for data integration, we introduced a hypergraph-based semi-supervised learning algorithm called *HyperPrior* [11, 12] to classify gene expression and arrayCGH data using biological knowledge as constraints on graph-based learning. *HyperPrior* is a robust twostep iterative method that alternatively finds the optimal labeling of the samples and the optimal weighting of the features, guided by constraints encoding prior knowledge. The prior knowledge for analyzing arrayCGH data is that probes that are spatially nearby in their layout along the chromosomes tend to be involved in the same amplification or deletion event. Based on the prior knowledge, *HyperPrior* imposes a consistent weighting of the correlated genomic features in graph-based learning.

Second, to perform integrated classification and analysis across multiple arrayCGH datasets and solve the discrepancy in probe setting by different platforms, we proposed an alignment based framework [13] to integrate arrayCGH samples generated from different probe sets. The alignment framework seeks an optimal alignment between the probe series of one arrayCGH sample and the probe series of another sample, intended to find the maximum possible overlap of DNA copy number variations between the two measured chromosomes. An alignment kernel is introduced for integrative patient sample classification and a multiple alignment algorithm is also introduced for identifying common regions with copy number aberrations.

Then, to build more reliable pairwise similarity relations between samples with high dimensional features for label prorogation, we proposed *GLNP* [14] to learn a nonnegative low-rank graph to capture global linear neighborhoods, under the assumption that each data point can be linearly reconstructed from weighted combinations of its direct neighbors and reachable indirect neighbors. In contrast to the general sparse representation of graphs which might suffer from disjoint components and incorrect neighbors, the global linear neighborhoods utilize information from both direct and indirect neighbors to preserve the global cluster structures, while the low-rank property retains a compressed representation of the graph. An efficient algorithm based on a multiplicative update rule is designed to learn a nonnegative low-rank factorization matrix minimizing the neighborhood reconstruction error. This semi-supervised learning approach can be applied to classification problems on arrayCGH datasets.

Last, to detect DNA CNVs from arrayCGH or genotyping-array data to correlate with cancer outcomes, we proposed a latent feature model that couples sparse sample group selection with fused lasso on CNV components to identify group-specific CNVs. Assuming a given group structure on patient samples by clinical information, *SGS-FL* (Sparse Group Selection on Fused Lasso) identifies the optimal latent CNV components, each of which is specific to the samples in one or several groups. The group selection for each CNV component is determined dynamically by an adaptive algorithm to achieve a desired sparsity.

1.3 Outline

The rest of the thesis will be organized into five chapters:

- In Chapter 2, we described the *HyperPrior* algorithm to integrate genomic data with general biological prior knowledge for robust cancer sample classification and biomarker identification.

- In Chapter 3, we described the alignment kernel to integrate arrayCGH datasets from different platforms for sample classification and common CNV identification.
- In Chapter 4, we described the *GLNP* algorithm to capture global linear neighborhoods from data for more accurate label propagation and semi-supervised learning.
- In Chapter 5, we described the *SGS-FL* model to identify group-specific CNVs.
- Finally, we summarized all these algorithms and models and then discussed possible future work in Chapter 6.

Chapter 2

Hypergraph-based Learning with Prior Knowledge

In this chapter, we propose a hypergraph-based iterative learning algorithm called *HyperPrior* to integrate genomic data with general biological prior knowledge for robust cancer sample classification and biomarker identification.

2.1 Introduction

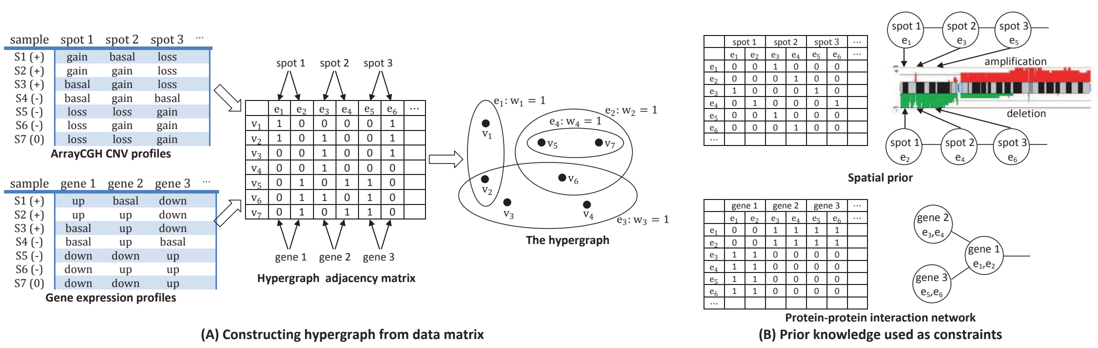
In the past decade, numerous cancer researchers have investigated high-throughput genomic data actively to reveal the molecular mechanisms underlying cancer development and progression. DNA copy number variations (CNVs) measured by array-based comparative genomic hybridization (arrayCGH) and microarray gene expressions are among the most widely studied high-throughput data [15]. Microarray gene expressions provide a genome-wide quantification of messenger RNA abundance, while arrayCGH data quantify the events of amplification or deletion of large DNA segments on chromosomes. A large number of high-resolution arrayCGH datasets and gene expression datasets were generated to study many different cancers [9, 16, 17, 18]. In these studies, two major objectives were

1. Detecting highly discriminative chromosomal copy number aberration regions or gene expression patterns as biomarkers of cancer-relevant phenotypes

2. Building reliable predictive models based on the biomarkers for cancer sample classification

Although many interesting and promising findings were reported in these studies, concerns have been raised on the unstable and inconsistent results in cross-validations and cross-platform comparisons due to the relatively small sample sizes in the studies [19]. To address the problem, researchers have proposed including other complementary genomic information such as pathways or functional annotations to aid model building and biomarker discovery. It is believed that the prior knowledge from complementary data can generate more robust models and more consistent discoveries across independent studies. For gene expression profiles, the availability of large protein-protein interaction networks, which contain information on gene functions, pathways and modularity of gene regulations, provides a desirable source of data for the purpose [20, 21, 22]. In arrayCGH data, microarray comparative genomic hybridization measures copy number information distributed along the genome at different resolutions. This information typically includes thousands of spot intensities. Intuitively, neighboring spots on the chromosomes tend to be highly correlated because a DNA aberration can expand to neighboring intervals [23]. It has also been shown that the spatial information of probes on chromosomes can also be used as prior knowledge to improve classification [23]. However, designing a unified strategy to integrate gene expressions with protein-protein interactions or to integrate arrayCGH data with the chromosomal spatial information is still a challenging data integration problem, since standard classification and feature selection methods do not meet the complexity of a joint learning on two different data types.

In this thesis, we propose a hypergraph-based iterative learning algorithm called *HyperPrior* to integrate genomic data with general biological prior knowledge for robust cancer sample classification and biomarker identification. The *HyperPrior* algorithm minimizes a cost function under a unified regularization framework. This framework elegantly takes biological prior knowledge (e.g. a correlation structure of spot regions or a protein-protein interaction network) as constraints on a hypergraph built from genomic data. *HyperPrior* is a natural extension of label propagation algorithms on hypergraphs [24, 25]. This algorithm helps handle the problem of learning optimal weighting of the hyperedges while all other methods assume a uniform weighting. To model the genomic



sample	spot 1	spot 2	spot 3	···
S1 (+)	gain	basal	loss	
S2 (+)	gain	gain	loss	
S3 (+)	basal	gain	loss	
S4 (-)	basal	gain	basal	
S5 (-)	loss	loss	gain	
S6 (-)	loss	gain	gain	
S7 (0)	loss	loss	gain	

ArrayCGH CNV profiles

sample	gene 1	gene 2	gene 3	···
S1 (+)	up	basal	down	
S2 (+)	up	up	down	
S3 (+)	basal	up	down	
S4 (-)	basal	up	basal	
S5 (-)	down	down	up	
S6 (-)	down	up	up	
S7 (0)	down	down	up	

Gene expression profiles

	e_1	e_2	e_3	e_4	e_5	e_6	···
v_1	1	0	0	0	0	1	
v_2	1	0	1	0	0	1	
v_3	0	0	1	0	0	1	
v_4	0	0	1	0	0	0	
v_5	0	1	0	1	1	0	
v_6	0	1	1	0	1	0	
v_7	0	1	0	1	1	0	

Hypergraph adjacency matrix

	spot 1		spot 2		spot 3		···
	e_1	e_2	e_3	e_4	e_5	e_6	···
e_1	0	0	1	0	0	0	
e_2	0	0	0	1	0	0	
e_3	1	0	0	0	1	0	
e_4	0	1	0	0	0	1	
e_5	0	0	1	0	0	0	
e_6	0	0	0	1	0	0	
···							

Spatial prior

	gene 1		gene 2		gene 3		···
	e_1	e_2	e_3	e_4	e_5	e_6	···
e_1	0	0	1	1	1	1	
e_2	0	0	1	1	1	1	
e_3	1	1	0	0	0	0	
e_4	1	1	0	0	0	0	
e_5	1	1	0	0	0	0	
e_6	1	1	0	0	0	0	
···							

Protein-protein interaction network

Figure 2.1: **Overview of *HyperPrior* framework**. **(A) Modeling genomic data by a hypergraph.** The original arrayCGH data or gene expression data is first converted to a weighted adjacency matrix. Since the weights in the adjacency matrix must be non-negative, for each genomic feature two complimentary features are generated to deal with the negative values. For example, the intensity of arrayCGH spot 1 (or the expression of gene 1) is split into two features, e_1 and e_2. For illustration purpose, the adjacency matrix in the example is binary. More generally real values can be kept from the original data in the matrix to preserve all information from the original data. A hypergraph is then built from the adjacency matrix. For better clarify, hyperedges e_5 and e_6 are omitted in the hypergraph. **(B) Modeling prior knowledge as a graph constraint.** The spatial relation or PPI network is used to derive relations between hyperedges as a prior graph. The relations define the constraints for hypergraph-based learning. Note that each gene expression corresponds to two features (up/down regulations). The two features of a gene will have a relation with the two features of another gene interacting with the first gene. For example, e_1 and e_2 are both associated with both e_3 and e_4 because gene 1 and gene 2 interact in the PPI.

data as a hypergraph, each sample is denoted by a vertex and each feature is denoted by two hyperedges corresponding to two states of the feature. In arrayCGH data, each spot intensity is represented by two hyperedges labeled as "amplification" and "deletion" of the associated DNA segment, respectively. In gene expression data, each gene expression value is denoted by two hyperedges labeled as "up-regulated" and "down-regulated", respectively. The hyperedges categorize samples by the two different states of features in the genomic data (Figure 2.1(A)). Our cluster assumption on the hypergraph is that the same type of samples tend to have similar states of features and thus are highly connected by the hyperedges. *HyperPrior* formulates optimization problems as learning labels and hyperedge weights together with the assignment of edge weights constrained by the biological relation between the genomic features (Figure 2.1(B)). Specifically, *HyperPrior* attempts to find a weighting of hyperedges that balances both the two-class separation on the hypergraph and the consistency with the biological constraints. The assumption is that neighboring spots or genes interacting with each other are more likely to receive similar weights. The resultant weights on the genomic features can be used to discover biologically interpretable biomarkers. Specifically, the highly weighted DNA aberration regions may suggest cancer-relevant DNA amplification and deletion events, and the highly weighted genes in densely connected subnetworks in a protein-protein interaction network may suggest relevant cancer pathways. These properties of the *HyperPrior* algorithm promise to improve prediction accuracy and provide more robust identification of discriminative biomarkers.

2.2 Related Work

Integrating multiple genomic data types for building predictive models or selecting features has been an important research focus in bioinformatics since several years ago [26, 27]. For example, [28] proposed a two layered Bayesian network approach to integrate relations from gene expressions, biological literature and gene sequences into a genome-wide functional network. [29] proposed a novel method to integrate gene-gene (or mRNA) relations and sample relations for gene selection. This method focuses on discovering geometric patterns to select genes with biological relevance and statistical significance.

DNA copy number is also an important data type for building robust predictive models. For example, previous studies by [30] and [31] used DNA copy numbers for cancer discriminant analysis. However, these methods ignore the spatial correlation among the sampling spots in arrayCGH data, and thus suffer from low accuracy and poor interpretability. Under the motivation of data integration, several other general computational methods were also proposed to use prior knowledge in classifying arrayCGH data and gene expression data [32, 22, 23, 20]. [23] proposed a variant of L_1-SVM called fused SVM for classifying arrayCGH data. By incorporating the spatial relation among DNA copy number variations along the genome as prior knowledge, new constraints are introduced into L_1-SVM learning. They demonstrated that the new algorithm achieved better classification accuracy on two cancer datasets. [20] improved cancer outcome prediction and biomarker reproducibility on two large scale gene expression datasets by incorporating a protein-protein interaction network into the model built from microarray gene expressions. In their approach, the integration of gene expressions and protein-protein interactions is achieved by two independent procedures: discriminative subnetworks are first identified from the PPI network and the subnetworks are then used as features to predict cancer metastasis. [22] proposed a similar method, which first computes the spectral graph structure of a gene network, and then uses the spectral graph structure to smooth microarray gene expressions before used for sample classification. [21] proposed a statistical method to score genes by several measures including their degree in a cancer-specific interaction network, their differential expressions in microarray data and their structural, functional and evolutionary properties. [32] proposed to add a graph Laplacian constraint to the L_1-norm linear regression model in a general regularization framework. The graph Laplacian encodes a network of known KEGG pathways. The new model can be efficiently solved by methods for lasso-type problems.

HyperPrior is different from the previous methods in both problem formulation and model implication. The regularization framework of *HyperPrior* is designed for simultaneously classifying samples and selecting features based on prior knowledge. *HyperPrior* explores the cluster structures in both the hypergraph and the prior graph in one unified learning framework. Thus, the learning problem is a combination of semi-supervised learning and variable selection. In section 2.3.1, we will also show that the core of *HyperPrior* is to learn a "kernel" for a diagonal linear transformation of the data along with learning labels on samples. Thus, *HyperPrior* can be interpreted as a novel semi-supervised and relaxed wrapper-feature-selection method constrained by prior knowledge.

2.3 Methods

In this section, we first introduce the *HyperPrior* algorithm and its regularization framework. We then describe how to model arrayCGH data with spatial prior knowledge and gene expression data with prior knowledge in a protein-protein interaction network by constrained hypergraphs.

2.3.1 *HyperPrior* algorithm

Notations

A hypergraph is a special graph that contains hyperedges. In a normal graph, each edge connects a pair of vertices, but in a hypergraph each edge can connect an arbitrary number of vertices in the graph. Let $V = \{v_1, v_2, \ldots, v_{|V|}\}$ be a set of vertices and $E = \{e_1, e_2, \ldots, e_{|E|}\}$ be a set of hyperedges defined on V: for any hyperedge $e \in E$, $e = \{v_1^{(e)}, v_2^{(e)}, \ldots, v_{|e|}^{(e)}\}$, where $\{v_1^{(e)}, v_2^{(e)}, \ldots, v_{|e|}^{(e)}\}$ is a subset of V. A hyperedge e and a vertex v are called incident if $v \in e$. A non-negative real number (a weight) can be assigned to each hyperedge by a function w (w can also be defined as a vector variable and we will use both notations interchangeably). The vertex set V, hyperedge set E and the weight function w fully defines a weighted hypergraph denoted by $G(V, E, w)$. The incidence matrix H for hypergraph $G(V, E, w)$ is a $|V| \times |E|$ matrix with elements defined as $h(v, e) = 1$ (or a real value if H is weighted) when $v \in e$ and 0 otherwise. The degree of a vertex v is defined as $d(v) = \sum_{e \in E} h(v, e)w(e)$, which is the (weighted) sum of the weights of the hyperedges incident with v. The degree of a hyperedge e is defined as $d(e) = \sum_{v \in E} h(v, e)$, which is the number of vertices incident with e. We define diagonal matrices D_v=diag($d(v_1)$,$d(v_2)$,...,$d(v_{|V|})$), D_e=diag($d(e_1)$,$d(e_2)$,...,$d(e_{|E|})$) and W=diag($w(e_1)$,$w(e_2)$,..., $w(e_{|E|})$).

Let $G(V, E, w)$ be a weighted hypergraph to model the genomic data: each patient sample is denoted by a vertex $v \in V$ and each hyperedge denotes one of the two states (+/-) of a genomic feature. The incidence matrix H between V and E are determined by the copy number variation log-ratios or gene expression intensities on the samples. We define a function y to assign initial labels to V in the hypergraph $G(V, E, w)$. If a vertex v is in the positive patient group, $y(v) = +1$; if it is in the negative patient group, $y(v) = -1$; and, if v is a test sample, $y(v) = 0$.[1]

[1] To normalize unbalanced datasets in our experiments, we set $y(v) = \frac{1}{n_1}$ for positive vertices and $y(v) = -\frac{1}{n_2}$ for negative vertices, where n_1 is the number of positive samples and n_2 is the number of negative samples.

Hypergraph-based learning

In hypergraph-based learning, our goal is to find the correct labels for the unlabeled vertices of the test samples in the hypergraph. Let f be the objective function (vector) of labels to be learned. Intuitively, there are two criteria for learning the optimal f:

1. We want to assign the same label to vertices that share many incidental hyperedges in common.

2. Assignment of the labels should be similar to the initial labeling y.

For criteria 1, we define the following cost function

$$\begin{aligned}\Omega(f,w) &= \frac{1}{2}\sum_{e\in E}\sum_{u,v\in e}\frac{w(e)h(u,e)h(v,e)}{d(e)}\left(\frac{f(u)}{\sqrt{d(u)}}-\frac{f(v)}{\sqrt{d(v)}}\right)^2 \\ &= f^T(I-D_v^{-\frac{1}{2}}HWD_e^{-1}H^TD_v^{-\frac{1}{2}})f, \qquad (2.1)\end{aligned}$$

where I is the identity matrix. Here, D_v and D_e are used for computing the normalization of the hypergraph Laplacian, and the unnormalized hypergraph Laplacian is $diag(HD_ew) - HWH^T$. Empirical results showed that the normalized form gives better classification performance for graph-based learning [25]. If the predicted labels on the vertices are consistent with the incidences with the hyperedges, the value of $\Omega(f, w)$ should be minimized. For criteria 2, we directly calculate the squared-loss between the predicted labeling f and the original labeling y as follows:

$$\sum_{u\in V}(f(u)-y(u))^2 = \|f-y\|^2 .$$

Incorporating prior knowledge

To introduce prior knowledge into the hypergraph-based learning, we assume that correlating genomic features should receive similar weights on their associated hyperedges. We define two different functions to encode the prior knowledge. The first function $\Psi_{lp}(w)$ is a network-Laplacian constraint [32]. We define an indicator $\delta_{i,j}$ to capture the pairwise relation between hyperedges e_i and e_j. The indicator $\delta_{i,j} > 0$ if the two genomic features associated with e_i and e_j are correlated in the prior knowledge; otherwise 0. Let Δ be the correlation matrix with $\Delta_{i,j} = \delta_{i,j}$, $\sigma(e_i) = \sum_{j=1}^{|E|}\delta_{i,j}$, which is the number of hyperedges that are correlated with e_i, and D_σ=diag$(\sigma(e_1), \sigma(e_2), ..., \sigma(e_{|E|}))$. To assign similar weights to correlated hyperedges, we define the following cost function over the hyperedge weights:

$$\begin{aligned}\Psi_{lp}(w) &= \frac{1}{2}\sum_{i,j=1}^{|E|} \delta_{i,j}\left(\frac{w(e_i)}{\sqrt{\sigma(e_i)}} - \frac{w(e_j)}{\sqrt{\sigma(e_j)}}\right)^2 \\ &= w^T(I - D_\sigma^{-\frac{1}{2}}\Delta D_\sigma^{-\frac{1}{2}})w. \end{aligned} \tag{2.2}$$

Minimizing $\Psi_{lp}(w)$ ensures that correlated hyperedges will be similarly weighted. The second function $\Psi_{nb}(w)$ is a neighborhood constraint [33]. Instead of directly evaluating the pairwise relation between hyperedges e_i and e_j, we require the weight of e_i to be close to the average weight of its neighbors. To assign weights to hyperedges that are consistent with such prior knowledge, we define the following cost function over the hyperedge weights:

$$\begin{aligned}\Psi_{nb}(w) &= \frac{1}{2}\sum_{i=1}^{|E|}\left(w(e_i) - \sum_{j=1}^{|E|}\frac{\delta_{i,j}}{\sigma(e_i)}w(e_j)\right)^2 \\ &= w^T(I - D_\sigma^{-1}\Delta)(I - D_\sigma^{-1}\Delta)^T w. \end{aligned} \tag{2.3}$$

Minimizing $\Psi_{nb}(w)$ ensures that each hyperedge will receive a weight close to the average weight of its correlated hyperedges. Both $\Psi_{lp}(w)$ and $\Psi_{nb}(w)$ are regularization terms to explore the network structure of the correlations among the variables in the function. But the two functions are different in their cluster assumptions: either penalizing discrepancy in pairwise similarity or neighborhood similarity.

Alternating optimization

After the prior knowledge is introduced, the learning task is to minimize the sum of the three cost terms, which is

$$\underset{f,w}{\text{minimize}} \;\; \Phi(f, w) = \Omega(f, w) + \mu \|f - y\|^2 + \rho\Psi(w) \tag{2.4}$$

subject to

$$w(e) \geq 0 \text{ for } \forall e \in E$$
$$\sum_{e \in E} h(v, e)w(e) = d(v) \text{ for } \forall v \in V,$$

where μ and ρ are positive real numbers and $\Psi(w) = \Psi_{lp}(w)$ or $\Psi_{nb}(w)$. The intuition of adding $\sum_{e\in E} h(v,e)w(e) = d(v)$ as another set of constraints is to maintain the hypergraph structure. This set of constraints can guarantee each $d(v)$ is fixed as a constant such that $\Omega(f,w)$ is always a linear function of w when f is fixed. In the unnormalized form of the hypergraph Laplacian, the constraint $\sum_{e\in E} h(v,e)w(e) = d(v)$ is not required, and a simple lower bound $\sum_{e\in E} w(e) = \tau$ ($\tau > 0$) can be used.

The objective function $\Phi(f,w)$ in Eqn. (2.4) is cubic in $(f,\, w)$. However, the formulation contains two sub-problems, both of which are quadratic convex problems if we independently optimize $\Phi(f,w)$ with respect to f or w. Specifically, if we fix w to be a specific weighting w_t satisfying the constraints $w_t(e) \geq 0$ for $\forall e \in E$ and $\sum_{e\in E} h(v,e)w_t(e) = d(v)$ for $\forall v \in V$, the objective function $\Phi(f, w = w_t)$ is convex in f; if we fix f to be a specific labeling of the vertices f_t, $\Phi(f = f_t, w)$ is also convex in w. So the algorithm can be implemented by alternating the following two steps:

1. First, the *HyperPrior* algorithm initializes w with a uniform weighting 1 over the hyperedges. Note that $w = 1$ is a solution to the linear system $Hw = diag(D_v)$ by the definition of D_v and thus, a valid solution to minimize$_{f,w}$ $\Phi(f,w)$. In the first step in each iteration, *HyperPrior* fixes w and optimizes $\Phi(f, w = w_t)$ with respect to f in the following optimization problem:

$$\underset{f}{\text{minimize}}\ \Omega(f, w = w_t) + \mu \left\| f - y \right\|^2 . \tag{2.5}$$

The cost term $\Psi(w = w_t)$ is removed from $\Phi(f, w = w_t)$ since it is a constant in the above optimization problem. Let $L = I - D_v^{-1/2} H W D_e^{-1} H^T D_v^{-1/2}$. In the cost term, we can prove $\Omega(f, w = w_t) = f^T L f$ (section 2.3.1). L is positive semi-definite given $\Omega(f, w = w_t) \geq 0$ for any f, which also implies that $\Omega(f, w = w_t)$ is convex in f. Therefore, we can simply take derivative with respect to f to get the optimal solution $f^* = (1-\alpha)((1-\alpha)I + \alpha L)^{-1} y$, where $\alpha = \frac{\mu}{1+\mu}$ [25]. This is equivalent to solving the linear system $(1-\alpha)((1-\alpha)I + \alpha L)f = y$, which can be efficiently computed by Jacobi iteration method [34].

2. In the second step in each iteration, the *HyperPrior* algorithm fixes $f = f_t$ learned in the previous step to learn the optimal weighting of hyperedges w by solving the

Algorithm 1 HyperPrior

```
Input: $y, H, \Delta, \mu, \rho$
Output: $f_t, w_t$
1: $t = 0, w_0 = 1, f_0 = y, c_0 = +\infty$
2: repeat
3:     $t = t + 1$
4:     Use network propagation to find optimal $f_t$
       $f_t = (1-\alpha)(I - \alpha D_v^{-1/2} H W_{t-1} D_e^{-1} H^T D_v^{-1/2})^{-1} y$
5:     Use quadratic programming to find optimal $w_t$
       $w_t = \text{argmin}_w \; \Omega(f = f_{t-1}, w) + \rho\Psi(w)$
       subject to $Hw = diag(D_v)$ and $diag(W) \succeq 0$
6:     $c_t = \Omega(f_t, w_t) + \mu||f_t - y||^2 + \rho\Psi(w_t)$
7: until $c_{t-1} - c_t \leq \pi$
8: return $f_t, w_t$
```

quadratic programming problem:

$$\underset{w}{\text{minimize}} \;\; \Omega(f = f_t, w) + \rho\Psi(w) \tag{2.6}$$

subject to

$$w(e) \geq 0 \text{ for } \forall e \in E$$
$$\sum_{e \in E} h(v,e)w(e) = d(v) \text{ for } \forall v \in V.$$

The cost $\mu \|f - y\|^2$ is removed from $\Phi(f, w = w_t)$ since it is a constant in the above optimization problem, and $\Omega(f = f_t, w)$ is a linear function of w. Since $\Psi(w) = w^T(I - D_\sigma^{-1/2}\Delta D_\sigma^{-1/2})w \geq 0$ for any w, $I - D_\sigma^{-1/2}\Delta D_\sigma^{-1/2}$ is positive semi-definite, which implies that $\Phi(f = f_t, w)$ is convex in w. Thus it is a convex quadratic programming problem with constraints which can be solved by standard convex optimization techniques.

In both steps, the total cost $\Phi(f, w)$ is guaranteed to be reduced until there is only very small change. Thus, our algorithm will finally stop at a small total cost. A local optimal solution can be found by solving the two optimizations alternatively by iteration [35], under the assumption that f and w can be independently optimized. The assumption does not guarantee a global optimal solution. The alternating optimization can be solved by an iterative algorithm proposed by us in [36], described in Algorithm 1. We implemented it in MATLAB and use ILOG/CPLEX package (version 11.1) for quadratic programming.

Proof of convexity

Let $L = I - D_v^{-1/2} H W D_e^{-1} H^T D_v^{-1/2}$, where I is the identity matrix and W is the diagonal matrix with $W_{ii} = w(e_i)$. We can show $\Omega(f, w) = f^T L f$ by

$$\begin{aligned}
\Omega(f, w) =& \sum_{e \in E} \sum_{u,v \in V} \frac{w(e)h(u,e)h(v,e)}{d(e)} \left(\frac{f^2(u)}{d(u)} - \frac{f(u)f(v)}{\sqrt{d(u)d(v)}} \right) \\
=& \sum_{e \in E} \sum_{u \in V} \frac{w(e)h(u,e)f^2(u)}{d(u)} \sum_{v \in V} \frac{h(v,e)}{d(e)} \\
&- \sum_{e \in E} \sum_{u,v \in V} \frac{w(e)h(u,e)h(v,e)}{d(e)} \frac{f(u)f(v)}{\sqrt{d(u)d(v)}} \\
=& \sum_{u \in V} f^2(u) \sum_{e \in E} \frac{w(e)h(u,e)}{d(u)} - \sum_{e \in E} \sum_{u,v \in V} \frac{f(u)w(e)h(u,e)h(v,e)f(v)}{\sqrt{d(u)d(v)}d(e)} \\
=& \sum_{u \in V} f^2(u) - \sum_{e \in E} \sum_{u,v \in V} \frac{f(u)w(e)h(u,e)h(v,e)f(v)}{\sqrt{d(u)d(v)}d(e)}.
\end{aligned}$$

Step three in the above derivation shows that $\Omega(f, w) = f^T L f$ if and only if

$$\sum_{e \in E} \frac{w(e)h(u,e)}{d(u)} = 1.$$

The constraints $\sum_{e \in E} h(v,e)w(e) = d(v)$ for $\forall v \in V$ keep D_v unchanged during the optimization and thus make L always positive semi-definite.

Time complexity

The time complexity of *HyperPrior* includes solving two minimization problems: fixing w to learn f and vice versa. The first problem can be solved by network propagation in $O(k_1|V||E|)$, where k_1 is the round of propagations [25]. The value of k_1 mainly depends on the eigenvalues of the Laplacian matrix. The second problem is a standard convex quadratic programming problem, which can be solved in polynomial time $O(|E|^p)$, where p is a real number. Thus, the time complexity of *HyperPrior* is $O(k_2(k_1|V||E| + |E|^p))$, where k_2 is the number of iterations of alternating optimization. Usually, $k_2 = 2$ or 3 in our experiments.

Model interpretation

In essence, the regularization framework of *HyperPrior* is a semi-supervised and relaxed wrapper-feature-selection method, which performs feature selection based on prior knowledge while classifying samples. A dissection of Eqn. (2.1) can explain the role of W in the learning framework. Given a (weighted) hypergraph incidence matrix H, we define its normalized adjacency matrix as $\bar{H} = D_v^{-\frac{1}{2}} H D_e^{-\frac{1}{2}}$. In the standard hypergraph-based learning framework [24], a linear kernel $K_{W=I}(V, V) = \bar{H} W \bar{H}^T$ is chosen to construct a similarity graph between objects used for semi-supervised learning in the normalized graph Laplacian $I - K_{W=I}(V, V)$, which is $(I - D_v^{-\frac{1}{2}} H W D_e^{-1} H^T D_v^{-\frac{1}{2}})$. Instead of fixing W to be the identity matrix, the *HyperPrior* framework treats $W^{\frac{1}{2}}$ as a diagonal linear transformation matrix for the original feature space. *HyperPrior* learns a $W^{\frac{1}{2}}$ to be used with linear kernel $K_W(V, V) = (\bar{H} W^{\frac{1}{2}})(\bar{H} W^{\frac{1}{2}})^T$. The *HyperPrior* framework derives an optimal W to generate the best labeling of the samples based on the prior knowledge given by $\Psi(w)$ ($w = diag(W)$). In general, W can be any linear transformation matrix. However, to make the learning problem tractable, we restrict W to be a diagonal matrix with positive weights of the features on the diagonal. If we further restrict the values in w to be binary (0 or 1), W is a projection matrix and K_W is a kernel for feature selection. Thus, with a binary w, the regularization framework is a model that performs wrapped feature selection based on prior knowledge for graph-based learning models. However, it is NP-hard to solve the integer programming problem. We relax w to be positive real values in our framework. Note that, to encourage sparse solution to feature selection, an 1-norm regularizer $|w|$ can be included and the new formulation can still be solved by the same method. However, the additional regularizer will also introduce one more hyper-parameter to tune.

The algorithm introduced by [22] was similarly motivated to search for a good projection W for data matrix H. But instead of learning the W, they directly derived the W from the eigen-decomposition of the graph Laplacian of the PPI network as a data preprocessing step. The *HyperPrior* framework attempts to solve both semi-supervised learning and wrapped feature selection together with an objective function on both f

and w. Compared with the lasso linear model introduced by [32], the joint learning of f and w in our framework creates a harder non-quadratic problem. However, the semi-supervised learning might give better generalization on the test samples.

Inductive learning

Although *HyperPrior* is designed for semi-supervised learning, it is convenient to use it for inductive learning as well [37]. Given an optimal weighting w^* and optimal labeling f^* learned by *HyperPrior*, for a new test sample $\hat{v}$, we minimize the objective function Eqn. (2.4) only with respect to this new label $f(\hat{v})$, that is

$$\underset{f(\hat{v})}{\text{minimize}} \ \frac{1}{2}\sum_{v\in V} S_{\hat{v},v}\left(\frac{f(\hat{v})}{\sqrt{d(\hat{v})}}-\frac{f^*(v)}{\sqrt{d(v)}}\right)^2 + \mu\,\|f(\hat{v})\|^2 + \text{constant}$$

where $S_{\hat{v},v} = \sum_{e\in E}\frac{h(\hat{v},e)h(v,e)w^*(e)}{d(e)}$. The analytical solution to this optimization problem can be calculated in $\Theta(n)$ as follows:

$$f(\hat{v}) = \frac{\sum_{v\in V} S_{\hat{v},v}\frac{f^*(v)}{\sqrt{d(v)}}}{\sum_{v\in V} S_{\hat{v},v}}\sqrt{d(\hat{v})}.$$

2.3.2 Modeling genomic data with prior knowledge

Modeling arrayCGH data with spatial prior

In arrayCGH data, each spot is assigned a log-ratio denoting how the corresponding DNA copy number varies compared to the normal level. The sign of the value represents either a “gain” (amplification) or a “loss” (deletion) of the DNA segment in the corresponding regions on the chromosomes. We performed k-means ($k = 3$) clustering on all log-ratios and then classified the values into three classes, “gain” state, “basal” state and “loss” state. The k-means ($k = 3$) clustering result is very stable on the one-dimensional data. To represent both the DNA amplification event and the deletion event at each spot, the log-ratio values are split into an amplification group and a deletion group (Figure 2.1(A)). In other words, we use two hyperedges to differentiate the amplification and deletion states of CNVs. For example, in Figure 2.1(A), sample S1 and S2 have a “gain” state in spot1 and sample S5, S6 and S7 have a “loss” state in spot1. Accordingly, sample S1 and S2 are covered by an amplification hyperedge e_1, and sample S5, S6 and S7 are covered by a deletion hyperedge e_2. The "basal" state is regarded as no event at the spot.

To both build a predictive model and identify discriminative regions, *HyperPrior* learns the weighting of all the hyperedges in the hypergraph built from the arrayCGH data. A spatial prior is introduced on the weights to be learned by the *HyperPrior* algorithm. Our first assumption is that DNA amplification and deletion events tend to occur in consecutive regions on the chromosomes. Thus, the weights of the hyperedges corresponding to adjacent spots should be similarly weighted. The second assumption is that only CNVs in a small portion of regions in the genome will contribute to a good classification. In Figure 2.1(B), the states of adjacent spots are connected in a prior graph as constraints on the weights in the learning regularization framework of *HyperPrior*. We set the connectivity indicator $\delta_{i,j} = 1$ in Eqn. (2.2)&(2.3) for the adjacent states. For example, the amplification state of spot 1 is connected to that of spot 2, and the one of spot 2 is connected to that of spot 3, etc.

Modeling gene expressions with protein-protein interactions as prior knowledge

Gene expression profiles are grouped into three states: a basal state or an up/down-regulated state (Figure 2.1(A)) based on the sign of the expression values. A hypergraph is built with (positive/negative/test) samples as vertices and gene expression states as hyperedges. Similarly, we assume that genes interacting with each other in the PPI network are functionally related, and thus, they should be similarly weighted. We set the connectivity indicator $\delta_{i,j} = \frac{1}{D_{i,j}}$ in Eqn. (2.2)&(2.3) between the gene pairs, where $D_{i,j}$ is the distance between two genes in the PPI network. The regularization framework seeks a solution for both sample classification and gene weighting by considering the connectivities in the hypergraph. The incorporation of the protein-protein interaction network provides prior knowledge on weighting interacting genes with similar values.

2.4 Experiments

We evaluated two versions of *HyperPrior*, *HyperPrior-LP* with the constraint given by Eqn. (2.2) and *HyperPrior-NB* with the constraint given by Eqn. (2.3) on artificial datasets, two arrayCGH datasets and two gene expression datasets. A large curated protein-protein interaction network constructed by [20] was used as prior knowledge for classifying the gene expression datasets. In all experiments, we compared the classification performance of *HyperPrior* with the hypergraph-based learning algorithm [25], SVMs with linear kernel and RBF kernel (Matlab Bioinformatics Toolbox (V3.0)). L_1-SVM and fused SVM [23] were included as additional baselines in the experiments on the arrayCGH datasets. The linear lasso model by [32] and the graph-Laplacian-transform method by [22] were included as additional baselines in the experiments on the gene expression datasets. The classification performance of all the methods were evaluated by leave-one-out accuracy or AUC of receiver operating characteristics (ROC): the normalized area under a curve plotting the number of true positives against the number of false positives by varying the threshold on the decision values [38].

2.4.1 Simulations

To mimic the noisy nature of the genomic data, we tested *HyperPrior-LP* on artificial hypergraphs with many noisy hyperedges. In all experiments, we labeled 50% of the vertices for training and held out the other 50% of the vertices for testing. We randomly generated hypergraphs with a large number of non-informative hyperedges connecting randomly selected vertices and a certain number of special hyperedges, each of which alone is not very informative but is highly informative in combination. We first generated a set of vertices with 80% of the vertices in one class and 20% of the vertices in the other class. The set was then split into 5 weak informative hyperedges with an equal number of vertices. The informative hyperedges were generated to simulate the concerted behavior of genomic features, which are often non-informative unless combined as a module. The prior knowledge was introduced as the pairwise constraints between the informative hyperedges. Some other random constraints between non-informative hyperedges were also introduced as noise.

The algorithms were tested on 100 hypergraphs generated as described above. The average AUC of the baselines and *HyperPrior-LP* with different percentages of informative hyperedges are reported in Figure 2.2. Because the results are similar for different choices of ρ and α ($\alpha = \frac{\mu}{1+\mu}$) parameters, we only plot the case with $(\alpha, \rho) = (0.5, 1)$. It

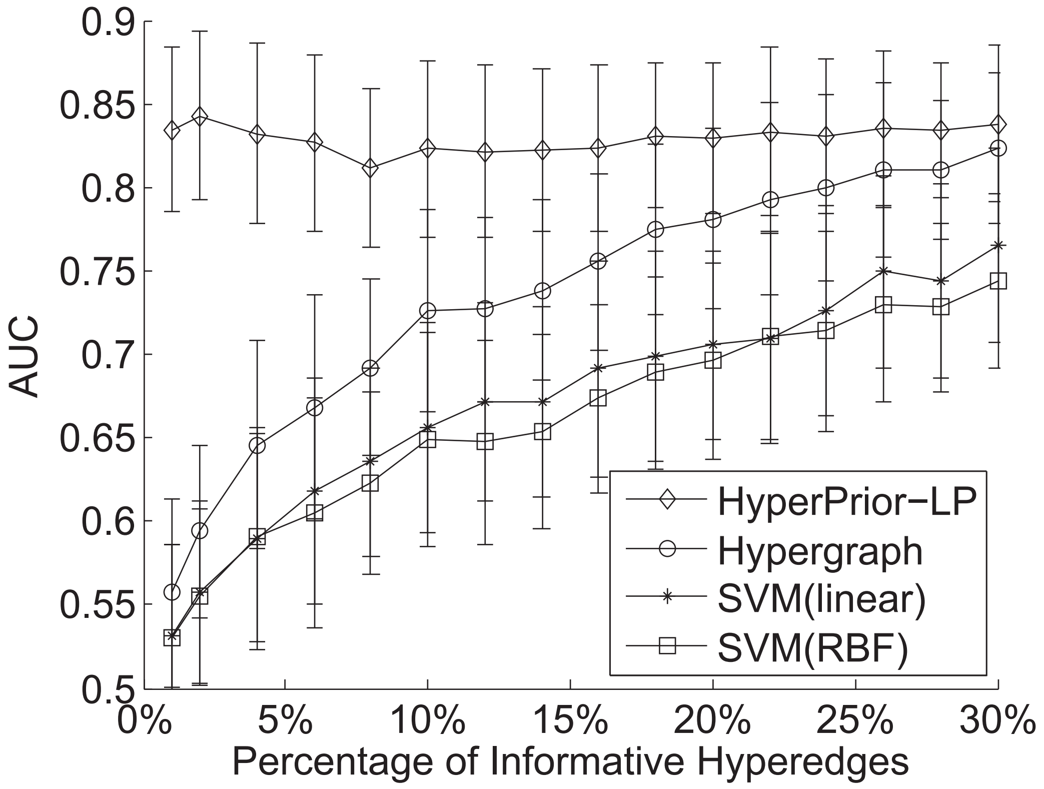
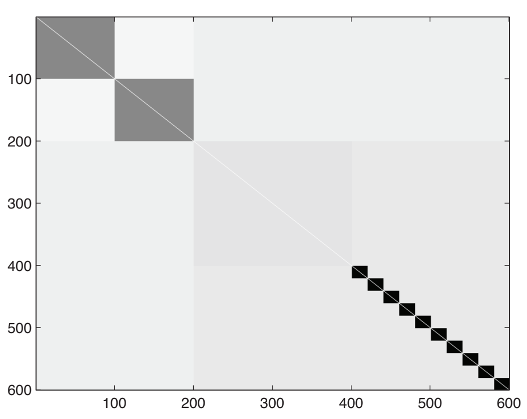
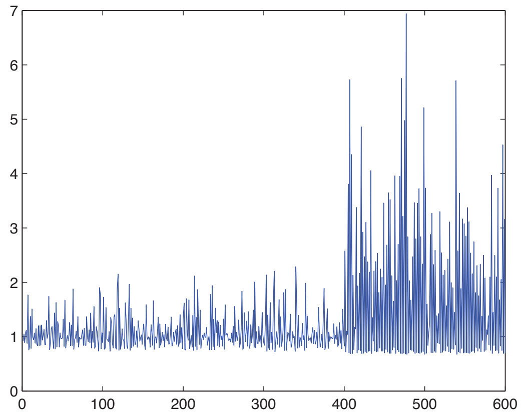


Figure 2.2: **Simulation results.** This plot compares the algorithms by averaged AUC over 100 trials. The x-axis is the percentage of informative hyperedges in the hypergraph. We set $(\alpha, \rho) = (0.5, 1)$ for *HyperPrior-LP*.

is clear in the plot that, when the prior knowledge gives useful information about interactions between informative hyperedges, the performance of our algorithm is significantly better than SVMs and the hypergraph-based algorithm with uniform weights. Since in this simulation, only very high-order combination of the hyperedges could provide good classification performance, SVMs performed poorly in all cases.



(A) Interactions among hyperedges



(B) Weights assigned by *HyperPrior*

Figure 2.3: **Simulation on informative hyperedges discovery.** (A) There are 10 small interacting modules among 200 informative edges and 2 larger interacting modules among non-informative modules. (B) The x-axis is the index of the hyperedges aligned with the indexes in plot (A). The y-axis is the weights. The 200 informative edges are assigned larger weights by *HyperPrior*.

Informative hyperedges selection

To test if the *HyperPrior* algorithm can select informative hyperedges, we design one additional experiment with more diverse prior knowledge on both informative and non-informative hyperedges. We generate 400 non-informative hyperedges and 200 informative hyperedges. The 200 informative hyperedges are grouped into 10 fully connected cliques in the interaction network. We also group 200 random hyperedges into 2 fully connected cliques. The adjacency matrix is shown in Figure 2.3(A). The 2 cliques of non-informative hyperedges are on the top-left of the matrix and the 10 cliques of informative hyperedges are on the bottom-right of the matrix. The weights learned by *HyperPrior* is plotted in Figure 2.3(B). It is evident that informative hyperedges are assigned much larger weights, which shows that the *HyperPrior* algorithm is capable of selecting true informative interaction components even under the presence of abundant irrelevant interactions. This result also suggests that the *HyperPrior* algorithm assigns weights to hyperedges based on both the predictability and modularity of the hyperedges, instead of the number of interactions that they have in the interaction network.

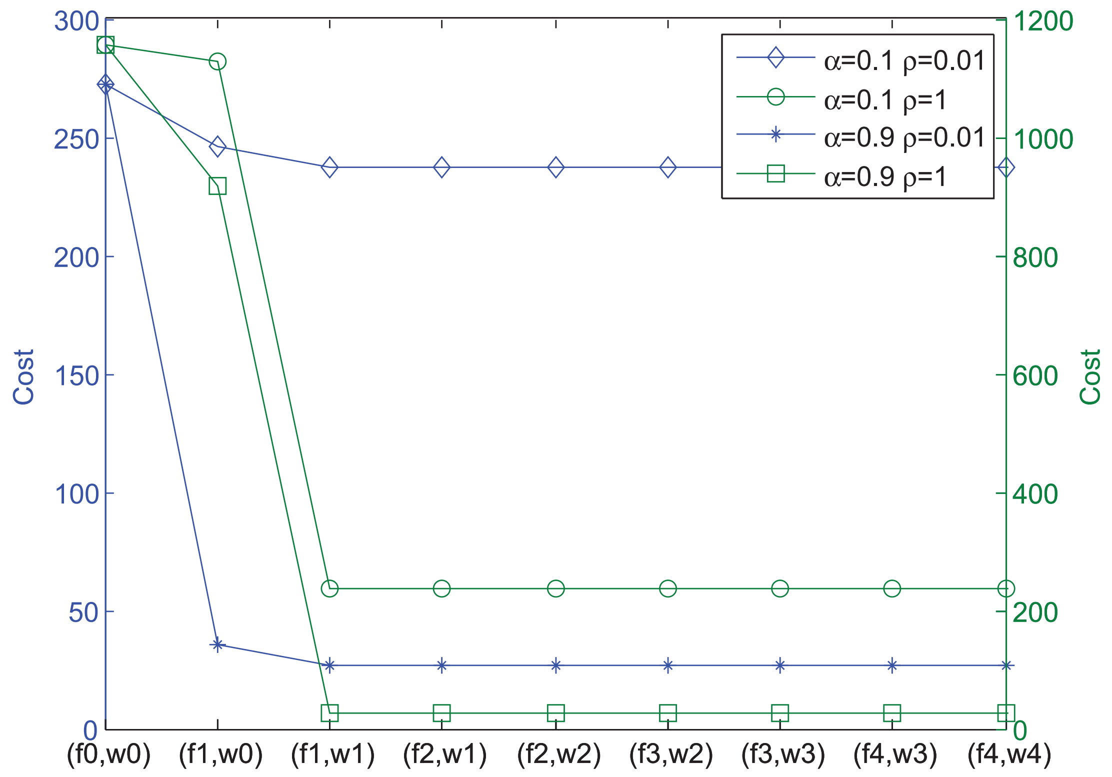


Figure 2.4: **Convergence of *HyperPrior*.** This plot shows the decrease of the cost function after each iteration of *HyperPrior*.

Accordingly, the *HyperPrior* algorithm achieves the highest AUC 0.874 in this experiment, while the hypergraph-based algorithm and SVM with linear kernel and RBF kernel only score 0.750, 0.596 and 0.604 respectively.

Convergency of the algorithm

To check the convergence of the *HyperPrior* algorithm, we measured the value of the cost function in each iteration on the real microarray gene expression datasets with selected 1,464 genes. The change of the cost function for different α and ρ parameters is shown in Figure 2.4. It is clear that the *HyperPrior* algorithm converges very fast. We also found that the value of f and w variables stay unchanged after the first 2 to 3 iterations.

In *HyperPrior*, we use $w = 1$ as the starting point. However, we also tried random

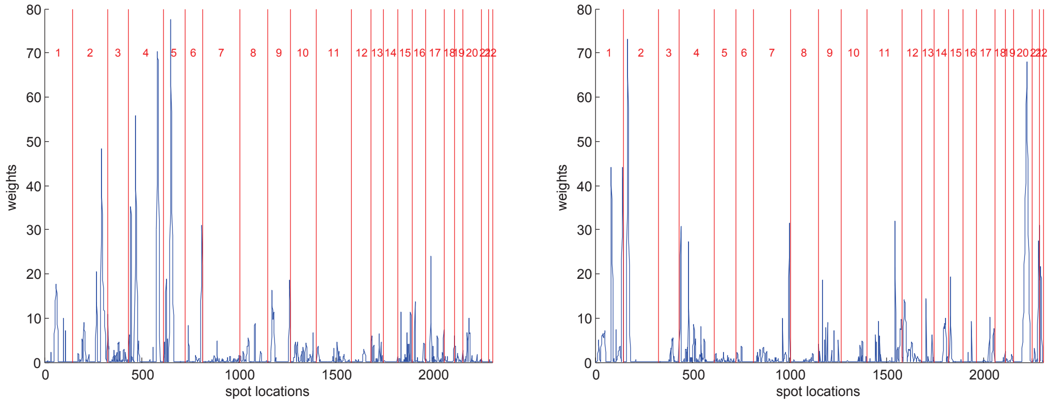
LOO errors	SVM (linear)	SVM (RBF)	L_1-SVM	fused SVM
Bladder tumors (by grade)	9	9	12	7
Bladder tumors (by stage)	9	9	13	7
Melanoma tumors	10	10	8	**7**
LOO errors	Hypergraph	*HyperPrior-LP*	*HyperPrior-NB*	
Bladder tumors (by grade)	11	**6**	**6**	
Bladder tumors (by stage)	9	**5**	6	
Melanoma tumors	**7**	**7**	**7**	

Table 2.1: **Classification performance on arrayCGH data.** This table shows the number of misclassified samples in the LOO cross-validation on the bladder cancer dataset with two different labeling schemes (by tumor grade or by cancer stage) and the melanoma cancer dataset. For the SVMs with linear and RBF kernels, combinations of parameters $C = \{10^{-5}, 10^{-4}, \ldots, 10^4, 10^5\}$ and $\sigma = \{10^{-5}, 10^{-4}, \ldots, 10^4, 10^5\}$ were tested. For the hypergraph-based algorithm and *HyperPrior*, parameters $\alpha = \{0.01, 0.1, 0.3, 0.5, 0.7, 0.9, 0.99\}$, and $\rho = \{10^{-3}, 10^{-2}, \ldots, 10^2, 10^3\}$ (for *HyperPrior* only) were tested. For L_1-SVM and fused SVM, combinations of parameters $\lambda = \{2^0, 2^1, \ldots, 2^9, 2^{10}\}$, and $\mu = \{2^{-10}, 2^{-9}, \ldots, 2^9, 2^{10}\}$ (for fused SVM only) were tested.

starting points and they all converged to the same solution as long as the initial constraints on w are satisfied. So empirically, the solution of *HyperPrior* is not affected by the starting point.

2.4.2 Classification of arrayCGH data

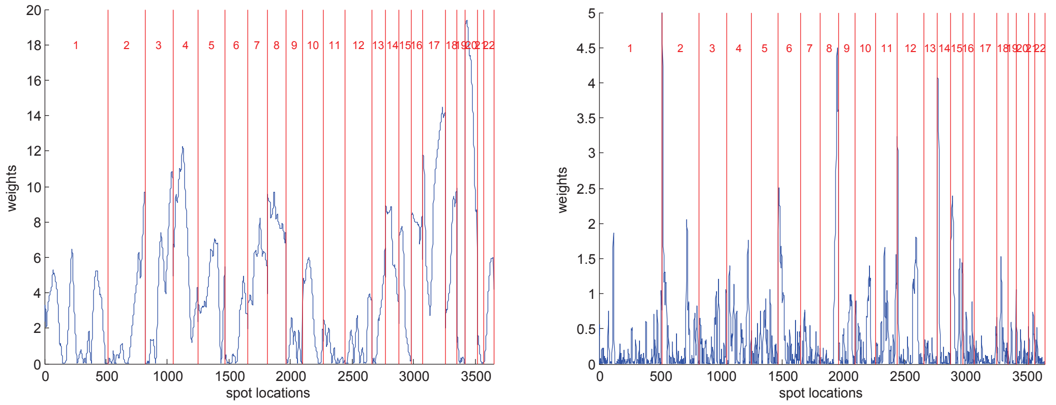
We tested *HyperPrior* on two arrayCGH datasets used by [23]. The first dataset contains arrayCGH profiles of 57 bladder tumor samples and the second one contains arrayCGH profiles of 78 melanoma tumor samples. Following the data preprocessing procedure in [23], we removed the probes in sexual chromosomes and tested three tumor classification problems: bladder tumors by grade (12 tumors of Grade 1 vs. 45 tumors of higher grades) and by stage (16 tumors of Stage T1 vs. 32 tumors of Stage T2+), and melanoma tumors by metastases (35 tumors that developed metastases within 24 months vs. 43 that did not). The L_1-SVM and fused SVM were implemented using the source code provided by [23]. A weighted incidence matrix H was used in the bladder cancer dataset because the results were more stable. We performed a cross-validation by a leave-one-out (LOO) procedure for the three classification problems. The number of misclassified samples by all the methods are reported in Table 2.1. On the bladder cancer dataset, *HyperPrior-LP* and *HyperPrior-NB* achieved the best classification accuracy compared to the other



weights of DNA amplifications weights of DNA deletions

(A) Bladder cancer

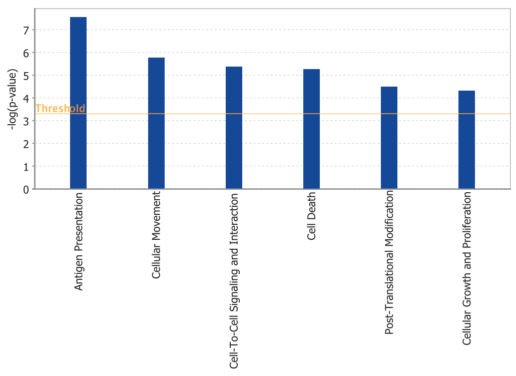
weights of DNA amplifications weights of DNA deletions



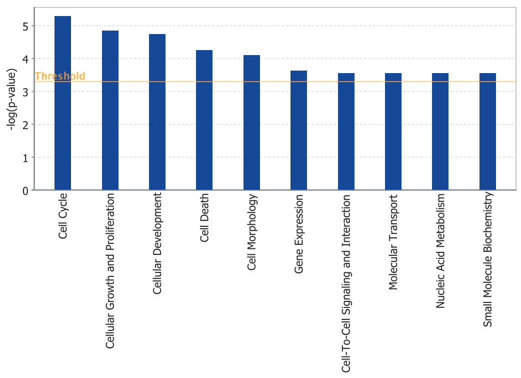
(B) Melanoma cancer

Figure 2.5: **Discriminative regions of DNA amplification and deletion.** The figures plot separately the weights of regions of “amplification state” and “deletion state”, assigned by *HyperPrior* with the α and ρ parameters giving the best results in cross-validation for the grade classification on bladder tumor samples and melanoma tumor samples. The spots are ordered by their locations on chromosomes and the corresponding weights are plotted in blue curves. Red lines represent the chromosome separations.

methods. On the melanoma dataset, the hypergraph algorithm, fused SVM, *HyperPrior-LP* and *HyperPrior-NB* all achieved the same error rate. Overall, *HyperPrior-LP* and



(A) Bladder cancer



(B) Melanoma cancer

Figure 2.6: Enriched biological functions in discriminative chromosomal regions.

HyperPrior-NB performed better than the baseline methods that do not utilize the spatial prior knowledge, and gave competitive performance against fused SVM, which also utilizes the same spatial prior knowledge.

The weights assigned by *HyperPrior-LP* with the optimal parameters are plotted separately for amplification events and deletion events along the chromosomes in Figure 2.5. In the bladder cancer dataset, the highly weighted regions of deletion show good agreement with the results reported by [39] in chromosome 2, 4, 7 and 11, but there is no significant overlap in highly weighted amplification regions. In the melanoma cancer dataset, many of the highly weighted regions of amplification events (chromosome 7, 8, 17 and 20), and deletion events (chromosome 4, 5, 8, 11, 14 and 15) show strong consistency with those identified by [40]. It is evident in the plots that only scarce chromosomal regions are highly weighted.

2.4.3 Functional analysis of discriminative chromosomal regions

We analyzed the genes located in the highly weighted chromosome regions with *Ingenuity* (`http://www.ingenuity.com/`) to check whether the genes involve over-represented GO categories and biological pathways relevant to bladder cancer and melanoma cancer. We selected the chromosome regions associated with the top 20 highly weighted amplification states and the top 20 deletion states on both datasets. On the bladder cancer dataset, 130 genes located in the amplification regions and 255 genes located in the deletion regions. On the melanoma cancer dataset, 205 genes and 28 genes located in the amplification regions and deletion regions, respectively. Using these genes as input, *Ingenuity* identified 6 and 10 enriched functions scoring a *p*-value less than 0.0005 on the two datasets, respectively (Figure 2.6). The enriched functions of bladder cancer include post-translation modification, antigen presentation and cellular movement, which are all consistent with those identified by [41, 42] and [43]. The enriched functions of melanoma cancer also include known gene functions related to cancer development such as cell cycle, cellular growth and proliferation, cellular development, and cell morphology [44, 40].

2.4.4 Classification of gene expressions

We then evaluated *HyperPrior* on two breast cancer gene expression datasets, the van 't Veer *et al.* dataset with 97 samples [9] and the van de Vijver *et al.* dataset with 295 samples [45]. A large curated human protein-protein interaction network was used as prior knowledge [20]. This network contains 57,235 interactions integrated from yeast two-hybrid experiments, predicted interactions from orthology and co-citatioin, and other literature reviews. The details of the quantization and normalization of the datasets are described in the original papers. The classification task is to classify patients who developed metastasis or were free of metastasis in five years after prognosis.

As suggested by [9], 231 genes were selected on a training set of 78 patients and the remaining 19 patients were held out as the test set in the van 't Veer *et al.* dataset. A leave-one-out cross-validation was then applied to the 78-patients training dataset to select parameters for classifying the 19-patients test dataset. The training error rate for each algorithm is reported in Table 2.2, 2.3, 2.4 and 2.5.

	σ/C	0.0001	0.001	0.01	0.1	1	10	100	1000	10000
SVM (linear)		0.218	**0.205**	0.244	0.244	0.231	0.244	0.244	0.244	0.244
SVM (RBF)	10	0.218	0.218	0.218	0.218	0.218	0.218	0.231	0.231	0.231
	100	0.218	0.218	0.218	0.218	0.218	**0.205**	0.244	0.244	0.231
	1000	0.218	0.218	0.218	0.218	0.218	0.218	0.218	**0.205**	0.244
	10000	0.218	0.218	0.218	0.218	0.218	0.218	0.218	0.218	0.218

Table 2.2: SVMs on 78 training samples from van 't Veer *et al.* dataset with 231 genes.

C/percentage	0.2	0.4	0.6	0.8	1
0.0001	0.231	0.218	0.231	0.231	0.231
0.001	0.244	0.231	0.269	0.231	0.218
0.01	0.218	0.269	0.282	0.269	0.256
0.1	0.231	0.269	0.218	0.218	0.244
1	0.231	0.256	0.218	**0.192**	0.231
10	0.244	0.269	0.218	**0.192**	0.231
100	0.256	0.269	0.218	**0.192**	0.231
1000	0.256	0.269	0.218	**0.192**	0.231
10000	0.256	0.269	0.218	**0.192**	0.231

Table 2.3: Rapaport *et al.*'s method on 78 training samples from van 't Veer *et al.* dataset with 231 genes.

λ_1/λ_2	0.0001	0.001	0.01	0.1	1	10	100	1000	10000
0.0001	0.256	0.256	0.269	0.269	0.269	0.269	0.269	0.269	0.269
0.001	0.256	0.256	0.256	0.269	0.269	0.269	0.269	0.269	0.269
0.01	0.269	0.256	0.282	0.269	0.269	0.269	0.269	0.244	0.295
0.1	**0.231**	**0.231**	0.244	0.282	0.308	0.321	0.295	0.282	0.269
1	0.346	0.346	0.346	0.372	0.359	0.333	0.308	0.256	**0.231**
10	0.295	0.295	0.295	0.295	0.295	0.282	0.282	0.282	0.282
100	0.564	0.564	0.564	0.564	0.564	0.564	0.564	0.564	0.564
1000	0.564	0.564	0.564	0.564	0.564	0.564	0.564	0.564	0.564
10000	0.564	0.564	0.564	0.564	0.564	0.564	0.564	0.564	0.564

Table 2.4: Li *et al.*'s method on 78 training samples from van 't Veer *et al.* dataset with 231 genes

	ρ/α	0.01	0.1	0.3	0.5	0.7	0.9	0.09
Hypergraph		**0.218**	0.231	0.231	0.231	0.244	0.269	0.231
HyperPrior (LP)	10	**0.244**	0.256	0.256	0.256	0.256	0.256	0.256
	1	**0.244**	0.256	0.256	0.256	0.256	0.256	0.256
	0.1	0.256	0.256	0.269	0.269	0.256	0.256	0.256
	0.01	0.308	0.321	0.295	0.295	0.269	0.256	0.321
	0.001	0.346	0.333	0.333	0.308	0.321	0.282	0.333
	0.0001	0.462	0.474	0.487	0.397	0.333	0.321	0.462
HyperPrior (NB)	10	**0.244**	**0.244**	**0.244**	**0.244**	0.256	0.269	**0.244**
	1	**0.244**	**0.244**	**0.244**	**0.244**	0.256	0.269	**0.244**
	0.1	**0.244**	**0.244**	**0.244**	**0.244**	0.256	0.269	**0.244**
	0.01	0.308	0.308	0.282	0.282	0.282	0.269	0.308
	0.001	0.346	0.333	0.333	0.308	0.321	0.269	0.333
	0.0001	0.462	0.462	0.487	0.385	0.333	0.321	0.462

Table 2.5: Hypergraph and *HyperPrior* on 78 training samples from van 't Veer *et al.* dataset with 231 genes

In the experiments on van de Vijver *et al.* dataset, we used for classification two subsets of hypothetical cancer susceptibility genes: 326 genes from *Ingenuity* and 1,464 genes from Cancer Genomics tool (`http://cbio.mskcc.org/CancerGenes`). We randomly run 5-fold cross-validation multiple times on van de Vijver *et al.* dataset and measure the average AUC. Note that within each experiment of a 5-fold cross-validation, another 4-fold cross-validation is used on the training set to determine the best parameters for *HyperPrior* and the baseline algorithms to test the held-out set. Table 2.6 and 2.7 list the cross-validation results of all algorithms on van de Vijver *et al.* dataset with 326 and 1,464 cancer genes. p-values from paired-sample t-test are also listed.

AUC	mean	std	vs. SVM (linear)	vs. SVM (RBF)	vs. Rapaport *et al.*	vs. Li and Li
SVM (linear)	0.676	0.061	1.000	0.403	0.297	0.001
SVM(RBF)	0.681	0.063	0.403	1.000	0.792	0.015
Rapaport *et al.*	0.682	0.072	0.297	0.792	1.000	0.041
Li and Li	0.695	0.071	0.001	0.015	0.041	1.000
Hypergraph	0.687	0.060	0.037	0.225	0.392	0.170
HyperPrior-LP	**0.697**	0.061	1.031E-04	0.003	0.011	0.737
HyperPrior-NB	**0.697**	0.060	1.086E-04	0.003	0.012	0.749
			vs. Hypergraph	vs. HyperPrior-LP	vs. HyperPrior-NB	
SVM (linear)			0.037	1.031E-04	1.086E-04	
SVM(RBF)			0.225	0.003	0.003	
Rapaport			0.392	0.011	0.012	
Li and Li			0.170	0.737	0.749	
Hypergraph			1.000	0.062	0.065	
Hypergraph-LP			0.062	1.000	0.985	
HyperPrior-NB			0.065	0.985	1.000	

Table 2.6: Accuracies of all algorithms on van de Vijver *et al.* dataset with 326 genes. Paired-sample t-test was performed to determine the significance of the improvement. p-values are listed in the table.

AUC	mean	std	vs. SVM (linear)	vs. SVM (RBF)	vs. Rapaport *et al.*	vs. Li and Li
SVM (linear)	0.671	0.066	1.000	0.425	0.296	0.018
SVM(RBF)	0.667	0.060	0.425	1.000	0.763	0.093
Rapaport *et al.*	0.665	0.067	0.296	0.763	1.000	0.189
Li and Li	0.657	0.068	0.018	0.093	0.189	1.000
Hypergraph	0.685	0.063	0.019	0.001	0.001	2.926E-06
HyperPrior-LP	**0.692**	0.062	2.960E-04	4.282E-06	3.497E-06	3.326E-09
HyperPrior-NB	**0.692**	0.062	3.232E-04	4.766E-06	3.876E-06	3.745E-09
			vs. Hypergraph	vs. HyperPrior-LP	vs. HyperPrior-NB	
SVM (linear)			0.019	2.960E-04	3.232E-04	
SVM(RBF)			0.001	4.282E-06	4.766E-06	
Rapaport *et al.*			0.001	3.497E-06	3.876E-06	
Li and Li			2.926E-06	3.326E-09	3.745E-09	
Hypergraph			1.000	0.187	0.196	
HyperPrior-LP			0.187	1.000	0.978	
HyperPrior-NB			0.196	0.978	1.000	

Table 2.7: Accuracies of all algorithms on van de Vijver *et al.* dataset with 1,464 genes. Paired-sample t-test was performed to determine the significance of the improvement. p-values are listed in the table.

Algorithms	van 't Veer *et al.*	van de Vijver *et al.*	
	231 genes	326 genes	1,464 genes
SVM (linear)	0.857	0.676	0.671
SVM (RBF)	0.857	0.681	0.667
Rapaport *et al.*	0.869	0.682	0.665
Li and Li	0.833	0.695	0.657
Hypergraph	0.857	0.687	0.685
HyperPrior-LP	**0.881**	**0.697**	**0.692**
HyperPrior-NB	0.869	**0.697**	**0.692**

Table 2.8: **Classification results on gene expression data.** On the van 't Veer *et al.* dataset, the AUC on the 19-patient test set is reported. On the van de Vijver *et al.* dataset, over the random 5-fold cross-validations (50 times on both the 326 genes and the 1,464 genes), the mean AUCs are reported.

The classification results for all algorihtms on both datasets are summarized in Table 2.8. It shows that both *HyperPrior-LP* and *HyperPrior-NB* performed better than SVMs and the method by [22] in all the experiments. In particular, *HyperPrior-LP* outperforms the five baseline algorithms in classifying the 19 test samples on the van't Veer *et al.* dataset by 3 to 6 percents. When the interaction network contains accurate and helpful information, larger ρs are chosen in cross-validation to take advantage of the prior knowledge (Table 2.5). Thus, we speculate that the protein-protein interaction network plays an important role in learning the better classifier, given the relatively large value for ρ in the experiments.

On both datasets, *HyperPrior* achieved around 2% percent improvement on the average AUCs. Although this improvement seems marginal, pairwise comparisons show that *HyperPrior* outperformed SVMs and the method by [22] significantly with p-value less than 0.05 by paired one-sample t-test (Table 2.6 and 2.7). The method by [32] performed similarly as *HyperPrior* (0.695 vs. 0.697) in the experiments with 326 genes on the van de Vijver *et al.* dataset, but this method did not perform well in the other two experiments. The hypergraph based algorithm achieved slightly worse results compared with *HyperPrior* in the experiments with 1,464 genes on the van de Vijer *et al.* dataset, but in the other experiments, the results were statistically worse.

Known Disease Gene	Gene Ranking		
	HyperPrior (LP) $\alpha = 0.5$, $\rho = 1$	*HyperPrior* (LP) $\alpha = 0.5$, $\rho = 0.1$	CC
TP53	1	1	1166
BRCA1	11	12	1285
KRAS2	15	19	1057
ESR1	17	16	122
HRAS	18	14	73
BARD1	56	62	350
ATM	60	59	1154
AKT1	70	79	737
TGFB1	107	112	628
CASP8	108	120	636
PTEN	129	137	708
SERPINE1	185	136	179
PPM1D	188	116	243
BRCA2	226	258	856
PIK3CA	450	421	127
STK11	588	588	1278

Table 2.9: The ranking of known breast cancer (OMIM#114480) susceptibility genes. We compared the ranking of the known cancer genes obtained by the *HyperPrior* algorithm with the ranking calculated by Correlation Coefficients (CC). We set $\alpha = 0.5$ and $\rho = 1$ and 0.1 to test *HyperPrior* algorithm.

2.4.5 Functional analysis of cancer genes

Cancer gene ranking of gene

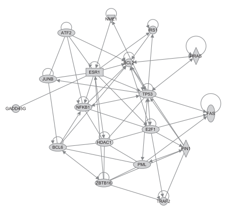
To demonstrate that *HyperPrior* is capable of identifying true cancer susceptibility genes, we compared the highly weighted genes by *HyperPrior* on the van de Vijver *et al.* dataset with known breast cancer causative genes reported in the overview section of breast cancer (MIM 114480) in Online Mendelian Inheritance in Man (May, 2007; `http://www.ncbi.nlm.nih.gov/omim/`). Specifically, we ranked the 1,464 cancer genes on van de Vijver *et al.* dataset and compare the ranking of known breast cancer genes with the ranking by correlation coefficients. While correlation coefficients gave very low rankings to the 16 known breast cancer causative genes in the dataset, *HyperPrior-LP* in two different settings ($\rho = 1$ and 0.1) assigned high ranks to most of the genes, with 14 out of 16 genes ranked in the top 300 genes (Table 2.9). Notable examples of the

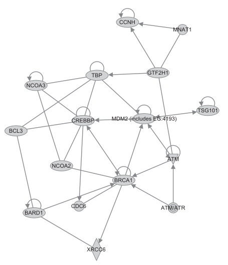
rank	$\alpha = 0.5$	$\alpha = 0.5$	$\alpha = 0.5, \rho = 1$	rank	$\alpha = 0.5$	$\alpha = 0.5$	$\alpha = 0.5, \rho = 1$
	$\rho = 1$	$\rho = 0.1$	with noise		$\rho = 1$	$\rho = 0.1$	with noise
1	TP53	TP53	TP53	26	HDAC1	SOS1	BUB1B
2	RB1	RB1	EGFR	27	CAV1	ATF2	CREBL2
3	MADH3	MADH3	RB1	28	TNFRSF6	SNW1	IGBP1
4	MAPK3	MAPK3	MADH3	29	CCNH	CAV1	
5	EGFR	EGFR	CREBBP	30	CDC25A	NFKB1	CCNA2
6	CREBBP	JUN	JUN	31	ONECUT1	HDAC1	GNAS1
7	JUN	CREBBP	CTNNB1	32	INSR	CCNH	PKMYT1
8	RAF1	CTNNB1	MAP2K4	33	NR3C1	NR3C1	MCM7
9	MADH2	RAF1	SLC2A5	34	CSNK2A1	CSNK2A1	PGR
10	CTNNB1	STAT1	SIL	35	MNAT1	ZNF145	MMP11
11	BRCA1	RASA1	SH3BGRL	36	JAK2	TNFRSF6	RPS13
12	STAT1	BRCA1	SLC16A1	37	SNW1	ONECUT1	BPAG1
13	MDM2	MADH2	SELP	38	CDC2	PML	P4HB
14	MDM2	HRAS	BRCA1	39	NFKB1	INSR	E2F1
15	KRAS2	YWHAZ	SDHD	40	PML	CDC25A	NESP55
16	MAPK1	ESR1	SFRS3	41	CDK4	IL6ST	PSMD7
17	ESR1	PTK2	SDHB	42	GTF2H1	CDC2	RPL4
18	HRAS	MDM2	SDHC	43	CSK	E2F1	RPL11
19	PTK2	KRAS2	LIF	44	HIF1A	CSK	PLOD3
20	SOS1	MDM2	EPHB2	45	IL6ST	TRAF2	SKP2
21	YWHAZ	MAPK1	SFRP1	46	SNRPD2	MNAT1	KPNA2
22	NRAS	JAK2	SET	47	CASP3	CDK4	FGG
23	ATF2	EEF1A1	MYBL2	48	FOXO1A	GTF2H1	FANCA
24	EGF	EGF	EEF1A1	49	STAT5A	CRK	DKFZP564A063
25	RASA1	NRAS	BIRC5	50	G22P1	HIF1A	G6PD

Table 2.10: The top 50 genes ranked by *HyperPrior*

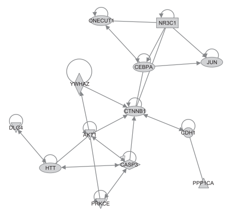
biomarker genes are tumor protein p53 (TP53), estrogen receptor 1 (ESR1), v-Ha-ras Harvey rat sarcoma viral oncogene homolog (HRAS), and v-Ki-ras2 Kirsten rat sarcoma viral oncogene homolog (KRAS), all of which were not identified in [9] but are highly ranked by *HyperPrior*.

We also introduced some noise to the PPI network to make the degree of each node no less than one half of the maximum degree in the network. The top 50 genes ranked by *HyperPrior* with two groups of parameters and with a PPI to which the noise is introduced are listed in Table 2.10.

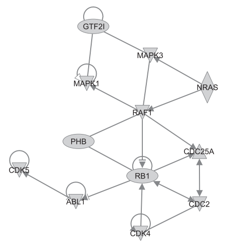

(A) TP53-subnetwork



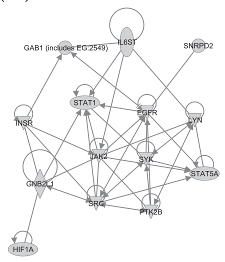
(B) BRCA1-subnetwork



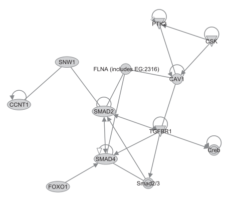
(C) AKT1-subnetwork



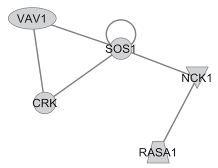
(D) RB1-subnetwork



(E) STAT1-subnetwork



(F) SMAD2-subnetwork



(G) SOS1-subnetwork

Figure 2.7: Seven interaction networks of the top 100 marker genes on van de Vijver *et al.* dataset. Known breast cancer causative genes such as TP53, ESR1 and BRCA1 play a central role in the networks. Other known susceptibility genes such as v-akt murine thymoma viral oncogene homolog 1 (AKT1), retinoblastoma 1 (RB1), signal transducer and activator of transcription 1, 91kDa (STAT1), SMAD family member 2 (SMAD2), and son of sevenless homolog 1 (SOS1) also tend to be hubs and interact with many other susceptibility genes in the networks. Note that we remove those marker genes that do not directly interact with other known susceptibility genes.

Cancer subnetworks

HyperPrior also identified seven cancer pathway networks enriched by the top 100 highly weighted genes. Two examples are TP53-subnetwork and BRCA1-subnetwork: TP53-subnetwork is involved with glucocorticoid receptor signaling, p53 signaling and B cell receptor signaling pathways, and BRCA1-subnetwork is over-represented with glucocorticoid receptor signaling, estrogen receptor signaling, and RAR activation. Other networks are also involved with glucocorticoid receptor signaling, RAR activation, estrogen receptor signaling and other canonical pathways. All these over-represented biological pathways are closely relevant to breast cancer (`http://cgap.nci.nih.gov/`). This observation again supports the hypothesis that cancer genes share specific pathways involved with disease and they often interact with each other in a protein-protein

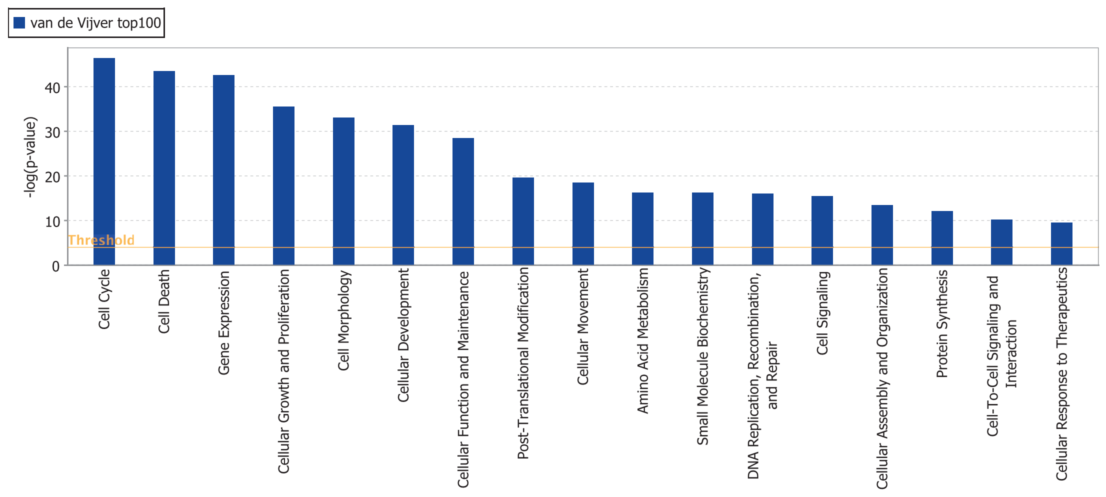


Figure 2.8: Enriched biological functions by the top 100 marker genes on the van de Vijver *et al.* dataset. The enriched functions are sorted by p-values calculated using the right-tailed Fisher Exact Test. All the enriched functions have p-value less than $1.0e-9$.

interaction network [21, 10, 20, 46]. We also analyzed the enriched biological functions of the biomarker genes by Gene Ontology (GO) annotations and pathway analysis with *Ingenuity*. The results are reported in Figure 2.7.

Enriched functions

We also analyzed the biological functions of the biomarker genes from van de Vijver *et al.* dataset by Gene Ontology (GO) annotations and pathway analysis with *Ingenuity* (version 5.5). We investigated whether the identified marker genes involve significantly over-represented GO categories and biological pathways that are related with breast cancer. With the top 100 marker genes as input, *Ingenuity* identifies 17 enriched functions scoring a p-value less than $1.0e-9$ on van de Vijver *et al.* dataset. Figure 2.8 shows the enriched biological functions from van de Vijver *et al.* dataset. All the 17 enriched functions of top 100 marker genes shows strong consistency with those identified by [44] and [10], indicating that these processes are significantly involved with the progression of cancer. Especially, the most significant functions such as cell cycle (p-value $= 4.03e-47$), cell death (p-value $= 3.44e-44$) , gene expression (p-value $=$

$2.43e-43$), and cellular growth and proliferation (p-value $= 2.7e-36$) are well known to be functionally involved with metastasis and development of breast cancer [47, 10, 20, 9]. Note that among the 17 functions, 11 functions are closely or exactly matched with the 21 functions discovered previously in [10].

2.5 Discussion

We introduce a hypergraph-based semi-supervised learning framework for sample classification and biomarker selection in arrayCGH and gene expression data with prior knowledge. We evaluated the algorithms with rigorous cross-validation and thorough parameterizations to show that the algorithms achieved promising classification performance and identified known cancer-relevant genetic elements from the genomic datasets. Despite the seemingly small improvement in the classification results, the improvement is mostly statistically significant and consistent on the datasets. As the availability and quality of biological knowledge continues to improve, more significantly better results are expected in the future.

The two variations *HyperPrior-LP* and *HyperPrior-NB* produced similar results. This observation suggests that both cost functions are viable choices to incorporate prior knowledge. We also used a weighted distance between the spots on chromosomes as correlation in the spatial prior graph for the arrayCGH datasets. But our preliminary results showed no improvement with the introduction of the more complex modeling. In an additional experiment, we also tested whether the high-ranking of the known breast cancer genes by *HyperPrior* was a biased output caused by the high connectivity of the known cancer genes in the PPI. We introduced random edges into the PPI network to make each gene have a degree that is at least half of the maximum degree in the network. We obtained similar top genes with the randomized network (Table 2.10). This result indicates that *HyperPrior* indeed can utilize clusters in the PPI as useful prior knowledge for biomarker selection.

Chapter 3

Integrating ArrayCGH Data with Probe Alignment

In this chapter, we propose a probe alignment kernel and a multiple probe alignment algorithm to integrate arrayCGH datasets from different platforms for sample classification and common CNV identification.

3.1 Introduction

Since DNA copy-number variations (CNVs) play an important role in tumorigenesis, one of the main tasks in copy number analysis is to identify the events of amplification or deletion on chromosomes. Many copy number datasets were generated by arrayCGH technology and these datasets were used to discriminate healthy patients from cancer patients and classify patients of different cancer subtypes. Thus, arrayCGH data is considered as a new source of biomarkers that provide important information of candidate cancer loci for the classification of patients and discovery of molecular mechanisms of cancers [7].

ArrayCGH allows rapid mapping of DNA copy-number variations of a tumor sample by a locus-by-locus measure. In an arrayCGH experiment, probes (short DNA segments on chromosomes) of different lengths and locations are chosen for comparative genomic hybridization. The CNV information at a probe location is reported by the log-ratio of the probe intensity between a case and a reference. The sizes and the number of the

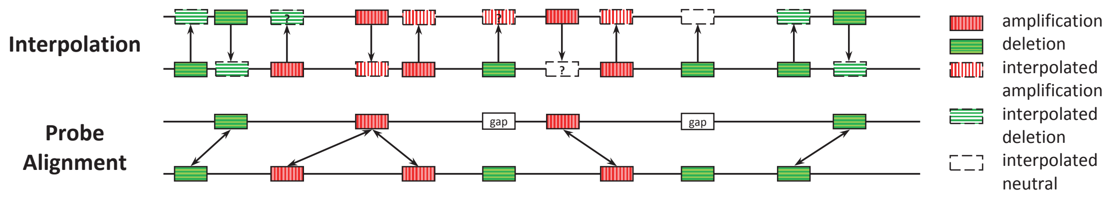


Figure 3.1: **A hypothetical example of interpolation and probe alignment.** Interpolation and probe alignment are applied to compare the same two chromosomes. The probe locations are marked with vertically striped blocks (amplifications) or horizontally striped blocks (deletions). The interpolation interpolates each probe to its corresponding position on the chromosome and add a new probe in the other series. The CNV value of the new probe is guessed by checking the neighboring probes in the same series. The probe alignment aligns nearby probe pairs in the two series with similar CNVs. The matched probe pairs are connected by arrows in the alignment. The interpolated probes marked with “?” are possibly wrongly labeled in the interpolation.

probe determines the resolution of an arrayCGH experiment. The length of the DNA probes used by current platforms ranges from 100 bp to 5 kb, and in a typical study, the number of probes ranges from several hundreds to tens of thousands. For example, a Human array 2.0 chromium surface array can consist of only 2,464 probes at 1.5 Mb resolution, while a high-resolution tiling array provides measurements of 36,288 probes.

The large difference in the sizes and numbers of probes makes integration of multiple arrayCGH datasets used for similar studies a challenging problem. From a data analysis perspective, the arrayCGH data are series of real values labeled by their chromosomal positions. The traditional approach is an interpolation between the two series of probe locations from two platforms. As described in Figure 3.1, in an interpolation probe locations on one platform are added (interpolated) to the target platform and the corresponding CNV intensities are guessed by intensities of the most nearby probes in the target platform. However, when the two platforms have large variations in the number of probes, the interpolation is not an appropriate way to integrate the two series of probes. Specifically, when the two series are both sparse, the interpolated points might give misleading or wrong information. For example, in Figure 3.1, some of the probes are interpolated to locations close to probes with opposite CNV. When the probes are sparsely located, this might introduce false positives. Moreover, some of the interpolated probes are far from other probes. In these cases (i.e. the interpolated probes marked by “?” in the figure), it is ambiguous to decide the CNV for the interpolated new probes.

In this thesis, we propose to integrate arrayCGH datasets with probe alignment. As shown in Figure 3.1, two series of probes from two arrayCGH samples are aligned based on both their chromosomal locations and their CNV log-ratios. The probes in one series are matched to the probes in the other series. The alignment representing an approximate positional matching of the two compared chromosomes with the maximum possible overlap of CNV events. The probe alignment seeking the maximum possible CNV overlap is motivated by the fact that two tumor samples are likely to share some common aberration regions related to tumor development and progression. If a probe in one series shares the same CNV with another nearby probe in the other series, it is likely that the two probes capture the same common tumor aberration in the two samples. Thus, the two probes should be aligned to enhance the signal. Based on this assumption, a probe alignment is more capable of detecting weak common CNV signals between two chromosomes if only sparse information is available. The problem of finding the best alignment of two probe series can be solved by a variation of the standard global sequence alignment algorithm [48]. Compared with interpolation, the main advantage of probe alignment is that the copy number of the chromosomal regions between probes are inferred based on the information of all the probes, instead of only the nearby ones, and the optimal global alignment provides the best guess of the CNVs in these regions.

Based on probe alignment, a probe alignment kernel derived from the probe alignment scores is introduced for classification of patient samples from multiple datasets. In the classification task, a binary classifier is trained to predict the tumor grade or cancer stages of a patient using the CNV information in different arrayCGH data. With the probe alignment kernel, many more patient samples from other datasets can be used to improve classification performance on one dataset. A multiple alignment algorithm is also designed to align all the probe series for identifying common CNV regions across patient samples from many datasets. In this case, many probe series from several arrayCGH datasets are aligned probe-by-probe, and the final alignment profile can reveal the detected common CNV regions along the chromosomes.

Sequence alignment algorithms are well-established methods for protein/nucleic acid sequence analysis [48]. Famous examples are Needleman-Wunsch algorithm, Smith-Waterman algorithm and fast approximations such as BLAST/PSI-BLAST [49]. The applications of these algorithms were relatively only limited to biological sequences rather than other types of data. It is worth mentioning that [50] proposed to use an alignment algorithm to align time series of gene expressions. Although motivated by another application, the central philosophy of finding a synchronization of two series to replace simple interpolation is closely related. There are several previous approaches for segmentation and CNV detection from array CGH data. For example, [51] proposed a Bayesian hidden Markov model to model relation between neighboring probes. Because these approaches assumes discrete copy number states without quantifying the distance between probes, they are not directly applicable in the multi-platform scenario.

3.2 Methods

In this section, we first describe how to align two probe series. We next introduce the probe alignment kernels for classification of CNV samples and a multiple alignment algorithm for identifying common disease CNV regions. Finally, the time complexity of probe alignment is analyzed.

3.2.1 Pairwise alignment of probe series

We denote the series of arrayCGH probes on a chromosome as a finite sequence of tuples $(x_1, l_1), (x_2, l_2), \ldots, (x_i, l_i), \ldots$, where each x_i denotes the log-ratio intensity of the ith probe and l_i denotes the location of the probe on the chromosome by kilo base pairs (kb). Given two such sequences $U = (u_1, a_1), (u_2, a_2), \ldots, (u_n, a_n)$ and $V = (v_1, b_1), (v_2, b_2), \ldots, (v_m, b_m)$ of length n and m respectively, we define three functions, $S(i, j)$, $L(i, j)$ and $R(i, j)$, for computing the alignment between the subsequences $(u_1, a_1), \ldots, (u_i, a_i)$ in U and the subsequences $(v_1, b_1), \ldots, (v_j, b_j)$ in V. $S(i, j)$ is the optimal alignment score when (u_i, a_i) is aligned with (v_j, b_j); $L(i, j)$ is the optimal alignment score when (u_i, a_i) is aligned with a gap, and $R(i, j)$ is the optimal score when (v_j, b_j) is aligned with a gap. Finally, a function $M(i, j)$ defined as the max of $S(i, j)$, $L(i, j)$ and $R(i, j)$ gives the optimal alignment score up to position i in U and position j in V. With initialization $S(0, *) = S(*, 0) = L(*, 0) = L(0, *) = R(*, 0) = R(0, *) = 0$, the functions can be recursively evaluated with a variation of standard dynamic programming as follows,

$$\begin{aligned} S(i,j) &= \max \begin{cases} M(i-1,j-1) + s(i,j) \\ S(i,j-1) + s(i,j) \\ S(i-1,j) + s(i,j) \end{cases} \\ L(i,j) &= M(i-1,j) + g \\ R(i,j) &= M(i,j-1) + g \\ M(i,j) &= \max\{S(i,j), L(i,j), R(i,j)\} \end{aligned}$$

where function $s(i,j)$ gives the substitution score between (u_i, a_i) and (v_j, b_j) and constant g is a gap penalty. The gaps in the alignment are introduced to capture the regions that are sufficiently covered by probes in one platform but not the other. Note in our formulation, a probe in one probe series can be matched with multiple probes in the other series, i.e. we also consider $S(i,j-1) + s(i,j)$ and $S(i-1,j) + s(i,j)$ when $S(i,j)$ is calculated. Because different platforms have different probe densities, it is more reasonable to allow one-to-many matching in the alignment.

Given two probe series U and V, the substitution score $s(i,j)$ between (u_i, a_i) and (v_j, b_j) needs to be designed to quantify a composition of two measurements: First, how close the two positions a_i and b_j are, and second, how similar the two CNV log-ratios u_i and v_j are. The substitution scores should encourage alignment of probe pairs that have similar amplification/deletion log-ratios at nearby locations. A simplest scoring function can be defined as follows,

$$s(i,j) = e^{\frac{-|a_i - b_j|}{\sigma}} * u_i * v_j \tag{3.1}$$

where $e^{\frac{-|a_i - b_j|}{\sigma}}$ quantifies how close the two positions are and $u_i * v_j$ is a simple measure of whether the two CNV log-ratios are similar or opposite. In this scoring function, the distance on a chromosome is normalized by a constant σ, which can be estimated from the actual probe locations in the probe series data. The closer the two probes, the larger the score. The similarity between two CNV log-ratios is simply taken as the product of the two values. The main advantage of this scoring function is that it is straightforward and parameter free. Note that, because the probes might represent the DNA copy numbers in chromosome regions of various lengths, the optimal probe alignment does not necessarily preserve the best sequential mappings between the two series by locations. Thus, a probe might not be matched with its closest peers in the other series in the alignment.

To produce more refined multiple probe alignments, the scoring function in equation (3.1) can be extended to be a positive function as follows,

$$s(i,j) = e^{\frac{-|a_i-b_j|}{\sigma}} * ([u_i * v_j]_+ + 1) \tag{3.2}$$

where $[u_i * v_j]_+$ is a refined product based on the sign and the value of u_i and v_j, defined as below,

$$[u_i * v_j]_+ = \begin{cases} u_i * v_j & \text{if } u_i * v_j \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

The product is then shifted by 1 to keep the positiveness. The refined similarity between u_i and v_j considers two scenarios. First, when u_i and v_j have the same sign, the similarity is still the product between them. In the second case, when u_i and v_j have different signs, the similarity between them is 0. In this scoring scheme, two probes with exactly the same position but different CNV signs will still be matched together to prevent needlessly penalizing the matches with gap insertions. This also implies that two probes at the same chromosomal position will still be matched even if they have opposite CNV events.

The choice of the gap penalty g depends on the substitution function $s(i,j)$. If the substitution function in Eqn. (3.1) is used, g can simply be set to a very small positive constant, which guarantees that a gap is preferred over matching two probes with different signs, and that only a very small value is added to the overall alignment score from the gap insertions. Intuitively, when g is small enough, an identical alignment and similar kernels will be resulted. If the substitution function in Eqn.(3.2) is employed, the gap penalty $g = e^{\frac{-\tau}{\sigma}}$ is used to match the scaling of the new scoring function. By definition, this gap penalty only allows two probes with different signs to match if their distance on the chromosome is less than τ. This means that two probes with different signs will be matched if they are close-by sufficiently on the chromosome. τ can be empirically chosen as a value smaller than σ.

3.2.2 Probe alignment kernel

To address the problem that there is no direct way to integrate arrayCGH/CGH profiles from different platforms in classification, we propose a kernel based on probe alignment, which measures the similarity between samples with their probe alignment scores. The probes located on the DNA of each chromosome of human genome are aligned and the alignment scores are summarized as the kernel value. The alignment kernel function computing the best global alignment score between U and V is given as follows,

$$K(U,V) = K(((u_1,a_1),\dots,(u_n,a_n)),((v_1,b_1),\dots,(v_m,b_m))) = \frac{M(m,n)}{m+n}$$

where the alignment score is normalized by $m+n$, the lengths of the two series to avoid the bias by the number of probes.

The kernel function $K(U,V)$ is used to compute the alignment score between two probe series of a particular human chromosome. Given two arrayCGH samples each with P chromosomes in genome (23 pairs of chromosomes in Human genome), the total alignment score between the two samples is the summation of the best global alignment scores of the P pairs of chromosomes. Let $\{U^{(1)},\dots,U^{(P)}\}$ and $\{V^{(1)},\dots,V^{(P)}\}$ denoting the two sets of probe sequences of the chromosomes. The probe-alignment kernel function $\mathcal{K}$ is defined as

$$\mathcal{K}(\{U^{(1)},\dots,U^{(P)}\},\{V^{(1)},\dots,V^{(P)}\}) = \sum_{i=1}^{P} K(U^{(i)},V^{(i)})$$

We choose the scoring function given in Eqn. (3.1) for the probe alignment kernel. When all the probe positions can be perfectly matched between two series, the probe alignment kernel tends to be very close to a simple linear kernel between the two probe series, where the kernel is simply the summation of the matched $u_i * v_j$ in the two series. Note that the probe alignment kernel is not positive semi-definite (PSD), similar to the alignment kernel for protein classification [52]. A positive constant is added to the diagonal of the kernel matrix to make it a valid kernel.

3.2.3 Multiple alignment of probe series

Similar to the multiple alignment algorithms for protein/DNA sequences, a multiple alignment procedure can also be used to align multiple probe series from different array-CGH datasets. The multiple probe alignment (MPA) can reveal common amplification

Algorithm 2 Multiple Probe Alignment (MPA)

```
Input: a set of probe series $X = \{X_1, X_2, \ldots, X_n\}$
Output: the merged probe series $X^*$
1: Compute the pairwise alignment score between all pairs in $\{X_1, X_2, \ldots, X_n\}$.
2: repeat
3:     Select the pair $X_a$ and $X_b$ with the largest alignment score.
4:     Merge $X_a$ and $X_b$ as a new series $X_{a,b}$ based on their alignment.
5:     Remove $\{X_a, X_b\}$ and add $X_{a,b}$: $X = X \cup \{X_{a,b}\} - \{X_a, X_b\}$.
6:     Compute the alignment score between $X_{a,b}$ and all other series in $X$.
7: until There is only one series left in $X$.
8: Return the final merged probe series $X^*$.
```

or deletion events even if the probe sets are at different chromosomal positions. We adopted the progressive multiple alignment strategy [53] and used the pairwise probe alignment with scoring function defined in equation (3.2) for base alignment. The algorithm is described in Algorithm 2.

The algorithm continuously merges the two most similar probe series and finally, gives the alignment merged from all the series. Merging two probe series is the crucial step. To accomplish the best merging, we slightly extended the definition of a probe series by adding a weight to each probe in the series to make a new sequence of triples (x_i, l_i, w_i). Each weight w_i quantifies the importance of the probe by keeping the count of the number of probes that have been merged into this probe in previous alignments. Specifically, two such series $X_a = (u_1, a_1, w_1), \ldots, (u_n, a_n, w_n)$ and $X_b = (v_1, b_1, w'_1), \ldots, (v_m, b_m, w'_m)$ are merged as a new series $X_{a,b}$ based on their alignment. There are two types of probes in the alignment: the probes which are aligned with a gap and the probes which are matched with some probe(s). New probes are calculated for these two cases as follows:

1. If a triple (u_i, a_i, w_i) or (v_j, b_j, w'_j) is matched with a gap in the alignment between X_a and X_b, the triple is kept unchanged and directly added into the new series $X_{a,b}$.

2. Because some consecutive probes in one series can be matched with some consecutive probes in the other series, multiple triples from both series might need to be merged as one new probe. Suppose $\{(u_i, a_i, w_i), (u_{i+1}, a_{i+1}, w_{i+1}), \ldots, (u_{i+k_1},$

$a_{i+k_1}, w_{i+k_1})\}$ and $\{(v_j, b_j, w'_j), (v_{j+1}, b_{j+1}, w'_{j+1}), \dots, (v_{j+k_2}, b_{j+k_2}, w'_{j+k_2})\}$ are the two sets of multiple triples of length k_1 and k_2 respectively that need to be merged. We combine them as a new tuple (z, c, w'') by

$$z = \sum_{p=i}^{i+k_1} u_p + \sum_{q=j}^{j+k_2} v_q$$

$$w'' = \sum_{p=i}^{i+k_1} w_p + \sum_{q=j}^{j+k_2} w'_q$$

$$c = \frac{\sum_{p=i}^{i+k_1} a_p * w_p + \sum_{q=j}^{j+k_2} b_q * w'_q}{w''}.$$

Note that the position of the new probe is calculated as a weighted average of the merged positions to relieve possible bias introduced by outlier probes.

3.2.4 Fast probe alignment in linear time complexity

The direct implementation of the recursive dynamic programming runs in quadratic time. Under an idea similar to fast banded sequence alignment, probe alignment can be computed in linear time complexity. The motivation is that those probe pairs that are too far from each other will never be matched in the optimal alignment; otherwise, there always exists a better path that replaces the bad matches with more insertions and deletions. Thus, in the linear time implementation, only those probe pairs that are in locations close enough need to be considered in the alignment. Specifically, we prove the following two propositions:

Given two probe series, $U = (u_1, a_1), (u_2, a_2), \dots, (u_n, a_n)$ *and* $V = (v_1, b_1), (v_2, b_2), \dots, (v_m, b_m)$*, and a small positive gap penalty* g*, let* $u_{max} = \max(|u_1|, \dots, |u_n|)$ *and* $v_{max} = \max(|v_1|, \dots, |v_m|)$*.*

Proposition 3.2.1. *The optimal alignment between* U *and* V *will only consist of gaps and pairs of aligned probes* (U_i, V_j) *with* $|a_i - b_j| < \theta$*, where* $\theta = \sigma * (\ln(u_{max} v_{max}) - \ln g)$*.*

Proposition 3.2.2. *The number of probe pairs that need to be considered in probe alignment is* $O(\frac{\theta m}{\delta})$*, where* $\delta = \min\{b_{j+1} - b_j\}$*.*

Proposition 3.2.1 states that in the optimal alignment, only those probe pairs with distance less than a threshold θ can be possibly aligned. Proposition 3.2.2 states that the number of such pairs is upper bounded by $\frac{\theta m}{\delta}$, where δ is the minimal distance between the adjacent probe locations in V. Thus, the dynamic programming only needs to explore a linear number of pairs defined as a function of θ and δ.

To prove that the time complexity of the dynamic programming algorithm for probe alignment scales linearly in the length the probe series, we will first prove two lemmas and then the theorem providing an upper bound of the time complexity.

Lemma 3.2.3. *Two probes U_i and V_j will not be matched in the dynamic programming if their matching score is less than the gap penalty, i.e. if $s(i,j) < g$, $M(i,j) = \max\{L(i,j), R(i,j)\}$.*

Proof. The recursive formulation is

$$S(i,j) = \begin{cases} M(i-1,j-1) + s(i,j) \\ S(i,j-1) + s(i,j) \\ S(i-1,j) + s(i,j) \end{cases}$$

Because $s(i,j) < g$,

$$\begin{aligned} M(i-1,j-1) + s(i,j) <& M(i-1,j-1) + g < M(i-1,j-1) + 2g = R(i-1,j) + g \\ \leq& L(i,j) \\ S(i,j-1) + s(i,j) <& M(i,j-1) + g = R(i,j) \\ S(i-1,j) + s(i,j) <& M(i-1,j) + g = L(i,j) \end{aligned}$$

Thus, $S(i,j) < \max\{L(i,j), R(i,j)\}$ and $M(i,j) = \max\{S(i,j), L(i,j), R(i,j)\} = \max\{L(i,j), R(i,j)\}$. If $s(i,j) < g$, the optimal $M(i,j)$ must be achieved by inserting a gap from $M(i-1,j)$ or $M(i,j-1)$. □

Lemma 3.2.4. *Let $u_{max} = \max\{|u_i|\}$ and $v_{max} = \max\{|v_j|\}$, and the scoring function $s(i,j) = e^{\frac{-|a_i-b_j|}{\sigma}} * u_i v_j$ is used. Then, if $|a_i - b_j| > \theta$, $s(i,j) < g$ for $\forall i,j$, where $\theta = \sigma * (\ln(u_{max}v_{max}) - \ln g)$.* [1]

[1] If the scorning function is defined by Eqn. (3.2), θ will be $\tau + \sigma * \ln(u_{max}v_{max} + 1)$. The proof is similar thus omitted.

Proof. This is true because

$$
\begin{aligned}
&|a_i - b_j| > \theta \\
\Rightarrow &|a_i - b_j| > \sigma * (\ln(u_{max}v_{max}) - \ln g) \\
\Rightarrow &|a_i - b_j| > \sigma * (\ln(u_i v_j) - \ln g) \\
\Rightarrow &|a_i - b_j| > -\sigma * \ln \frac{g}{u_i v_j} \\
\Rightarrow &-\frac{|a_i - b_j|}{\sigma} < \ln \frac{g}{u_i v_j} \\
\Rightarrow &e^{-\frac{|a_i - b_j|}{\sigma}} < \frac{g}{u_i v_j} \\
\Rightarrow &s(i,j) = e^{-\frac{|a_i - b_j|}{\sigma}} * u_i v_j < g
\end{aligned}
$$

This lemma shows that two probes with distance larger than a cutoff θ will not need to be checked in dynamic programming. □

By Lemma 3.2.3 and Lemma 3.2.4, we can prove the following theorem:

Theorem 3.2.5. *The dynamic programming algorithm for probe alignment scales linearly in the length of probe series.*

Proof. Let $k_i = \arg_j\{b_j \leq a_i < b_{j+1}\}$, $\delta = \min_j\{b_{j+1} - b_j\}$ and $d = \frac{\theta}{\delta}$. If $j < k_i - d$, $|a_i - b_j| = a_i - b_{k_i} + b_{k_i} - b_j \geq b_{k_i} - b_j \geq (k_i - j)\delta > d\delta = \theta$. If $j > k_i + 1 + d$, $|a_i - b_j| > \theta$ also holds for similar arguments. Then $\forall j \notin [k_i - d, k_i + 1 + d]$, by Lemma 3.2.3 and Lemma 3.2.4, $M(i,j) = \max\{L(i,j), R(i,j)\}$. Let $X = \{(i,j) | i \in \{1, \ldots, m\}, j \in [k_i - d, k_i + 1 + d]\}$. We will prove it is sufficient to check only the (i,j) pairs in X to achieve the optimal solution.

Suppose $\{(1,1), \ldots, (i_k, j_k), \ldots, (i_{k+1}, j_{k+1}), \ldots, (m,n)\}$ is an optimal solution for $M(m,n)$ and pairs between (i_k, j_k) and (i_{k+1}, j_{k+1}) are not in X. Without loss of generality, suppose they are in the lower-left region ($j < k_i - d$). We know the optimal path outside X can only be achieved by inserting gaps, so $M(i_{k+1}, j_{k+1}) = M(i_k, j_k) + (i_{k+1} - i_k + j_{k+1} - j_k)g$. In this case, we can safely find another path in X which yields the same score. For example, let the path from (i_k, j_k) to (i_{k+1}, j_{k+1}) be $\{(i_k, j_k), (i_k + 1, j_k), \ldots, (i_{k+1} - 1, j_k), (i_{k+1}, j_k), (i_{k+1}, j_k + 1), \ldots, (i_{k+1}, j_{k+1} - 1), (i_{k+1}, j_{k+1})\}$. Then it is a path in X which yields the same cost. (If some pairs in the middle are still

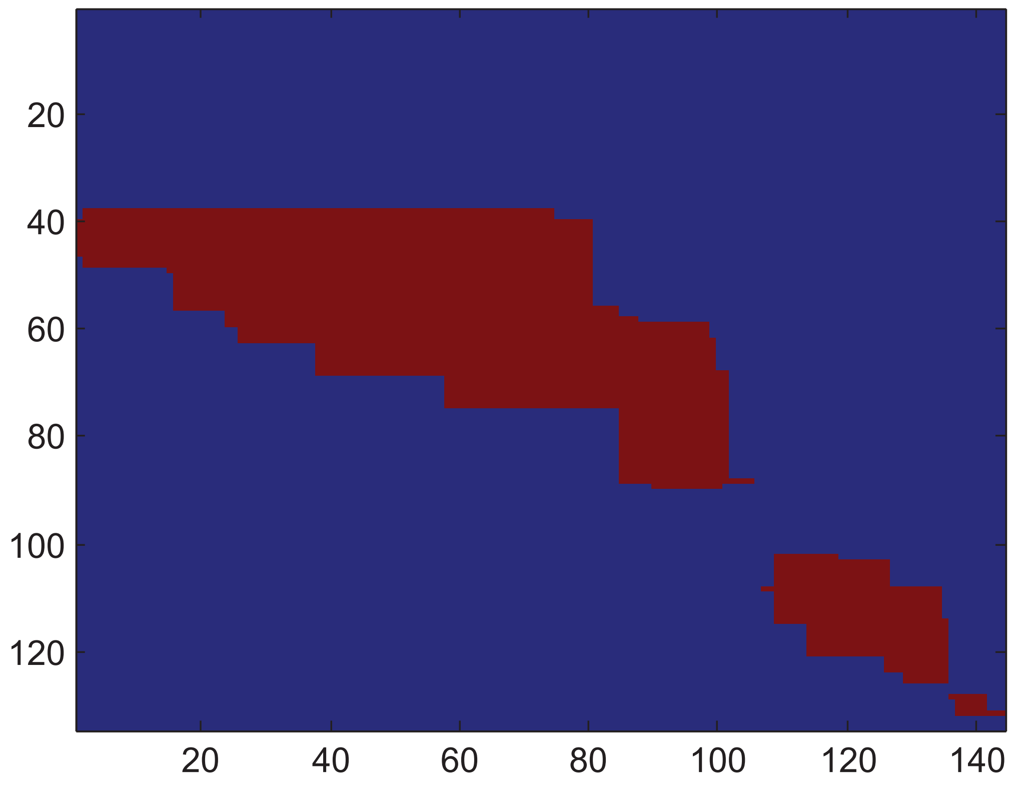


Figure 3.2: An example of the fast alignment implementation

outside X, we can use the same strategy to make it in X. Actually, it is possible that $M(i_{k+1}, j_{k+1}) = S(i_{k+1} - 1, j_{k+1}) + s(i_{k+1}, j_{k+1})$. However, we can still prove there is a path in X that achieves the same score. The proof is omitted here.) Thus we proved it is sufficient to use pairs only in X to calculate the optimal solution in our dynamic programming algorithm.

For any i, the number of j such that $(i, j) \in X$ is at most $2\lfloor\frac{\theta}{\delta}\rfloor + 1$. Thus, the time complexity of our dynamic programming algorithm is bounded by $O(\frac{\theta m}{\delta})$, which is independent to the size of the probe series. It only depends on the values and the densities of the probes. $\square$

Comments: In the proof, we used a universal $d = \frac{\theta}{\delta}$. However we can use different d_i for each i in the program implementation to further speed up the algorithm. The calculation of d_i still runs in linear time thus not change the upper bound of time complexity. However, to guarantee the solution is optimal, we have to make sure $\forall i, k_i - d_i \leq k_{i+1} - d_{i+1}$ and $k_i + d_i \leq k_{i+1} + d_{i+1}$. An example of this implementation is shown in Figure 3.2. X is the red region in the figure.

3.3 Experiments

We evaluated both the probe alignment kernel and the multiple probe alignment algorithm with experiments on three bladder cancer datasets as well as simulations. SVMs were used as the classifier in all the classification tasks. Both linear and RBF kernels were tested for the interpolation method. In all experiments, we tested SVM parameter $C = \{10^{-4}, 10^{-3}, \cdots, 10^{4}\}$ and $RBF\_\sigma = \{10^{-3}, 10^{-2}, \cdots, 10^{3}\}$.

3.3.1 Simulations

In the simulations, we first generated $10,000$ possible locations for probe sampling with a Markov model. We assume that locations are from two types of chromosome regions: gene-rich regions and non-coding regions. The probe density in gene-rich regions is higher than that in non-coding regions. Specifically, the distance between adjacent locations is 1 in gene-rich regions and 10 in non-coding regions. The Markov model takes two states {"gene-rich region", "non-coding region"} with a transition probability 0.9 for staying at a state and 0.1 for jumping between the states. Thus, continuous gene-rich or non-coding regions with variable lengths will be generated by the Markov model. We randomly generated 50 samples of probe series in the case group and another 50 in the control group by random sampling from the $10,000$ locations in each experiment. We simulated the DNA amplification and deletion events in the case samples for two test scenarios. In the first scenario, we randomly selected 20 regions (10 amplifications and 10 deletions), each of which consists of 10 consecutive locations out of the $10,000$ locations as discriminant CNVs on the chromosome. In the second scenario, we randomly selected 20 regions with variable numbers of locations between 1 and 20. This strategy generates CNV regions with significantly different lengths to mimic short chunks as well as large chunks of DNA amplification/deletions. The feature value of a probe that is not in a CNV region is generated from a normal distribution $N(0, 0.5^2)$. The rule implies that in normal individuals, a DNA amplification or deletion is a relatively rare event. A probe

Method	Fixed number of probes				
	$n = 100$	$n = 500$	$n = 1000$	$n = U(1 - 1000)$	$n = N(500, 100^2)$
Linear Kernel	0.502	0.759	0.922	0.701	0.764
RBF Kernel	0.519	0.719	0.863	0.659	0.716
Align Kernel	**0.597**	**0.875**	**0.950**	**0.811**	**0.857**
Method	Variable number of probes				
	$n = 100$	$n = 500$	$n = 1000$		
Linear Kernel	0.536	0.814	0.963		
RBF Kernel	0.535	0.753	0.905		
Align Kernel	**0.601**	**0.850**	**0.968**		

Table 3.1: **Classification accuracy on artificial datasets.** SVMs were used as the classifier with the kernels. The accuracies are averages of 10 random experiments for each case. n is the number of probes in each series. The probes are a random subset of the $10,000$ locations generated for each sample in three ways: 1) constant numbers (from 100 to $1,000$) of features were extracted out of the $10,000$ features for a sample such that each sample has the same number of features but at different locations; 2) Different numbers of features were extracted for each sample from a discrete uniform distribution on $[1, 1000]$; and 3) different numbers of features for each sample were generated from a normal distribution $N(500, 100^2)$ (rounded to positive integers).

in the CNV regions in a case sample will take a value from a normal distribution $N(1/-1, 0.5^2)$ to indicate a measured CNV value at the locations in the amplification/deletion regions.

Classification

We compared the standard interpolation with the alignment kernel using the cost function defined in equation (3.1) and a small gap $g = 0.001$. In the standard interpolation, a linear interpolation maps the missing positions in each series. The feature value at an interpolated position is assigned the distance-weighted average of the two nearest positions. After interpolation, each sample will have a feature vector of the same dimension, and standard classifiers such as SVM can be used for classification. We randomly generated datasets and tested the classification accuracy with a 5-fold cross-validation. In each fold, another 4-fold cross-validation on the training set is used to tune parameters for each algorithm. The average accuracy of each algorithm is reported in Table 3.1. The alignment kernel clearly outperformed the interpolation method. When probes are

$\frac{\lvert X\rvert}{mn}$	100	1000	5000	$U(1,1000)$	$N(1000,100^2)$
$\sigma = 100$	0.145	0.161	0.169	0.157	0.160
$\sigma = 10$	0.015	0.016	0.017	0.015	0.016

Table 3.2: **The time complexity of fast alignment algorithm in simulations.** In the table, m and n are the number of probes and σ is parameter in Eqn. 3.1. We tested $\sigma = \{10, 100\}$ and set $g = 0.001$, which is the same value of g as in the classification. $\frac{|X|}{mn}$ denotes the ratio of the pairs that need to be computed by the fast implementation to that by the quadratic time algorithm.

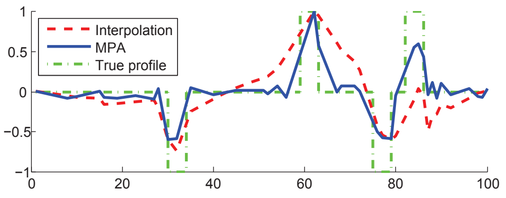
sparse, the improvement is up to 12%. When there are more probes available, the improvement is smaller. Especially, when $1,000$ probes are sampled, the improvement is only 3%. In the experiments with variable number of probes per sample or with variable number of locations per amplification/deletion region, the alignment kernel also significantly outperformed the interpolation. The experiments proved that the alignment kernel could handle sparse probe series with varying numbers of probes or varying lengths of amplifications/deletions well for classification.

Efficiency of the fast alignment algorithm

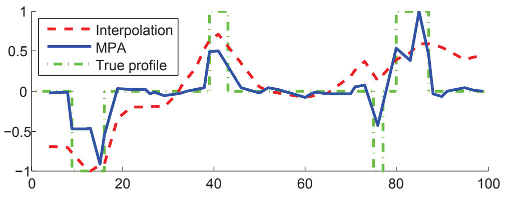
To test the efficiency of the fast alignment algorithm, we used the same method to generate several artificial datasets with fixed number of probes for each sample. Let X be the set of (i, j) pairs defined in the proof of Theorem 3.2.5 and m, n be the length of the two probe series. We measured the value of $\frac{|X|}{mn}$, which denotes the ratio of the number of pairs that needs to be computed by the fast implementation, to the number of pairs that is computed by the quadratic time algorithm. The smaller the value is, the more efficient the faster alignment algorithm is compared to the original quadratic time algorithm. The experiments were repeated 10 times and the averages were reported in Table 3.2. Clearly, the fast implementation significantly improved the running time.

Multiple alignment

To compare the multiple alignment algorithm with the interpolation method, we also tested alignment of multiple probe series for identifying the true common CNV regions. The multiple alignment algorithm used the scoring function defined by Eqn. (3.2) with $\tau = \frac{\sigma}{10}$. For better visualization, we only report the results on small datasets without



(A) Example of constant number of locations



(B) Example of variant number of locations

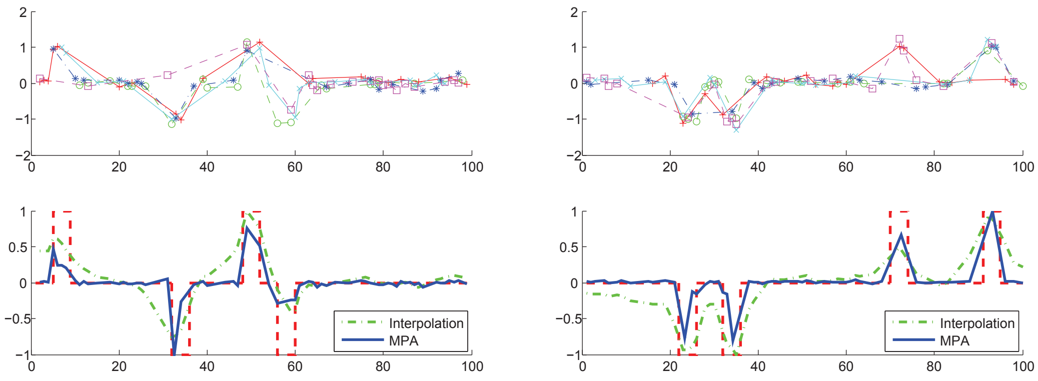
Figure 3.3: **Comparison between multiple probe alignment and interpolation in simulation.** CNV profiles generated by multiple alignment algorithm (MPA) and interpolations are compared with the four true common CNV regions plotted in the step function. All the values in the plots were rescaled to the range $[-1, 1]$ for better visualization.

distinguishing gene-rich regions and non-coding regions although similar results were observed on larger datasets. We generated 5 samples with 100 locations, within which there are 2 amplification regions and 2 deletion regions. In the example in Figure 3.3(A), each region contains 5 consecutive locations. In the example in Figure 3.3(B), each region contains 1 to 10 consecutive locations. We randomly sampled 10 features for both cases. From the two examples, the result of multiple alignment is clearly more accurate in detecting the exact locations of amplifications and deletions. Since the interpolation approach propagates the information from a probe to its interpolated neighbors, this propagation-based assumption often results in blurry boundaries and fails to distinguish close-by CNV events, as visualized in the examples in Figure 3.3. We also generated more multiple probe alignment examples in Figure 3.4. In these examples, we used the same way to generate artificial datasets but we randomly sampled 20 instead 10 features for each sample. Thus the profiles learned by multiple probe alignment and interpolation are more similar to the true profile. However, we can see the multiple probe alignment is still more accurate than interpolation.

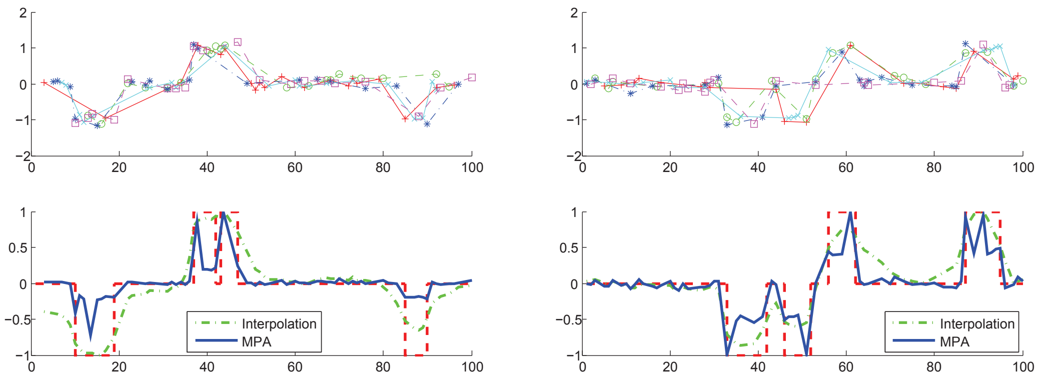
The results suggest that the multiple alignment algorithm is more robust to tolerate probe sparsity and noise in the data.

3.3.2 Bladder cancer datasets

We collected three different arrayCGH datasets generated for studying bladder cancer. The first dataset (D_1) introduced by [39] was generated with a HumanArray 2.0 array



(A) Examples of constant number of locations



(B) Examples of variant number of locations

Figure 3.4: **More examples for comparison between multiple probe alignment and interpolation in simulation.** In both (A) and (B), the top row: The original five aligned probe series plotted in 5 different lines; the bottom row: CNV profiles generated by multiple alignment algorithm (MPA) and interpolations. The four true common CNV regions are plotted in the red step function along with the profiles. All the values in the plots were rescaled to the range $[-1, 1]$ for better visualization.

consisting of 2,464 probes at 1.5Mb resolution. The second dataset (D_2) introduced by [54] was also generated with a HumanArray 2.0 array but consisting of 2,385 probes at 1.3MB resolution. The third dataset D_3 introduced by [55] was generated with a high-resolution tiling BAC Re-Array set 1.0 containing 36,288 BAC probes. The datasets were post-processed by the authors and the clones from sexual chromosomes were not included in the study because they are not comparable between male and

Distance	1	2	3	4	5	6	7	8	9	10	11
D_1	8.6e4	8.3e4	7.0e4	6.5e4	6.5e4	5.2e4	5.2e4	4.9e4	4.5e4	4.9e4	4.2e4
D_2	8.6e4	8.3e4	7.0e4	6.6e4	6.5e4	5.3e4	5.3e4	5.1e4	4.7e4	4.9e4	4.4e4
D_3	8.5e4	8.3e4	6.6e4	6.7e4	5.9e4	5.8e4	5.3e4	5.0e4	4.8e4	4.6e4	4.5e4
Distance	12	13	14	15	16	17	18	19	20	21	22
D_1	4.2e4	3.3e4	3.0e4	2.4e4	3.1e4	3.0e4	2.8e4	2.1e4	2.2e4	8.5e3	1.1e4
D_2	4.2e4	3.3e4	3.0e4	2.4e4	3.0e4	2.8e4	2.8e4	2.1e4	2.2e4	7.9e3	1.1e4
D_3	4.4e4	3.2e4	3.0e4	2.9e4	3.0e4	2.5e4	2.6e4	2.2e4	2.1e4	1.2e4	1.2e4

Table 3.3: Average probe distance (kb) on each dataset and chromosome.

female samples. After the pruning, dataset D_1 contains 98 samples and 2,142 probes, dataset D_2 contains 57 samples and 2,308 probes, and dataset D_3 contains 38 samples and 24,384 probes. We tested classification of patient samples by tumor grades in the three datasets. The two classes are distinguished by lower tumor grades or high tumor grades. Specifically, we considered 'Low' vs. 'High' in D_1, and '$\leq$G2' vs. '$>$G2' in D_2 and D_3 [23, 11].

Parameter selection

The σ parameter is the normalization term of the probe distances in Eqn. (3.1) and (3.2), and the gap penalty g defines the scoring scale of a gap in the alignment.

We plotted the probe locations against their indices on all 22 chromosomes of the three datasets used in the thesis. Most of the curves are close to a linear function, implying that the probe density ranges approximately in the same scale in each chromosome of the three datasets. We also calculated the average distance for all pairs of probes on each chromosome in the three datasets in Table 3.3. The average probe distance is similar in each dataset and chromosome in the range $(10^4, 10^5)$. Thus, any σ in the range can be used for a good normalization. Based on the statistics on each chromosome in Table 3.3, we set σ to be the average distance between all pairs of probes on a chromosome.

Note that different σs were used for probes on different chromosomes. In the three datasets, the σs for the 22 chromosomes are all in the order of $10^4 \sim 10^5$. Given the estimated σ from the data, an additional cross-validation on the training set is performed to choose the gap penalty g in $\{10^{-5}, \cdots, 10^{-1}\}$. In multiple alignment, σ can be chosen in the same way, but the g parameter is actually decided by the minimal

Method	D_1	D_2	D_3
Linear Kernel	0.808	0.612	0.835
RBF Kernel	0.824	0.603	0.806
Raw Kernel	0.812	0.642	0.846
Alignment Kernel	**0.870**	**0.797**	**0.883**

Table 3.4: **Improvement of bladder cancer classification from data integration with probe alignment kernel.** The accuracies are averages of 50 random 5-fold cross-validations.

allowed matching distance τ (Eqn. (3.2)). Since g will only take values in the range $[e^{-1}, 1]$, if $\tau < \sigma$, empirically any $\tau < \sigma$ can give reasonable multiple alignment on both artificial and real datasets. From the definition $g = e^{-\frac{\tau}{\sigma}}$, we can expect the results of multiple alignment will not be sensitive to τ as long as $\frac{\sigma}{10} < \tau < \sigma$. Thus, in the multiple alignments, we set $\tau = \frac{\sigma}{10}$.

Classification of patient samples

To evaluate the classification performance of the alignment kernel, we performed a special *cross-dataset validation*. We first selected a target dataset to run 5-fold cross-validation. In the experiment on each fold, we also used the samples in the other two datasets as additional training samples. We performed the cross-dataset validation for D_1, D_2 and D_3, each with 50 times of random 5-fold cross-validation. One additional baseline is the Raw kernel function defined by [56], which defines pairwise relations between CNV states as a new variation of linear kernel. Note that the experiments of all the compared methods were on the same 5 folds of each dataset. In the experiment on each fold, another 4-fold cross-validation on the training set was performed to tune parameters for each algorithm, specifically, the SVM C parameter and the gap parameter g for probe alignment, the SVM C parameter for linear kernel and raw kernel, and the SVM C parameter and the RBF_σ for RBF kernel.

Two categories of comparisons were performed. First, we tested SVMs used with linear kernel, RBF kernel and Raw kernel on the three datasets independently but with the alignment kernel using additional training data from the other two datasets. These experiments were purposed to show that integrating samples from other datasets in the training set can generate more accurate classifiers. The results are reported in Table

Additional training	Target dataset	Interpolation			Alignment kernel
		Linear	RBF	Raw	
D_2	D_1	0.805	0.817	0.816	**0.829**
D_3	D_1	0.812	0.833	0.828	**0.868**
D_1	D_2	0.635	0.707	0.659	**0.791**
D_3	D_2	0.666	0.755	0.674	**0.766**
D_1	D_3	0.819	0.785	**0.893**	0.847
D_2	D_3	0.839	0.812	**0.852**	0.848
D_2, D_3	D_1	0.813	0.842	0.797	**0.870**
D_1, D_3	D_2	0.668	0.785	0.664	**0.797**
D_1, D_2	D_3	0.812	0.834	0.881	**0.883**

Table 3.5: **Comparison of the probe alignment kernel and interpolation in cross-dataset validation.** The accuracies are averages of 50 random 5-fold cross-validations. The second column shows the target dataset for 5-fold cross-validation. The first column shows the datasets used as additional training samples.

3.4. Clearly, using the additional data, the alignment kernel significantly improved the classification accuracy on the three datasets. Next, the alignment kernel was compared with the other kernels on the interpolated data with the interpolated features on multiple datasets in all possible combinations. The average accuracy of all the methods are listed in Table 3.5. The alignment kernel outperformed or tied with the best of linear kernel, RBF kernel and raw kernel on the interpolated data in almost all the cases except it is the second best when tested with D_3 as target and D_1 for additional training.

Intuitively, a σ close to the average probe distance and a small g can provide a good kernel for classification in the alignment framework. We verified the intuition by additional classification experiments on the three bladder cancer datasets by all combinations of parameters ($g \in \{10^{-10}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and $\sigma \in \{10^1, 10^2, 10^3, 10^4, 10^5\}$) and listed the results in Table 3.6.

Detecting common CNV regions

To evaluate how well the three datasets can be integrated to detect the common CNV regions in bladder cancer tumors, we applied the multiple alignment algorithm to align the samples with higher tumor grades in the three datasets, and compared the alignment result with the interpolation result. Since all the samples in the same dataset have identical probe locations, we first calculated the average log-ratio intensities of the three

g/σ	$D_1(D_2)$					$D_2(D_1)$					$D_3(D_1)$				
	10^3	10^4	10^5	10^6	10^7	10^3	10^4	10^5	10^6	10^7	10^3	10^4	10^5	10^6	10^7
10^{-10}	0.80	**0.82**	0.82	0.82	0.82	0.62	0.78	0.76	0.77	0.77	0.81	0.81	0.81	0.78	0.78
10^{-5}	0.80	**0.82**	0.82	0.82	0.82	0.62	0.78	0.76	0.77	0.77	0.81	0.81	0.81	0.78	0.78
10^{-4}	0.80	**0.82**	0.82	0.82	0.82	0.62	0.78	0.76	0.77	0.77	0.81	0.81	0.81	0.78	0.78
10^{-3}	0.81	0.82	0.82	0.82	0.82	0.62	0.78	0.76	0.78	0.77	0.80	0.81	0.81	0.78	0.78
10^{-2}	0.80	0.78	0.81	0.81	0.81	0.62	0.74	0.76	**0.78**	0.77	0.82	0.83	0.81	0.79	0.78
10^{-1}	0.78	0.73	0.73	0.78	0.79	0.61	0.71	0.77	0.75	0.75	0.73	0.77	**0.87**	0.78	0.76

g/σ	$D_1(D_3)$					$D_2(D_3)$					$D_3(D_2)$				
	10^3	10^4	10^5	10^6	10^7	10^3	10^4	10^5	10^6	10^7	10^3	10^4	10^5	10^6	10^7
10^{-10}	0.85	0.83	0.86	0.86	0.85	0.69	0.73	0.76	0.75	0.75	0.86	0.83	**0.87**	0.85	0.85
10^{-5}	0.85	0.83	0.86	0.86	0.85	0.69	0.73	0.76	0.75	0.75	0.86	0.83	**0.87**	0.85	0.85
10^{-4}	0.85	0.83	0.86	0.86	0.85	0.69	0.73	0.76	0.75	0.75	0.86	0.83	**0.87**	0.85	0.85
10^{-3}	0.84	0.83	**0.86**	0.86	0.86	0.69	0.73	0.76	0.75	0.75	0.86	0.83	**0.87**	0.85	0.85
10^{-2}	0.79	0.83	0.85	0.85	0.85	0.68	0.71	0.75	0.76	**0.76**	0.84	0.83	0.87	0.84	0.83
10^{-1}	0.78	0.73	0.78	0.80	0.79	0.59	0.61	0.68	0.70	0.71	0.83	0.83	0.85	0.86	0.86

g/σ	$D_1(D_2, D_3)$					$D_2(D_1, D_3)$					$D_3(D_1, D_2)$				
	10^3	10^4	10^5	10^6	10^7	10^3	10^4	10^5	10^6	10^7	10^3	10^4	10^5	10^6	10^7
10^{-10}	0.82	0.86	**0.87**	0.85	0.85	0.66	0.77	0.79	**0.79**	0.79	0.83	0.82	**0.90**	0.78	0.76
10^{-5}	0.82	0.86	**0.87**	0.85	0.85	0.66	0.77	0.79	**0.79**	0.79	0.83	0.82	**0.90**	0.78	0.76
10^{-4}	0.82	0.86	**0.87**	0.85	0.85	0.66	0.77	0.79	**0.79**	0.79	0.83	0.82	**0.90**	0.78	0.76
10^{-3}	0.82	0.85	**0.87**	0.85	0.85	0.66	0.78	0.79	**0.79**	0.79	0.82	0.82	**0.90**	0.78	0.76
10^{-2}	0.79	0.83	0.85	0.86	0.86	0.66	0.78	0.78	0.77	0.78	0.82	0.83	0.88	0.78	0.76
10^{-1}	0.74	0.68	0.76	0.77	0.77	0.62	0.69	0.75	0.75	0.75	0.78	0.84	0.85	0.83	0.77

Table 3.6: **Classification accuracies by all parameters.** The values in the big table are the classification accuracies on three real datasets for all combinations of parameters with the best result bold. The datasets in parentheses are used as training sets. The results are quite stable as long as g is not too large and σ is not too small.

datasets to get three consensus CNV probe series. The multiple alignment method was then used to produce a *multiple probe alignment profile* of the three consensus probe series. The probe intensities in the profile are rescaled to be in the range [-1,1]. We then report the probe positions with an absolute intensity larger than 0.4 and identified 49 regions with an average length of 20,000 kb. We also used linear interpolation to get an average of all samples and also rescaled the value to be in range [-1,1]. To make a fair comparison with alignment method, we reported 708 center regions in the interpolation profile, each with absolute intensity values larger than 0.8. The positions that are centered at the 708 regions with length 10,000 kb were taken as the common CNV regions. The results of multiple probe alignment and interpolation are compared in Figure 3.5. Many of the common CNV regions identified by the two methods are similar, but the alignment profile is smoother. The interpolation method reported many

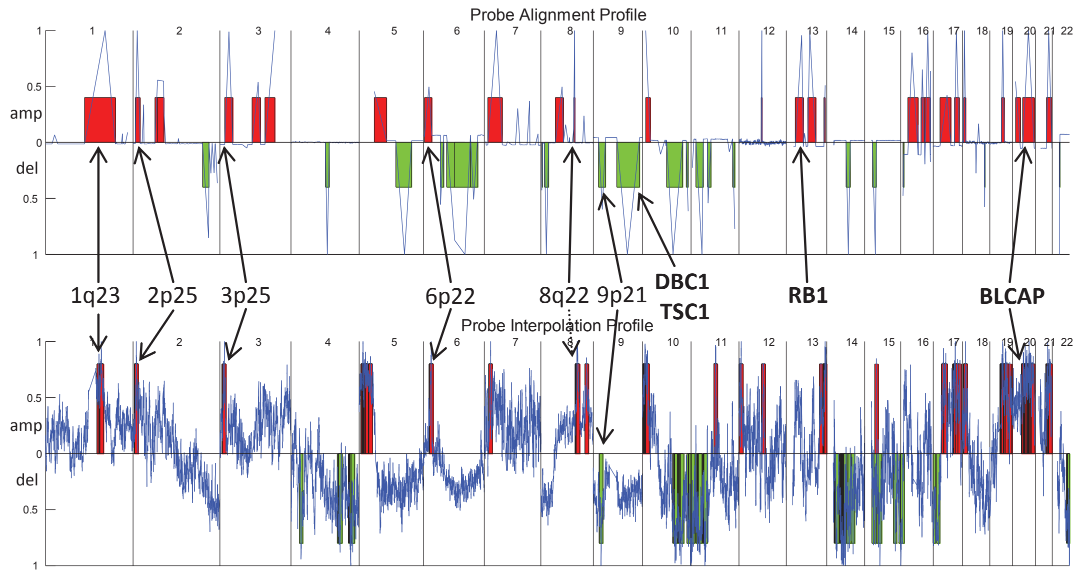


Figure 3.5: **Comparison of multiple probe alignment profile and interpolation profile.** The multiple probe alignment profile is above the interpolation profile. The common DNA amplification regions and deletion regions are plotted with blocks above or under the x-axis, respectively. The probes in the profiles are ordered by their locations on chromosomes (from chromosome 1 to chromosome 22) and the corresponding intensities are plotted by the curves. The vertical lines represent the chromosome separations. The locations of the known cancer-related CNVs and the four bladder cancer genes (DBC1, TSC1, RB1 and BLCAP) are marked with arrows.

more short fragments, many of which are false positives. One remarkable finding is that the multiple probe alignment captured four known bladder cancer genes in the common CNV regions, while the interpolation only identified one. We applied the multiple alignment algorithm to align all samples in the three datasets and compared the results with interpolation in Figure 3.6.

The multiple probe alignment detected four bladder cancer genes, DBC1, TSC1, RB1 and BLCAP, and several common amplification/deletion regions associated with bladder cancer. DBC1 (deleted in bladder cancer 1) is a gene that shows loss of heterozygosity in some bladder tumors. TSC1's somatic mutations are associated with the tumors cases with one copy of the TSC1 gene missing due to a deletion in chromosome 9. The two genes were identified at the end of the second deletion regions in the alignment profile on chromosome 9. RB1 with mutations, believed to contribute to the development of

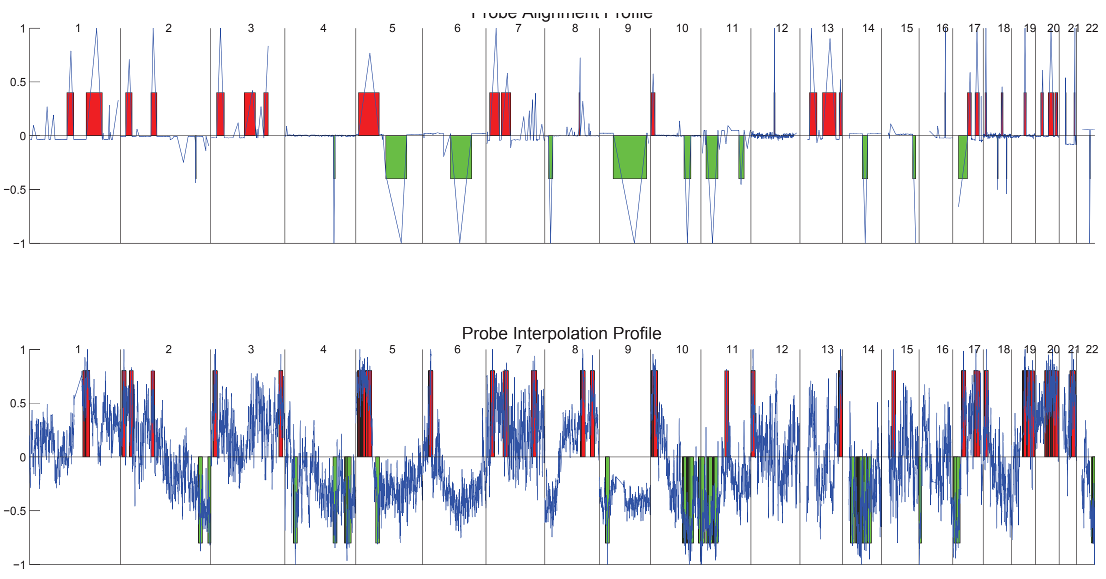


Figure 3.6: **Comparison of multiple probe alignment profile and interpolation profile.** The multiple probe alignment profile is above the interpolation profile. The red blocks denote the common DNA amplification regions and the green blocks denote the common DNA deletion regions. The probes in the profiles are ordered by their locations on chromosomes (from chromosome 1 to chromosome 22) and the corresponding intensities are plotted in blue curves. The vertical black lines represent the chromosome separations. This profile is different from that in the paper in that it is generated by **all** samples in the datasets instead of only samples with high tumor grades.

bladder cancer, is identified at the first amplification region on chromosome 13. BLCAP (bladder cancer associated protein), believed to be involved in the cancercinogenesis, is identified at the second amplification region at chromosome 20. Besides these bladder cancer genes, the multiple probe alignment also identified some common amplification and deletion regions associated with bladder cancer. For example, we identified an amplification event at location 6p22 with a main target gene E2F3 [57]. Deletions of part of or all chromosome 9 are common events in bladder tumors and we identified these events including a known location 9p21. The results are compared with previous studies on the datasets [39, 55] in Table 3.7. The multiple probe alignment can retrieve most findings from these independent studies, which implies that probe alignment successfully integrated information from the three datasets. The multiple alignment also missed some amplification/deletion regions, many of which might be false positives that were manually identified in a very small number of samples.

Chr	CNV	Location (kb)	Region: Clone/Gene	Reference	IP
1	amp	109,770 - 196,034	1q24: RP11-193J5	[39]	N
			1q23	[55]	Y
2	amp	6,680 - 20,594	2p25: YWHAQ, GRHL1, HPCAL1 RRM2, KLF11	[55]	Y
			2p25: ITGB1BP1, CPSF3, ADAM17 YWHAQ, TAF1B, RRM2		Y
2	del	195,683 - 213,371	2q35: PECR*	[39]	N
3	amp	13,853 - 36,353	3p25: TATDN2*, SEC13L1*, VGLL4*	[55]	Y
5	del	103,124 - 147,258	5q21: EFNA5	[39]	N
			5q22-23: APG12L, AP3S1		N
6	amp	2,798 - 23,690	6p22: ID4, E2F3, CDKAL1, SOX4 LOC401237	[39]	Y
			6p22: ID4, MBOAT1, E2F3, CDKAL1 SOX4, PRL, HDGFL1	[55]	Y
8	del	2,615 - 4,726	8p23: MYOM2*, RP11-246G24*	[39]	N
8	del	10,133 - 21,594	8p23: MSRA, RP11-254E10	[39]	N
8	amp	91,893 - 95,685	8q22: SPAG1*, RNF19*	[39]	Y
			8q22: POLR2K*, RNF19*, PABPC1* YWHAZ*, NCALD*	[55]	Y
9	del	14,417 - 34,511	9p23: RP11-264O11*	[39]	N
			9p22: RP11-109M15		Y
			9p21: CDKN2A, RP11-33O15		Y
			9p13: UNQ470, C9orf23, DCTN3 ARID3C, OPRS1, GALT		N
			9p13: IL11RA, CCL27, CCL19 CCL21, SPAG8, HINT2		N
			9p21: CDKN2A, CDKN2B, MTAP	[55]	Y
9	del	65,774 - 129,655	**DBC1**, **TSC1***		N
			9q21: TRPM3, RP11-8L13, RP11-14J9	[39]	N
			9q21 (refer to Figure 5B in [55])	[55]	N
10	amp	9,147 - 21,593	10p14: SFMBT2*, RP11-33J8*, RP11-35I11*	[39]	Y
12	amp	62,137 - 64,839	12q14: MDM2*, CTB-82N15*	[39]	Y
			12q15: FRS2*, CCT2* LRRC10*,VMD2L3*	[39]	Y
13	amp	25,534 - 47,854	**RB1**		N
13	amp	105,706 - 108,884	13q33: TNFSF13B	[39]	Y
19	amp	32,898 - 40,965	19q13: DPF1*, PPP1R14A*, SPINT2* KCNK6*, ACTN4*	[39]	Y
20	amp	28,098 - 60,257	**BLCAP**		Y

Table 3.7: **Common CNV regions and cancer genes detected by multiple probe alignment.** This table compares the findings of multiple probe alignment with the discoveries in [39] and [55]. Genes followed by * are not exactly in the region but they are less than 5,000kb away from the region. The last column shows whether a region is also discovered by interpolation (IP).

3.4 Discussions

Integration of diverse genomic datasets has become one of the central problems in cancer genomics. Most of previous work on arrayCGH data analysis focused only on segmentation of the probe series for deriving real CNV events. In this thesis, we introduce a general probe alignment framework to integrate arrayCGH datasets generated on different platforms. We demonstrated with experiments that the probe alignment-based approaches have a good potential to generate significantly improved classification performance and detect more accurate common CNVs. The results suggest that these approaches are powerful tools for integrative studies of multiple arrayCGH datasets. There are three technical issues in applying probe alignment: fast implementation, parameter tuning and kernel selection.

- We tested the fast probe alignment described in section 3.2.4 to show that only a small fraction of the entries in the dynamic programming matrix needs to be computed. Although the probe locations and log-ratios are highly variable in different platforms, we empirically observed that the time complexity is indeed close to linear in our experiments (Figure 3.2). However, it is also possible that a larger gap penalty is needed to reduce the time complexity significantly, which might lead to worse classification results. In this case, approximation is necessary to restrict the alignment to a smaller number of probe pairs to produce suboptimal alignment scores for classification.

- Intuitively, a good σ should be in the same scale of the average probe distances as estimated in Table 3.3 to distinguish the difference in the distances between matched probes. Small positive gs will result in identical alignment and similar good classification results with less impact from the gaps. We verified the intuitions by additional classification experiments in Figure 3.6. For a more rigorous treatment of the problem, based on a comprehensive analysis of the datasets (Table 3.3), we suggest a strategy that estimates the σ as the average probe distance and selects a gap penalty by a cross-validation in the training set, as a robust strategy to choose good parameters for SVM classification. For the multiple alignment case, we suggest the same strategy for choosing σ and taking a gap penalty g that

Additional Training	Target Dataset	Adding Constant	Nearest PSD
D_2	D_1	0.824 (0.077)	0.686 (0.101)
D_3	D_1	0.859 (0.069)	0.789 (0.083)
D_1	D_2	0.767 (0.116)	0.703 (0.125)
D_3	D_2	0.754 (0.120)	0.662 (0.118)
D_1	D_3	0.810 (0.122)	0.826 (0.135)
D_2	D_3	0.873 (0.109)	0.875 (0.116)
D_2, D_3	D_1	0.868 (0.061)	0.711 (0.093)
D_1, D_3	D_2	0.786 (0.115)	0.733 (0.121)
D_1, D_2	D_3	0.894 (0.105)	0.861 (0.107)

Table 3.8: **Comparison of two approaches to generate a valid kernel in cross-dataset validation.** The accuracies (standard deviations) are averages of 50 random 5-fold cross-validations. The first column shows the datasets used as additional training samples. The second column shows the target dataset for 5-fold cross-validation. We fixed parameters $g = 10^{-3}$ and $\sigma = 10^5$ in this comparison.

is decided by the allowable distance between the matched probes for generating reasonable multiple alignment. In practice, both strategies worked well.

- Finally, the probe alignment kernel is not guaranteed positive semi-definite. Besides adding a positive constant on the diagonal, another alternative that we tested is to use the nearest positive semi-definite matrix of the alignment-score matrix as the kernel matrix [58]. In [58], the authors described a way to find the nearest symmetric positive semidefinite matrix in the Frobenius norm in the following theorem:

  Theorem 3.4.1. *Let $A \in \mathbb{R}^{n \times n}$ and $B = (A + A^T)/2$. Let $B = UH$ be a polar decomposition ($U^T U = I, H = H^T \geq 0$). Then $X_F = (B + H)/2$ is the unique positive approximation of A in the Frobenius norm.*

  We also tested the classification performance by the nearest positive semi-definite matrix and reported the results in Table 3.8. However, the performance is not as good as adding a constant to the diagonal of the scoring matrix, which implies that the nearest PSD matrix is not a good kernel in the experiments.

A promising future direction is to combine segmentation approaches with alignment. The segmentation approaches detect the actual CNV intervals in each probe series and the alignment approach can be modified to align the intervals. We performed an initial segmentation analysis on the three bladder cancer datasets with the segmentation method proposed in [59]. At the suggested significance level, the segmentation detected very few intervals of amplifications and deletions that can be used for alignment. Thus, one difficulty is how to choose the parameters such as window length or significance levels for getting sensible segmentation results for alignment. In addition, aligning the detected CNV intervals in segmentation requires development of a new alignment approach to compare CNV intervals. Thus, the alignment approach in this paper is not directly applicable. Another possible extension is to distinguish regions with different probe density (gene-rich vs non-coding) with different normalization for further improvement of classification or common CNV detection. Certainly, this extension requires more sophisticated treatment of the alignment functions, which will introduce additional complexity to the problem.

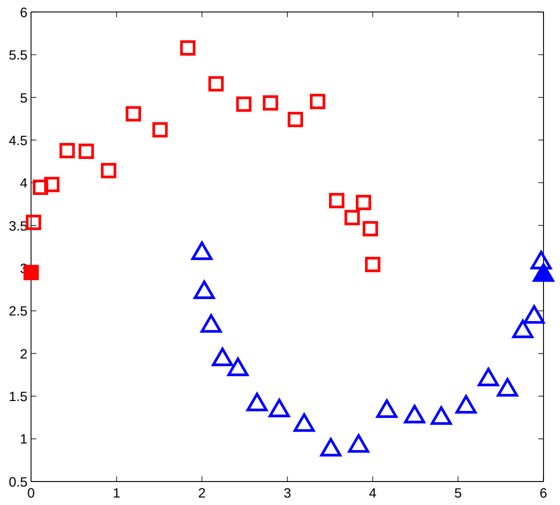
Chapter 4

Global Linear Neighborhoods for Efficient Label Propagation

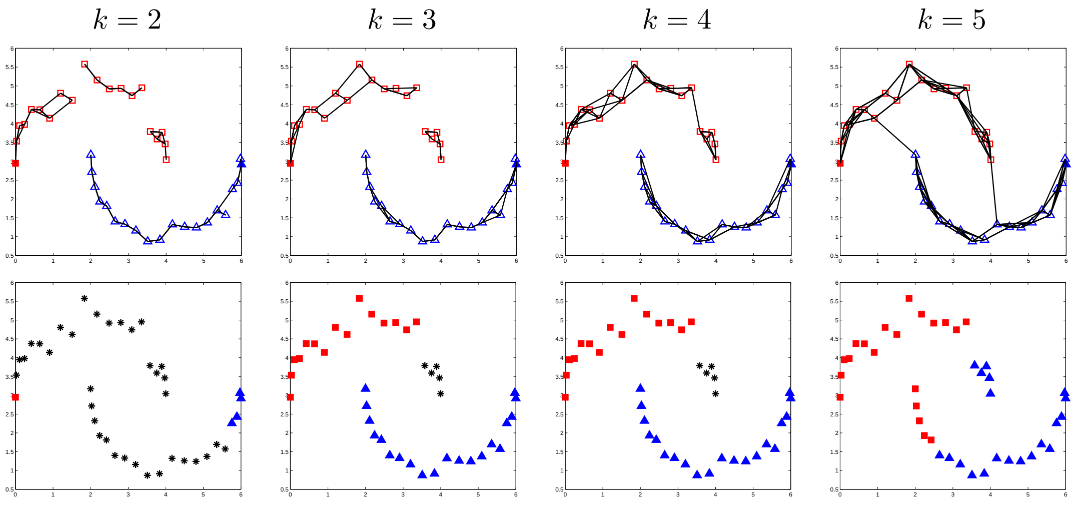
In this chapter, we propose to learn a nonnegative low-rank graph to capture global linear neighborhoods from data, under the assumption that each data point can be linearly reconstructed from weighted combinations of its direct neighbors and reachable indirect neighbors. Label propagation based on the learned low-rank graph showed better performance in several classification problems. The whole approach can be applied to semi-supervised learning problems on arrayCGH or gene expression data as well as general data from other domains.

4.1 Introduction

In many real world applications of data mining and machine learning, much more unlabeled data could be obtained and utilized with the labeled data to build more robust and accurate classification models. Graph-based semi-supervised learning algorithms perform a combined analysis of labeled and unlabeled data in a similarity graph between the data points [60, 61, 62, 63]. The algorithms initialize the vertices of the labeled data with +1/-1 and the vertices of the unlabeled data with 0, and then assign labels to the unlabeled data by label propagation on graph. The label information on the vertices is iteratively propagated between the neighboring vertices and the propagation process will finally converge to the unique global optimum minimizing a quadratic criterion [64].



(A) A two-moon dataset



(B) Local linear neighborhoods and classification results

Figure 4.1: A toy example of isolated graph components or incorrect neighborhoods resulted by local linear neighborhood selection. (A) The original dataset contains data points in two-moon clusters. Only one data point (denoted by a solid square or triangle) in each class is labeled. (B) *LNP* algorithm proposed by [65] is applied to the dataset with $k = 2, 3, 4, 5$ nearest neighbors selected as linear neighbors for each data point. The four figures in the top row show the identified neighbors connected by edges. The four figures in the bottom row show the classification results after label propagation on the linear neighborhoods. The black asterisks are the data points that are not classifiable by label propagation.

These algorithms are based on the "cluster assumption": nearby data points in the graph should be labeled similarly and data points in the same global cluster in the graph should also be labeled similarly.

How to estimate the similarities between data points is a core problem in the label propagation algorithms. The commonly used measure is Gaussian kernel similarity, which is sensitive to the variance parameter. A more rigorous treatment of the problem is to learn a similarity measure based on relations between data points. Under the assumption of linear neighborhoods, i.e. each data point can be reconstructed by a weighted linear combination of its neighbors, [65] proposed to represent the graph by a sparse matrix of the positive linear coefficients for reconstructing a data point by the k-nearest neighbors. Furthermore, [66] and [67] proposed to select the reconstruction neighbors with L_1 norm constraint to learn a sparse matrix. A limitation of these methods is that the neighbors are selected "locally" — the decision is only based on the individual relation between the reconstructed data point and the other data points. The local neighbors can lead to disjoint components and incorrect neighbors in the graph. We demonstrate the problems in a toy example in Figure 4.1. In this example, the algorithm proposed by [65] is applied to classify data points from two moon clusters shown in Figure 4.1(A). When k is too small, several disconnected subgraphs (components) are observed in the top plots in Figure 4.1(B). Since there is no label information that can be propagated into the components, the data points in the components are not classifiable by label propagation in the bottom plots in Figure 4.1(B). Note that, since the learned similarity matrix is not symmetric, a connected graph might not be strongly connected (Some of the vertices are not reachable due to the direction of the edges). When k is too large such as $k = 5$ in Figure 4.1(B), edges are introduced to connect data points in different classes. The classification performance will deteriorate with the wrong neighborhood information. In both cases, instead of providing a better similarity measure for label propagation, the new similarity matrix fails to capture the underlying global structure of the data.

In this paper, we propose to learn a nonnegative low-rank graph to capture global linear neighborhoods. The global linear neighborhoods assume that each data point can be linearly reconstructed from weighted combinations of its direct neighbors and indirect neighbors reachable by any steps of random walk. The influence of the neighbors is controlled by a decay factor inversely exponentially proportional to the number of steps of random walk. The larger the distance, the smaller the influence to the reconstruction. The global linear neighborhoods preserve the global cluster structures by exploring both the direct neighbors and the indirect neighbors, and thus, rarely contain isolated components. We demonstrate that global linear neighborhoods can be approximated by a low-rank factorization of the unknown reconstruction matrix. Therefore, even if the reconstruction matrix of global linear neighborhoods is not sparse, the low-rank factorization still provides a compressed representation of the graph. We further demonstrate that, to require all entries to be nonnegative edge weights in a graph for label propagation, the low-rank factorization matrix can be efficiently learned by a multiplicative update rule that minimizes the reconstruction error at a local optimum.

4.2 Related Work

In a given dataset $\mathcal{X} = \{x_1, \ldots, x_l, \ldots, x_n\}$ and a given label set $\mathcal{L} = \{1, \ldots, c\}$, $\{x_1, \ldots, x_l\}$ are data points in $\mathbb{R}^m$ labeled by $\{y_1, \ldots, y_l | y_i \in \mathcal{L}, i = 1, \ldots, l\}$ and $\{x_{l+1}, \ldots, x_n\}$ are unlabeled data points in $\mathbb{R}^m$. In graph-based semi-supervised learning, a similarity graph $G = (V, E)$ is first constructed from the dataset $\mathcal{X}$, where the vertex set $V = \mathcal{X}$ and the edges E are weighted by adjacency matrix W. By relaxing the class label variables as real numbers, label propagation algorithm iteratively updates the predicted label Z by

$$Z^{t+1} = \alpha W Z^t + (1 - \alpha) Y$$

where t is the time step, $\alpha \in (0, 1)$ is a constant and Y is a $n \times c$ binary matrix encoding the labeling of data points against each class [63]. To derive a reliable W for label propagation, simple strategies are suggested in [68]. A graph is constructed by choosing the nearest neighbors with

- ϵ-neighborhoods: Nodes i and j are neighbors if the Euclidean distance $\|x_i - x_j\|^2 < \epsilon$.
- k-nearest neighbors: Nodes i and j are neighbors if i is among the k-nearest neighbors of j or vice versa.

After the neighborhood of each node is decided, the edges between the neighbors are weighted by $W_{ij} = 1$ or by a Gaussian kernel $W_{ij} = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$ [63]. However, as it was analyzed by [65] and [66], the difficulty to determine the optimal σ causes unstable classification performance by label propagation.

Based on ideas similar to Local Linear Embedding (*LLE*) [69], [65], [66] and [67] proposed several approaches for estimating more robust similarity matrix W for label propagation. These methods are based on a common linear neighborhood assumption that each data point can be reconstructed by a weighted linear combination of its selected neighbors, i.e. $x_i = \sum_{j:x_j \in \mathcal{N}(x_i)} W_{ij} x_j$, where $\mathcal{N}(x_i)$ is the neighborhood of node i. Linear Neighborhood Propagation (*LNP*) proposed by [65] learns the W matrix by minimizing the reconstruction error:

$$\varepsilon(W) = \sum_i \left\| x_i - \sum_{j:x_j \in \mathcal{N}(x_i)} W_{ij} x_j \right\|^2 \tag{4.1}$$

subject to $\sum_{j:x_j \in \mathcal{N}(x_i)} W_{ij} = 1$, $W_{ij} \geq 0$ and $W_{ij} = 0$ if $x_j \notin \mathcal{N}(x_i)$. $\mathcal{N}(x_i)$ is chosen as the set of k-nearest neighbors of x_i in Euclidean distance. To avoid a fixed neighborhood for each point, Sparsity Induced Similarity measure (*SIS*) was proposed by [66] to learn W regularized by L_1 norm as follows:

$$\min_W \|W\|_{l_1} \tag{4.2}$$

subject to $x_i = \sum_{j:j \neq i} W_{ij} x_j$ for any i. By minimizing the L_1 norm, W is expected to be a sparse adjacency matrix. The approach proposed by [67] is similar to *SIS*, based on an algorithm for finding nonnegative sparse representation. [70] also proposed a slightly different objective function to deduce a sparse W and used a parameter-free label propagation algorithm for semi-supervised learning.

The "local neighborhood" assumption is similar to Markov assumption (conditional independence), where each sample is not related to any samples other than its k-nearest neighbors. A common problem of the above approaches is that, when k is very small or a certain sparsity level is expected, the graph edges defined by a few selected neighbors might define a weakly connected graph or more severely, separate the graph into several disjoint components (Figure 4.1). In general, the limitation is introduced by neighborhood selection with local measures. In other words, these algorithms tend to select a few most similar data points as neighbors and the connectivities of these neighbors with other nodes are not considered. To see this mathematically, each row of the sparse matrix representing the local neighborhoods is actually learned independently of the other rows as in LLE. Another limitation is that the learned W is not a symmetric matrix that could be interpreted for cluster structures. To ensure symmetry, W can be symmetrized as $\frac{W_{ij}+W_{ji}}{2}$ [66]. However, the symmetrization might not be reasonable when W_{ij} and W_{ji} are very different.

4.3 The Algorithm

Let X be the $n \times m$ data matrix where X_{ij} is the value of the jth feature in the ith data point. Let W be the symmetric $n \times n$ similarity matrix between the data points to be learned. Instead of explicit selecting k neighbors to make W sparse as in Eqn. (4.1), or enforcing the sparsity by minimizing the L_1-norm as in Eqn. (4.2), we propose to learn a rank-k non-negative symmetric W by the following optimization problem:

$$\min \mathcal{Q}(F) = \left\| X - FF^T X \right\|^2 \tag{4.3}$$

subject to $F_{ij} \geq 0$ where F is a $n \times k$ matrix and $W = FF^T$. We first explain the motivation of Eqn. (4.3) under global linear neighborhoods. An algorithm with a multiplicative update rule is then proposed to solve the problem. Later we give a intuitive explanation of the role of F in Eqn. (4.3). Finally, we show how the algorithm be applied for inductive learning.

4.3.1 Global linear neighborhood

Suppose W is the compressed representation of the graph. Let $S = D^{-1/2}WD^{-1/2}$ where D is a diagonal matrix with D_{ii} equal to the sum of the ith row of W. In global linear neighborhoods, we assume each data point can be reconstructed by itself, its neighbors, its neighbors' neighbors, etc. The influence of the neighbors is decayed by a factor α for each increase in distance. Specifically, We assume the data can be reconstructed by global linear neighborhoods as follows:

$$X = (1-\alpha)(X + \alpha SX + \alpha^2 S^2 X + \ldots) = (1-\alpha)\sum_{i=0}^{\infty}(\alpha S)^i X$$

where $\alpha \in (0,1)$. This assumption is a generalization of local linear neighborhood [69], in which each data point is only reconstructed by its selected neighbors (the $i=1$ term in global linear neighborhood). Since $\sum_{i=0}^{\infty}(\alpha S)^i X = (I-\alpha S)^{-1}X$, it is equivalent to find S to minimize

$$\left\|X-(1-\alpha)(I-\alpha S)^{-1}X\right\|^2. \tag{4.4}$$

Since $S = D^{-1/2}WD^{-1/2}$ is a real symmetric matrix, its eigen decomposition can be written as $S = Q\Lambda Q^T$ where Q is an orthonormal matrix and Λ is real and diagonal. By replacing S with $Q\Lambda Q^T$,

$$(1-\alpha)(I-\alpha S)^{-1}X = (1-\alpha)(I-\alpha Q\Lambda Q^T)^{-1}X = (1-\alpha)Q(I-\alpha\Lambda)^{-1}Q^TX. \tag{4.5}$$

Since S is similar to a stochastic matrix, the diagonal entries in Λ are in $[-1,1]$. Thus, $(I-\alpha\Lambda)^{-1}$ is a positive definite diagonal matrix with diagonal entries in $[\frac{1}{1+\alpha}, \frac{1}{1-\alpha}]$. To introduce a compressed representation, we only keep the k principle components of $(I-\alpha S)^{-1}$. Actually, the k principle components of $(I-\alpha S)^{-1}$ are the same k components of S corresponding to the k largest eigenvalues. Given the eigen decomposition in Eqn. (4.5), keeping the first k principle components of $(I-\alpha S)^{-1}$ is equivalent to keeping the top k entries only on the diagonal of $(I-\alpha\Lambda)^{-1}$, denoted by a $k\times k$ diagonal matrix $(I-\alpha\Lambda)^{-1}_{(k)}$. Let $F_{(k)} = \sqrt{1-\alpha}\,Q_{(k)}(I-\alpha\Lambda)^{-\frac{1}{2}}_{(k)}$, where $Q_{(k)}$ are the corresponding k columns for the k principle components in Q, and $F_{(k)}$ is a $n\times k$ matrix. Since $(I-\alpha S)^{-1} = Q(I-\alpha\Lambda)^{-1}Q^T \approx Q_{(k)}(I-\alpha\Lambda)^{-1}_{(k)}Q^T_{(k)}$, Eqn. (4.4) can be approximated by

$$\left\|X-(1-\alpha)(I-\alpha S)^{-1}X\right\|^2 \approx \left\|X-F_{(k)}F^T_{(k)}X\right\|^2,$$

which is similar to Eqn. (4.3). Thus, although it is not feasible to directly solve the non-convex Eqn. (4.4) for S, F learned by Eqn. (4.3) can be considered as a low-rank nonnegative factorization of $(1-\alpha)(I-\alpha S)^{-1}$ in Eqn. (4.4).

Mathematically, FF^T keeps most of the properties of $(1-\alpha)(I-\alpha S)^{-1}$. Specifically, both matrices contain only nonnegative entries and both are positive semi-definite and symmetric. In general it is not guaranteed that a learned F can be used to derive a valid S as a solution for Eqn. (4.4). The following derivations show the connection between S and F:

$$S \approx Q_{(k)}\Lambda_{(k)}Q^T_{(k)} \approx \frac{F(I-\alpha\Lambda)^{\frac{1}{2}}_{(k)}}{\sqrt{1-\alpha}}\Lambda_{(k)}\frac{(I-\alpha\Lambda)^{\frac{1}{2}}_{(k)}F^T}{\sqrt{1-\alpha}} = F\frac{(I-\alpha\Lambda)_{(k)}\Lambda_{(k)}}{1-\alpha}F^T.$$

Clearly F can be normalized by the unknown variable $\Lambda_{(k)}$, which are the k largest eigenvalues of S, to derive an approximated S. A strategy to estimate $\Lambda_{(k)}$ is to derive the top k eigenvalues of FF^T by eigen decomposition. However, this requires a good estimation that satisfies two conditions that $\Lambda_{(k)}$ is between $[-1, 1]$ and S is nonnegative, which is not always feasible.

Note that, to make the problem more general, we remove the requirement of orthogonal columns in F in Eqn. (4.3). After F is learned, we can compute a new similarity matrix $\hat{W} = FF^T$. Being symmetric and positive, $\hat{W}$ can be directly used as the similarity matrix for label prorogation.

4.3.2 Multiplicative update rule

To solve the optimization problem in Eqn. (4.3), we propose an algorithm similar to nonnegative matrix factorization [71]. We first calculate the gradient of F by

$$\frac{\partial \mathcal{Q}}{\partial F} = -2(X - FF^T X)X^T F - 2X(X^T - X^T FF^T)F.$$

The additive update rule from the gradient descent algorithm is

$$F_{ij} \leftarrow F_{ij} + \eta_{ij}[(X - FF^T X)X^T F + X(X^T - X^T FF^T)F]_{ij}$$

where $\eta_{ij} > 0$ is the learning rate. If we further assume that X contains only nonnegative values, the same technique used in [71] and [72] can be used to learn a nonnegative F by the following multiplicative update rule:

$$F_{ij} \leftarrow F_{ij} \times \sqrt{\frac{(2XX^T F)_{ij}}{(FF^T XX^T F + XX^T FF^T F)_{ij}}} \tag{4.6}$$

where $\times$ represents element-wise multiplication. The algorithm randomly initializes F to be a nonnegative matrix and then iteratively updates F with Eqn. (4.6) until convergence. Empirically, each update reduces the cost function in Eqn. (4.3) as shown in our experiments. Thus, the algorithm learns a local optimal solution. Let l be the number of iterations needed for convergence. By calculating the matrix multiplication in a proper order, the time complexity of the algorithm is $\Theta(l(m + k)nk)$, which is also $\Theta(lmnk)$ since $k \ll m$. Empirically the algorithm needs hundreds of iterations to converge. Assuming the number of iterations is bounded by a constant, the total time complexity is linear with the number of samples or features.

4.3.3 A clustering view

An intuitive interpretation of F by Eqn. (4.3) is that F reveals k-way clustering of dataset X and each F_{ij} tells the membership of the ith data point to the jth cluster. If F is the ideal binary membership indicator with $F_{ij} = 1$ if the ith data point belongs to the jth cluster and otherwise $F_{ij} = 0$, the columns in F are naturally orthogonal. Thus, F is not required to contain orthogonal columns in (4.3) since the cluster components in the optimal F are often almost orthogonal (as shown in our experiments). Since F can contain any nonnegative value instead of just binary values, it can be explained as soft cluster memberships as in EM clustering. It is also natural to define the pairwise similarities as $\hat{W} = FF^T$ since $W_{ij} = \sum_{l=1}^{k} F_{il}F_{jl}$ implies how likely the ith and jth data points are in the same cluster.

Our algorithm is different from spectral clustering [73] or Laplacian embedding [68, 74] because both the approaches require a predefined similarity measure to start with. Since we still assume each data point can be reconstructed linearly from its neighborhood and we explicitly minimize the reconstruction error $\left\|X - FF^T X\right\|^2$, F is more appropriate for label propagation than the membership matrix learned by clustering techniques.

4.3.4 Outline of GLNP algorithm

The complete Global Linear Neighborhood Propagation (GLNP) algorithm is described in Algorithm 3. *GLNP* first learns the F matrix from data $\mathcal{X}$ using the multiplicative update rule in Eqn. (4.6). A normalized S matrix is then computed from F. Finally, S is used to run label propagation for semi-supervised learning.

4.3.5 Efficient inductive learning

Although *GLNP* is mainly proposed for transductive learning or semi-supervised learning, it can also be extended for efficient inductive learning. Given the learned optimal $n \times k$ matrix F for n samples $\{x_1, \ldots, x_n\}$ and the corresponding Z matrix obtained after label propagation, let f_i be a column vector obtained by the transpose of the ith row of F and z_i be the ith row of Z. For a new test point u, we first learn a $k \times 1$

```
Algorithm 3 GLOBAL LINEAR NEIGHBORHOOD PROPAGATION (GLNP)
Input: $\mathcal{X} = \{x_1, \ldots, x_l, \ldots, x_n\}$ is a set of $n$ data points in $\mathbb{R}^m$; $\{x_1, \ldots, x_l\}$ are labeled
    by $\{y_1, \ldots, y_l\}$ and $\{x_{l+1}, \ldots, x_n\}$ are unlabeled; rank $k$ and balancing parameter
    $\alpha \in (0, 1)$.
Output: The labels of the data points $\{x_{l+1}, \ldots, x_n\}$.
1: Construct a $n \times m$ matrix $X$ from $\mathcal{X}$ where $X_{ij}$ is the value of the data point $x_i$ at
   the $j$th dimension and rescale $X$ if necessary. Construct a $n \times c$ matrix $Y$ where
   $Y_{ij} = 1$ if $1 \le i \le l$ and $y_i = j$, and 0 otherwise.
2: Randomly initialize a nonnegative $n \times k$ matrix $F$. Use the multiplicative update
   rule
       $F_{ij}^{t+1} \leftarrow F_{ij}^{t} \sqrt{\frac{(2XX^TF^t)_{ij}}{(F^t{F^t}^TXX^TF^t + XX^TF^t{F^t}^TF^t)_{ij}}}$
   to update $F$ until convergence.
3: Construct a $n \times n$ matrix $S = D^{-1/2}FF^TD^{-1/2}$ where $D$ is a diagonal matrix with
   its $(i, i)$-element equal to the sum of the $i$th row of $FF^T$.
4: Initialize a $n \times c$ matrix $Z = Y$, and iterate until convergence
       $Z^{t+1} = \alpha SZ^t + (1 - \alpha)Y.$
5: Output the labels of each data point $x_i$ by $y_i = \mathrm{argmax}_j\, Z_{ij}$ for $i = l + 1, \ldots, n$.
```

column vector f_u from the following optimization problem:

$$\min_{f_u} \left\| x_u^T - \sum_{i=1}^{n} (f_u^T f_i) x_i^T \right\|^2 \tag{4.7}$$

subject to $f_u \ge 0$. We can further derive Eqn. (4.7) as follows:

$$\begin{aligned}\left\| x_u^T - \sum_{i=1}^{n} (f_u^T f_i) x_i^T \right\|^2 &= \sum_{j=1}^{m} \left\| x_{uj} - \sum_{i=1}^{n} (f_u^T f_i) X_{ij} \right\|^2 = \sum_{j=1}^{m} \left\| x_{uj} - \sum_{i=1}^{n} f_u^T (f_i X_{ij}) \right\|^2 \\ &= \sum_{j=1}^{m} \left\| x_{uj} - f_u^T \sum_{i=1}^{n} (f_i X_{ij}) \right\|^2 = \|Cf_u - x_u\|^2\end{aligned}$$

where C is a $m \times k$ matrix and the jth row ($j = 1, \ldots, m$) of C equals to $\sum_{i=1}^{n}(f_i^T X_{ij})$. C can be interpreted as the similarity matrix between the m features and the k cluster components. Eqn. (4.7) seeks the best f_u — a k dimensional embedding of x_u such that Cf_u can reconstruct x_u. Eqn. (4.7) is a standard nonnegative least-squares problem with

k variables. Since k is usually very small, the problem can be solved efficiently. After f_u is learned, by the inductive form of label propagation described in [64], predictions against each class can be calculated by

$$z_u = \sum_{i=1}^{n} f_u^T f_i z_i.$$

The predicted label of u is then $y_u = \text{argmax}_j z_{uj}$. In the inductive learning setting, we only need to precompute and save the $m \times k$ matrix C instead of the $n \times m$ data matrix X. Given $k \ll n$ in most cases, this is a significant saving in space, while other label prorogation algorithms such as [63] and [65] have to save the full data matrix X for inductive learning.

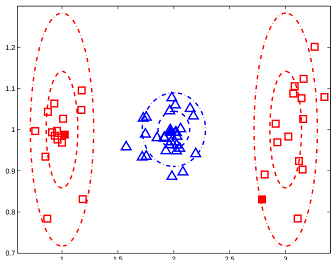
4.4 Experiments and Analysis

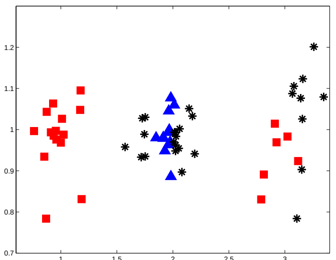
We experimented with simulations, four UCI datasets, one arrayCGH dataset and three high-dimensional microarray gene expression datasets.

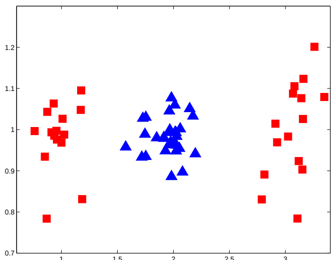
4.4.1 Simulations

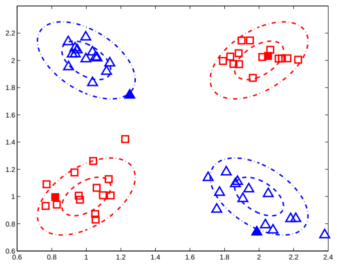
To compare global linear neighborhoods with local linear neighborhoods, we generated two toy datasets shown in Figure 4.2. In both datasets, data points in two classes (red squares vs. blue triangles) were generated from mixtures of different numbers of Gaussian distributions. 30 data points were sampled from each class. Both datasets are visualized with the mean and variance plotted by ellipses for each Gaussian distribution (Figure 4.2(A)&(D)). In the first dataset, the data points in the first class (red squares) were sampled from a mixture of two Gaussian distributions, and the data points in the second class (blue triangles) were sampled from a single Gaussian distribution (Figure 4.2(A)). In the second dataset, data points in the two classes (also red squares vs. blue triangles) were generated from two different mixtures of two Gaussian distributions (Figure 4.2(D)).

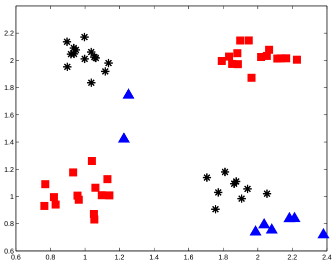
One data point from each cluster was randomly selected as the training set, denoted by solid squares or triangles. For *LNP*, we set $\alpha = 0.99$ in label propagation since it generated the best results on the two simulated datasets. For *GLNP*, the results were


(A) Dataset 1 (3 Clusters)


(B) *LNP* $k = 3$


(C) *GLNP* $k = 3$


(D) Dataset 2 (4 Clusters)


(E) *LNP* $k = 4$

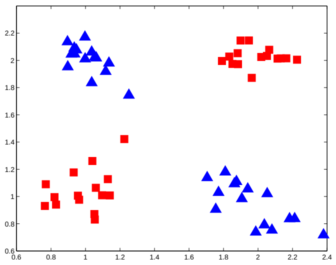
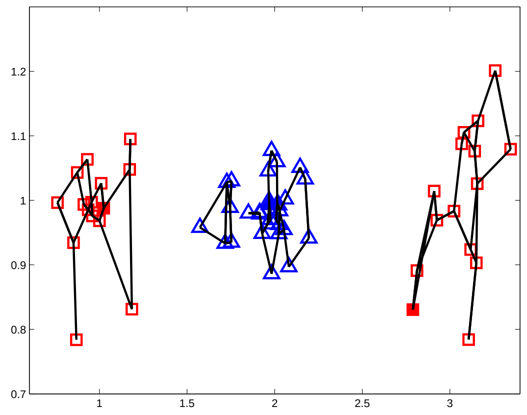

(F) *GLNP* $k = 4$

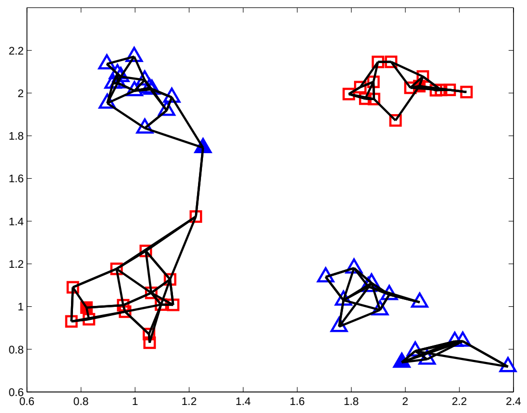
Figure 4.2: Simulation results of classification. (A) The first toy dataset; (B)&(C) Classification results by *LNP* and *GLNP* on the first dataset; (D) The second toy dataset; (E)&(F) Classification results by *LNP* and *GLNP* on the second dataset.

similar as long as α was not too large. Thus, we set $\alpha = 0.1$. For both *LNP* and *GLNP*, we set $k = 3$ for the first dataset and $k = 4$ for the second dataset to match the number of Gaussian clusters. Euclidean distance was used to select the k-nearest neighbors for *LNP*.

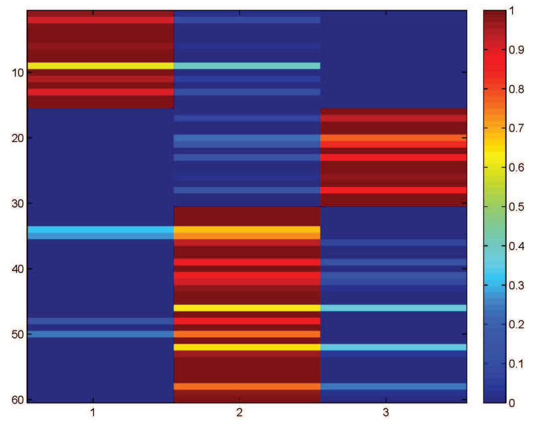
The classification results by *LNP* and *GLNP* are shown in Figure 4.2(B)&(E) and Figure 4.2(C)&(F). *LNP* was unable to classify some data points after label propagation, denoted by black asterisks. In these examples, the graphs defined by the adjacency matrix W learned by *LNP* are separated into several small disjoint components or not strongly connected. If there is no training data in a component, the data points in the component will remain unlabeled after label propagation. The graphs constructed by *LNP* for both datasets are visualized in Figure 4.3(A)&(B). Although *LNP* picked the k-nearest neighbors to learn W, some data points are connected to less than k neighbors due to the nature of the optimization problem defined in Eqn. (4.1). Even if some data points seem to be connected in Figure 4.3(A), those links cannot be used for label propagation because the data points are not strongly connected in the graph. In



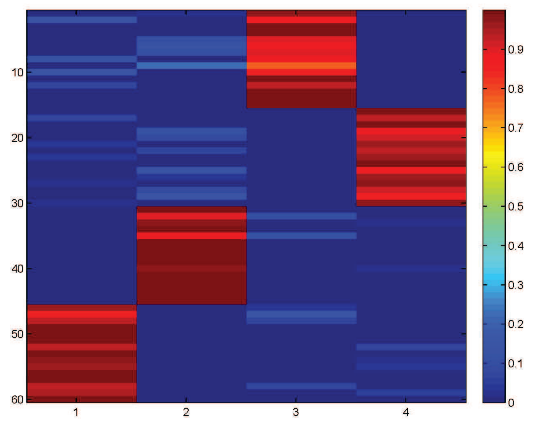
(A) Neighborhood ($k = 3$)



(B) Neighborhood ($k = 4$)



(C) Component ($k = 3$)



(D) Component ($k = 4$)

Figure 4.3: Simulation results of learned neighborhoods. (A)&(B) Neighborhoods learned by *LNP*; (C)&(D) Cluster components F learned by *GLNP*.

both datasets, *LNP* failed to make predictions for around one third to a half of the data points (Figure 4.2(B)&(E)). By increasing k, *LNP* could make perfect classification on the first simulated dataset. However in the second dataset, *LNP* would either have disjoint components or make wrong classifications for around one fourth of data points because wrong neighbors were selected (Figure 4.3(B)). In other words, *LNP* was never able to correctly classify around one fourth of the data points regardless of the choice of k.

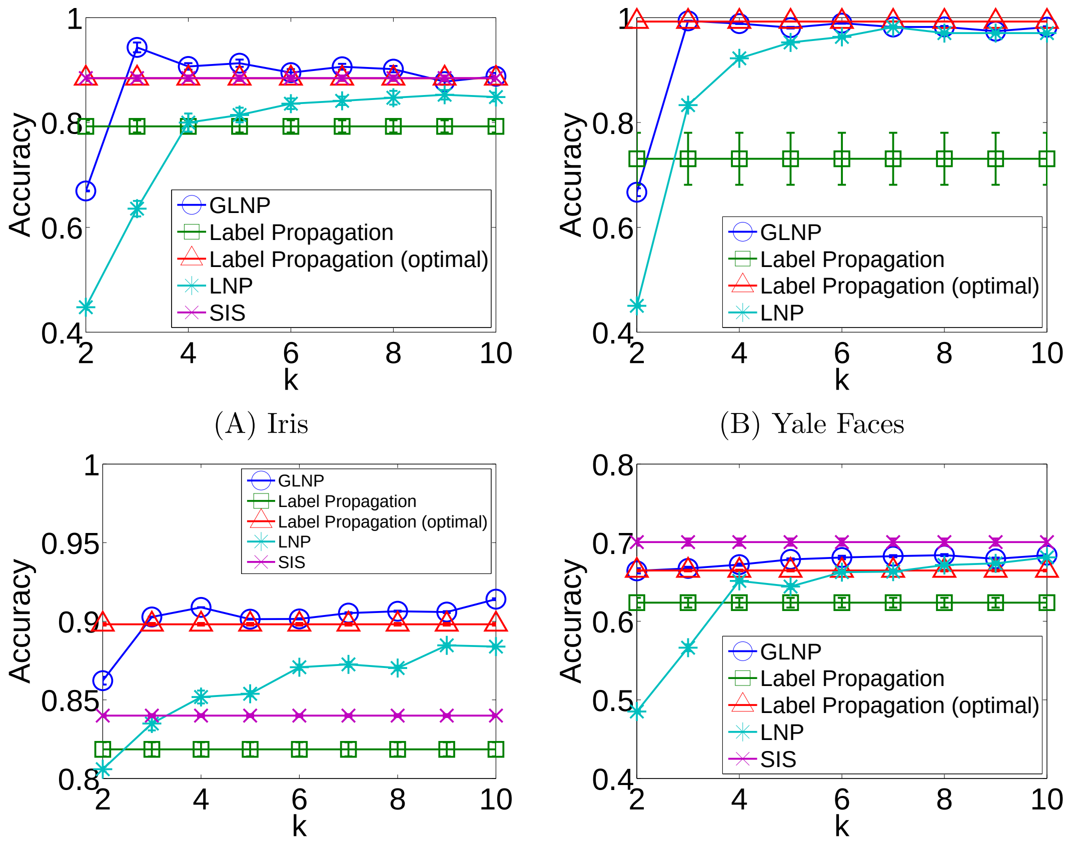
GLNP classified all data points perfectly in both datasets (Figure 4.2(C)&(F)). The cluster components F learned by *GLNP* are shown in Figure 4.3(C)&(D). The internal cluster structures of the data were learned correctly by the components distinguishing the Gaussian clusters in the two datasets.

4.4.2 Experiments on UCI datasets and biomedical datasets

In the experiments on the eight real datasets, we compared *GLNP* with Linear Neighborhood Propagation (*LNP*) [65], Sparsity Induced Similarity measure (*SIS*) [66] and the original label propagation with Gaussian Kernel (*LP*). As it was shown in [65], since the performance of *LP* is very sensitive to the variance parameter σ^2 and highly data-dependent, there is no good strategy to choose σ^2. Thus, in all experiments we report the results with both the best σ and $\sigma^2 = \text{mean}\left\{\frac{\|x_i - x_j\|^2}{2}\right\}$. The parameter α for label propagation was tuned to achieve optimal performance for all methods. We also tested different k parameters for a comprehensive comparison of *GLNP* and *LNP*. Assuming that at least K clusters are expected to distinguish K classes, we tested k from K to 10 where K is the number of classes in the dataset for both *LNP* and *GLNP*.

Four UCI datasets were tested. The first dataset is the Iris flower dataset. It contains 50 samples from each of three species of Iris flowers (*setosa*, *virginica* and *versicolor*) with four features. The second dataset is the Yale Face Database B from [75]. We used the processed data from [76], which includes images of individuals 2, 5 and 10, and down-sampled each image to 30×40 pixels. The processed data contains 1755 images with 1200 features. The third dataset is Glass Identification data (Glass). It has 9 continuous numerical features describing each of 214 instances in two classes: Window vs non-Window glasses. The fourth dataset is the Wine dataset, which contains 178 samples from three types of wines.

We also tested four high-dimensional biomedical datasets. The first dataset is a bladder cancer arrayCGH dataset with 57 patient samples proposed by [54]. Among the 57 samples, 31 of them had tumor grade G3 and 26 of them had lower tumor grade than G3. It was generated with a HumanArray 2.0 array and consists of 2,385 probes. Only probes from the first 22 chromosomes were used in the experiments, which ended up with 2,308 probes for each profile. The second dataset is the breast cancer gene expression profiles from [9]. It contains the expression profiles of 24,481 genes for 97 patient samples, among whom 51 patients had good prognosis and 46 patients had poor prognosis. The third dataset is the lung cancer gene expression data from [77]. It

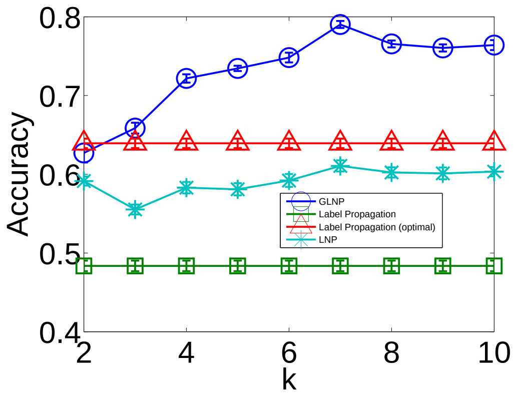


(A) Iris (B) Yale Faces (C) Glass (D) Wine

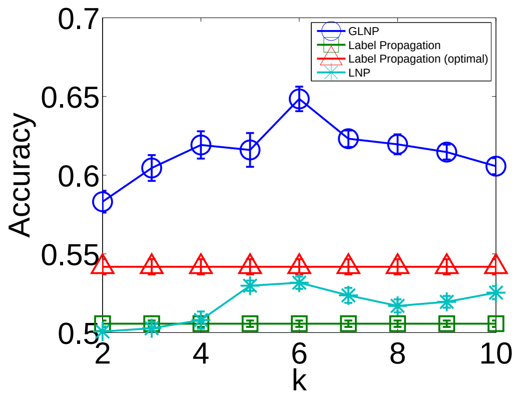
Figure 4.4: Classification results on the four UCI datasets. The variances are plotted as error bars.

contains the expression values of 22,283 genes from 58 tumor and 49 non-tumor tissues (from 20 never smokers, 26 former smokers, and 28 current smokers). The fourth dataset is another lung cancer gene expression data from [78]. It contains the expression profiles of 22,283 genes for paired adjacent normal-tumor samples from 27 patients underwent surgery for lung cancer, which add up to 54 samples for classification.

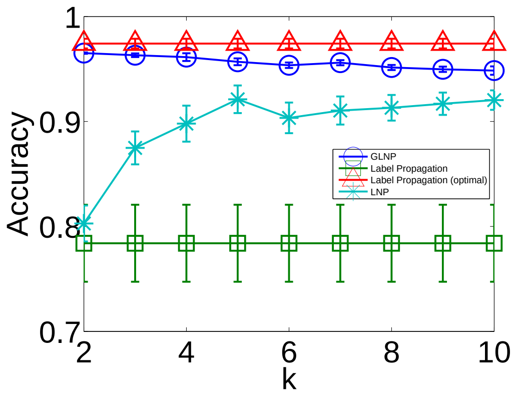
We randomly selected 1% samples as the training set for Iris flower dataset and Yale Face dataset and run random trials of classification multiple times to report the average accuracy for each method. Since the other six datasets are harder to classify, 50% samples were selected as the training set for the arrayCGH bladder cancer dataset and 10% samples were selected as the training set for the reminaing five datasets in



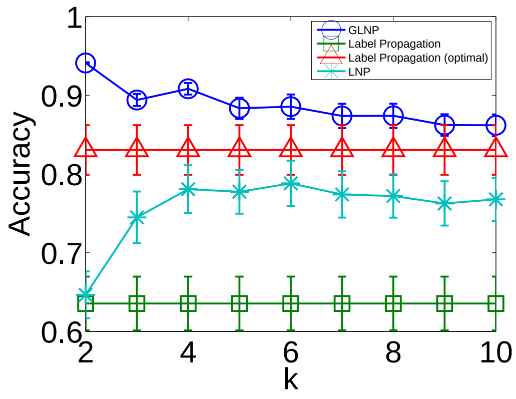
(A) Bladder cancer (Stransky et al.)



(B) Breast cancer (van 't Veer et al.)



(C) Lung cancer (Landi et al.)

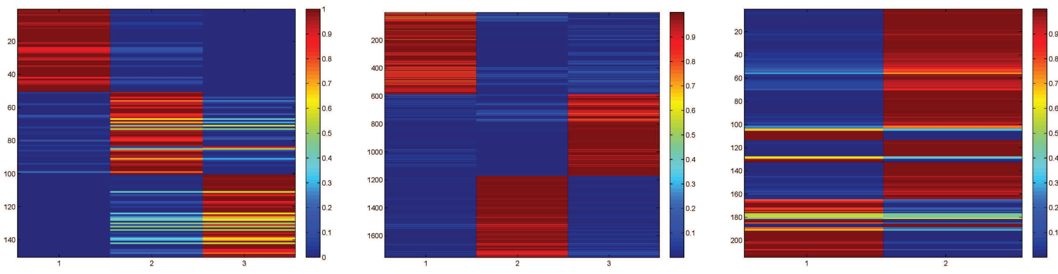


(D) Lung cancer (Su et al.)

Figure 4.5: Classification results on the four biomedical datasets. The variances are plotted as error bars.

each trial. On each dataset, we run 50 trials with different randomly selected training sets. We also make sure there is at least one training point for each class in each trial.

The classification results on the UCI datasets and the biomedical datasets are reported in Figure 4.4 and Figure 4.5, respectively. *GLNP* performed consistently better than *LNP* with almost any choice of k in the experiments on all the datasets. The lower variance of GLNP compared to other methods suggest that the performance of *GLNP* is more stable. Another observation is that the performance of *GLNP* is not very sensitive to parameter k as long as k is not too small. As *GLNP* interprets k as the number of clusters in the data, it is expected that when k is smaller than the number of classes, the performance of *GLNP* might not be as competitive in several cases. *LNP*

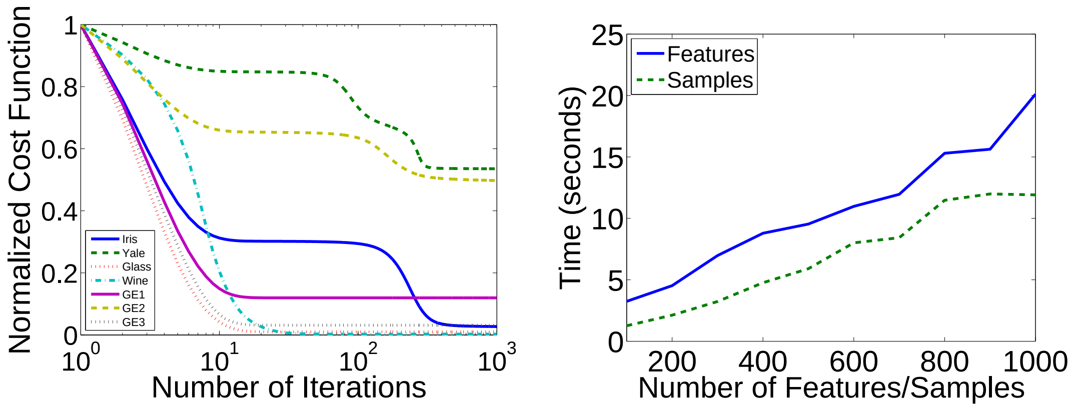


(a) Iris $k = 3$ (b) Yale $k = 3$ (c) Glass $k = 2$

Figure 4.6: The cluster components F learned by *GLNP* on three datasets.

performed reasonably well on relatively noise-free datasets because the label of a sample can be reliably predicted by just checking its local neighbors. However, when the noise in the data leads to wrong neighborhood relations as shown in the simulations, the global information is necessary to achieve better performance. This explanation is clearly illustrated by the results on the four biomedical datasets, where the performance of *GLNP* is more significantly better than *LNP*. *SIS* performed best on Wine dataset. However, *SIS* could not scale to Yale Faces and other biomedical datasets due to the low scalability of its optimization. In addition, it is infeasible to run *SIS* on datasets with a larger number of features than samples. Although *LP* performed well with the optimal σs, it is not possible to estimate the optimal σs from data. The default σs generated much worse results compared to the optimal σs.

To analyze the cluster components F learned by *GLNP*, we plot the components learned from the Iris, Yale Faces and Glass datasets in Figure 4.6. The three datasets were selected since they are less noisy and contain clearer cluster structures. k was chosen to be the same as the number of classes in the datasets. It is clear that each component represents one class in the three datasets. Figure 4.7(A) shows the convergence property of *GLNP*. Since the convergence rates are similar in the datasets under different choices of k, we plot the cost function in each iteration with k equal to the number of classes on the four UCI datasets and the three gene expression datasets. In all cases, *GLNP* converged well within less than 500 iterations, and the cost function monotonically decreases after each iteration. We also tested subsets of Yale Face dataset with different numbers of samples or features ranging from 100 to 1000 to report the average run time of *GLNP* in Figure 4.7(B). The result shows that the run time of



(A) Convergence (B) Run time

Figure 4.7: Convergence and run time of *GLNP*. (A) Convergence of *GLNP* on the seven datasets (x-axis is in logarithmic scale); (B) Run time on Yale Face dataset with different number of samples and features.

GLNP is approximately linear with the number of samples or features.

4.5 Discussions

In this thesis, we introduce the concept of global linear neighborhood and *GLNP* algorithm for learning a similarity matrix used by label propagation. Compared with local linear neighborhoods, global linear neighborhoods preserve the global structures among the data points, and thus, constitute more robust and reliable similarity graphs for label propagation. The sparsity of the neighborhoods is implicitly enforced by learning k cluster components, instead of choosing local neighbors. Thus, global linear neighborhood and *GLNP* algorithm are fundamentally different from other existing approaches for learning similarity graph for label propagation based on local neighborhood assumption.

GLNP algorithm adopts a multiplicative update rule to find a nonnegative factorization matrix of the global similarity matrix. The strategy is similar to nonnegative matrix factorization and thus, scales reasonably well on large datasets. Empirically, *GLNP* converges fast in all the experiments reported in this study (Figure 4.7(A)). *GLNP* was also able to scale up on datasets with more than 10,000 samples (results not shown). Thus, *GLNP* possesses better scalability compared with *SIS*.

Neighborhood learning for label propagation is different from general metric learning [79, 80] or covariance matrix learning by General Gaussian Models. For example, [81] proposed a semi-supervised sparse metric learning algorithm by first performing affinity propagation and then optimizing the sparse metric by alternating linearization. The sparsity in their paper is defined on the metric instead of the affinity matrix. However, the purpose of sparse graph representation learning is to learn the best sparse or compressed graph that preserves the graph structure for effective label propagation. Thus, the new graph needs to be non-negative, preferably highly connected and sparse or low-rank. These properties might not be easily achieved by extending Gaussian models because the learned covariance matrix cannot contain diagonal blocks and needs to be sparse or low-rank. *GLNP* is also fundamentally different from the graph compression approaches based on a preprocessing with clustering. For example, the authors in [82] proposed to first find k cluster centers and then construct a local linear embedding of data points by closest cluster centers. Both *GLNP* and [82] take product of embedding matrix as graph matrix. However, in *GLNP* the embedding dimension k equals to or is slightly larger than the number of classes, and the approach in [82] relies on the clustering for the number of anchor points which is much larger than the number of classes. In philosophy, *GLNP* is global embedding and the approach in [82] is still a "local" embedding but by cluster centers. The sparsity is obtained by the low-rank embedding for *GLNP* but by the nearest anchor points for the approach in [82]. Although the clustering step was introduced to compress the data, it is in general a difficult and unreliable preprocessing step, which requires another similarity measure between the data points.

Although the multiplicative update rule assumes that the data is nonnegative, *GLNP* can also handle data with both positive and negative values by using the additive update rule with nonnegative constraints. Furthermore, by rescaling the data to be in a positive interval, the faster multiplicative update rule can be applied and empirically, *GLNP* also produced significantly better results on the rescaled datasets. Finally, it is not always possible to find the corresponding similarity matrix (S) used to compute the global linear neighborhoods, with the cluster components F. However, it is less a concern since our focus is on finding a sparse global similarity matrix for label propagation to improve classification performance.

Chapter 5

Sparse Group Selection on Fused Lasso Components

In this chapter, we propose a latent feature model that couples sparse sample group selection with fused lasso on CNV components to identify group-specific CNVs. Assuming a given group structure on patient samples by clinical information, sparse group selection on fused lasso (SGS-FL) identifies the optimal latent CNV components, each of which is specific to the samples in one or several groups. The group selection for each CNV component is determined dynamically by an adaptive algorithm to achieve a desired sparsity.

5.1 Introduction

Array-based comparative genomic hybridization (ArrayCGH) measures the relative copy number at thousands of locations by probes along the chromosome. For each test sample, the copy number data generated by arrayCGH experiments is generally a series of log intensity ratios by comparing the intensity of the test sample to the intensity of the reference sample at all probe locations. The values in the series are usually ordered according to the physical locations of the corresponding probes along the chromosome. One important task in arrayCGH data analysis is to identify the regions of the chromosome where CNV events are observed. Since CNVs are usually caused by deletion and duplication of relatively large regions of the genome, CNV events tend to occur in contiguous blocks instead of several small fragments. Thus, the intensities of adjacent probes are usually correlated and the probes located in the same block of a CNV event are expected to share the same or similar intensities.

Several change-point detection methods were proposed for single-sample analysis [59, 83]. Since CNVs identified from one single sample might not be reliable, it is more important to identify CNVs from multiple samples in the same cancer study set. Most multi-sample analysis methods report CNVs that are shared by majority of samples in the study. Since these methods ignore the differences in phenotypes among individual samples, the identified CNVs contain both common CNVs and disease-specific CNVs. The phenotype information can help to distinguish disease-specific CNVs from common CNVs. However, it is a challenging problem to integrate phenotype information in multi-sample CNV analysis.

5.2 Related Work

Since CNV data are series of log intensity ratios at the sampled locations (probes), the adjacent probe locations are more likely to be associated in the same CNV event. For single-sample CNV detection, fused lasso appears to be a promising model [84]. In the fused lasso, L_1-norm is used in the penalty term to smooth the data by encouraging sparsity of the data and also the sparsity of the change points. Specifically, the method finds the segmented series β by solving the optimization problem

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_i (y_i - \beta_i)^2 \right\}$$

subject to $\sum_j |\beta_j| \leq s_1$ and $\sum_j |\beta_j - \beta_{j+1}| \leq s_2$, where y_i is the log2 ratio measurement of the ith probe and β_i is the corresponding value after smoothing. Here, s_1 controls the overall sparsity of CNV regions (the number of nonzero values in β) and s_2 controls the number of CNV alterations (the number of change points between the adjacent probes). By examining the non-zero values in β, CNVs can be identified for the sample. The procedure is repeated for all the samples and the resulted segmented series are aggregated to report the identified common CNVs.

For multi-sample CNV detection, all samples are analyzed simultaneously in one optimization framework to identify the amplification or deletion regions shared across all samples. For example, for p copy-number profiles of length n, [85] and [86] proposed the following optimization problem

$$\min_{U \in \mathbb{R}^{n \times p}} \|Y - U\|^2 + \lambda \sum_{i=1}^{n-1} \|U_{i+1,\bullet} - U_{i,\bullet}\|,$$

where Y is the $n \times p$ CNV profile matrix, U is the de-noised segmentation approximating Y and $U_{i,\bullet}$ is the ith row of U. A fast group least-angle regression (LARS) algorithm can be applied to solve the optimization framework approximately to detect shared change-points from the multiple CNV profiles. Since the change-points are detected from all profiles in the framework, it is expected to be more accurate than detecting change-points independently from each CNV profile.

Under the same motivation that CNVs are usually shared by multiple samples, instead of approximating the profile matrix Y by a segmentation matrix of the same size, another more advanced modeling is to detect the shared CNVs as latent fused features by low-rank matrix factorization decomposed from Y. For example, the widely used dimensionality reduction method principal component analysis (PCA) can decompose Y into orthogonal principle components. The projection of Y to a low-dimensional space obtains coefficients of the principle components to preserve the variance. The idea of presenting the main information of data in a low-dimensional space is effective in reducing noise in high-dimensional data. However, practically it is not feasible to interpret the principle components as CNVs since the principle components cannot be explained as CNV patterns without fusing the adjacent features with lasso.

More recently, a Fused Lasso Latent Feature Model (FLLat) was proposed by [87] for detecting latent CNV components. FLLat took advantage of the shared information among all samples and further identified some specific relationships between samples on real breast cancer arrayCGH data. For the profile matrix Y with S samples and L probes, FLLat decomposes it as a weighted sum of a fixed number of latent feature components, which are smoothed by fused lasso. The corresponding optimization problem for FLLat is

$$\min_{\Gamma,\Theta} \sum_{s=1}^{S} \sum_{l=1}^{L} \left(Y_{ls} - \sum_{j=1}^{J} \Gamma_{lj} \Theta_{js} \right)^2 + \lambda_1 \sum_{j=1}^{J} \sum_{l=1}^{L} |\Gamma_{lj}| + \lambda_2 \sum_{j=1}^{J} \sum_{l=2}^{L} |\Gamma_{lj} - \Gamma_{l-1,j}|$$

subject to $\sum_{s=1}^{S} \Theta_{js}^2 \leq 1$ for each j, where J is the number of latent features and Y_{ls} is the log intensity ratio of the lth probe for the sth sample. This model minimizes the sum of the square errors as well as the fused lasso penalties on the latent feature Γ. It is clear that the model does not assume any structure on the weights of the latent fused lasso components Θ.

5.3 Method

5.3.1 Motivation

It is well acknowledged that there is often group-structured prior information on the samples in cancer genomic datasets, which accounts for the heterogeneities among the patients. For example, the samples can be grouped by different tumor grades or stages, or by survival and metastatic status. The samples in each or some of the groups might be associated with CNV components that are only associated with the samples in the group(s). It is actually known that samples with different phenotypes also show different frequencies of CNVs. For example, low and medium grade tumors of bladder cancer generally contain few changes [55]. Thus, it is more biologically interesting to identify CNV patterns for the samples under the group-structure given by prior information. To achieve this objective, we propose *sparse group selection on fused lasso components* (SGS-FL) for integrating group information on the fused lasso components. SGS-FL assumes a group structure on the component coefficients and attempts to select only a small number of groups for each component. SGS-FL also requires that the coefficients of latent features to be non-negative for better distinguishing CNVs as regions with amplifications or deletions. The outline of SGS-FL is given in Figure 5.1.

5.3.2 Notation

We denote the arrayCGH profiles on a chromosome as a $m \times n$ matrix Y where m is the number of samples and n is the number of probes. Y_{ij} is the $\log_2$ intensity ratio measured for the ith sample on the jth probe. Assume that the probes are ordered by their positions on the chromosome. The objective is to decompose Y into a $m \times K$ matrix X and a $K \times n$ matrix W such that $\frac{1}{2}\left\|Y - XW\right\|_F^2$ is minimized where K is the number of the components of latent features. Here W are the components of the latent

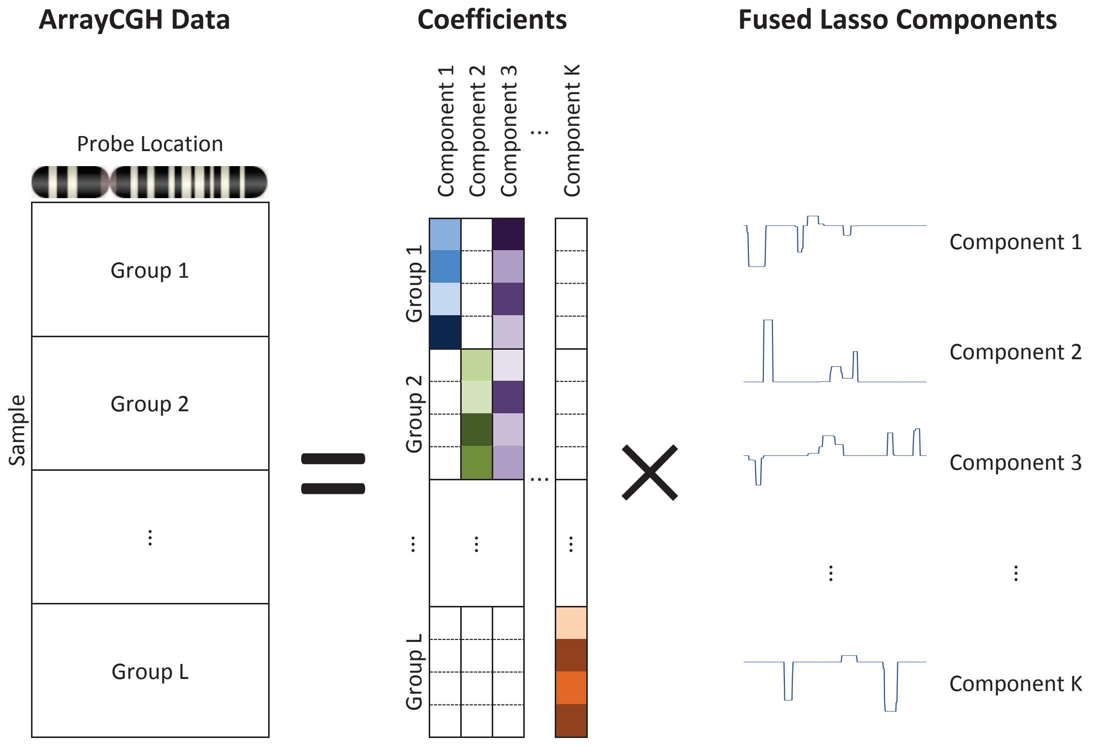


Figure 5.1: Outline of SGS-FL. The arrayCGH data is factorized into the coefficient matrix and K CNV components. After the factorization, each CNV sample can be reconstructed by the sum of the components weighted by the coefficients. The fused lasso penalty on each component encourage the step-function pattern to model real CNV events. The patient samples are divided into L groups. For each component, only the samples in the selected groups will have nonzero weights. For example, only samples in group 1 have nonzero weights for component 1 and only samples in group 1 and 2 have nonzero weights for component 3. The group selection enforces the sparseness of the coefficients by groups.

features and X are the coefficients of the components for all samples. Each sample $Y_{i,\bullet}$ is reconstructed by the linear weighted sum of latent features $\sum_k X_{ik} \cdot W_{k,\bullet}$. We further assume that the m samples are categorized into L disjoint groups as $G = \{g_1, g_2, \ldots, g_L\}$ where $g_l \subset \{1, 2, \ldots, m\}$ is the set of indexes of all samples in the lth group.

5.3.3 Regularization framework

We propose the following optimization problem to minimize reconstruction error and fused lasso under the group selection constraints:

$$\min_{X,W} \frac{1}{2} \|Y - XW\|_F^2 + \lambda_1 \sum_{k=1}^{K} |W_{k,\bullet}| + \lambda_2 \sum_{k=1}^{K} \sum_{j=2}^{n} |W_{kj} - W_{k,j-1}| \tag{5.1}$$

subject to

$$\begin{aligned} b_{lk} X_{g_l,k} &= 0 \text{ for } l = 1, \ldots, L \text{ and } k = 1, \ldots, K \\ X_{ik} &\geq 0 \text{ for } i = 1, \ldots, m \text{ and } k = 1, \ldots, K \\ \sum_{i=1}^{m} X_{ik}^2 &= 1 \text{ for } k = 1, \ldots, K \end{aligned}$$

where b_{lk} is a binary indicator variable of selecting the lth group for the kth latent feature component and $X_{g_l,k}$ is a sub-vector of $X_{\bullet,k}$ with indexes $i \in g_l$. If the m samples are divided into L strictly non-overlapping groups, X can be rearranged into a matrix of $L \times K$ vectors as

$$X' = \begin{pmatrix} X_{g_1,1} & \cdots & X_{g_1,K} \\ \vdots & \ddots & \vdots \\ X_{g_L,1} & \cdots & X_{g_L,K} \end{pmatrix}.$$

b_{lk} is introduced for group selection on each fused lasso component $W_{k,\bullet}$. Specifically, if $b_{lk} = 1$, all the coefficients of $X_{g_l,k}$, the kth latent feature component from the lth group, need to be 0; otherwise if $b_{lk} = 0$, the coefficients of $X_{g_l,k}$ can be any nonnegative weights. In other words, group l is selected for component k if $b_{lk} = 0$. The b_{lk} acts as a gating of the weights $X_{g_l,k}$ on the component $W_{k,\bullet}$ and $W_{k,\bullet}$ is only specific to the samples in the selected groups (with $b_{lk} = 0$). In the optimization problem, b_{lk}s are chosen dynamically in each iteration of the optimization algorithm according to a fixed global parameter $r \in [0, 1]$.

5.3.4 Sparse group selection

Given the parameter $r \in [0, 1]$ and the current weights and components, b_{lk}s are determined for each latent feature component separately. We first define variance factors

$v^{(k)} = (Y - X_{\bullet,\neq k} W_{\neq k,\bullet}) W'_{k,\bullet}$ where $X_{\bullet,\neq k}$ is the matrix after removing the kth column from X and $W_{\neq k,\bullet}$ is the matrix after removing the kth row from W. Each $v^{(k)}$ evaluates the importance of the component $W_{k,\bullet}$ to the reconstruction error. The larger the $v^{(k)}$, the more important the $W_{k,\bullet}$. Then, the role of a group l to the component k is evaluated as

$$\gamma_l^{(k)} = \frac{\left\| v_{g_l}^{(k)} \right\|}{\sqrt{|g_l|} \left\| W_{k,\bullet} \right\|^2} \tag{5.2}$$

where $|g_l|$ is the cardinality of g_l and $v_{g_l}^{(k)}$ is the sub-vector of $v^{(k)}$ with indexes in g_l. The importance vector for group l is normalized by the size of group l and the 2-norm of component $W_{k,\bullet}$. Then, we can sort the L groups by $\gamma_l^{(k)}$ such that $\gamma_{q_1}^{(k)} \geq \gamma_{q_2}^{(k)} \geq \cdots \geq \gamma_{q_L}^{(k)}$. Based on the ranking of the groups, b_{lk}s are calculated by

$$b_{q_l,k} = \begin{cases} 0 & \text{if } l = 1 \text{ or } \sum_{s=1}^{l-1} \gamma_{q_s}^{(k)} / \sum_{s=1}^{L} \gamma_{q_s}^{(k)} < r \\ 1 & \text{otherwise.} \end{cases} \tag{5.3}$$

For each component $W_{k,\bullet}$, at least one group is selected and additional groups are selected based on their importance proportional to the total importance. More intuition of group selection by the ranking of $\gamma_{q_l}^{(k)}$ and its connection to group lasso are discussed in section 5.3.6.

5.3.5 Alternative optimization

Eqn. (5.1) can be solved with alternative optimization to get an empirical solution. We alternate between fixing W and solving for X and vice versa until both W and X do not change anymore in the iterations. The complete SGS-FL algorithm is described in Algorithm 4. W is initialized with the first K principle components of Y computed by PCA (line 1). Then, we solve X column by column given W (line 3-14) and solve W row by row given X (line 15-19). The algorithm iterates until both X and W converge. Specifically, for the kth column of X, we first compute $\{b_{lk}\}$ as in Eqn. (5.3) (line 5-11), and then update $X_{\bullet,k}$ by solving the following sub-optimization problem on line 12:

$$\min_{X_{\bullet,k}} \frac{1}{2} \left\| Y - XW \right\|_F^2 \tag{5.4}$$

subject to $b_{lk} X_{g_l,k} = 0$, $X_{ik} \geq 0$ and $\sum_{i=1}^{m} X_{ik}^2 = 1$. Eqn. (5.4) can be solved with standard quadratic optimization techniques. This procedure is repeated iteratively for each column of X until X does not change anymore. Similarly, for each row of W, we

Algorithm 4 Sparse Group Selection on Fused Lasso components (SGS-FL)

Input: arrayCGH data $Y \in \mathbb{R}^{m\times n}$, the number of latent features $K \in \mathbb{Z}^+$, the group-sparsity-controlling parameter $r \in [0,1]$, the parameters $\lambda_1, \lambda_2 \in \mathbb{R}^+$ for lasso and fused lasso penalties.

Output: The non-negative coefficient matrix $X \in \mathbb{R}^{m\times K}$, the latent feature matrix $W \in \mathbb{R}^{K\times n}$.

```
1:  Initialize $W$ as the first $K$ principle components of $Y$ and $X$ as 0.
2:  repeat
3:    repeat
4:      for $k = 1, \ldots, K$ do
5:        $v^{(k)} \leftarrow (Y - X_{\bullet,\neq k} W_{\neq k,\bullet}) W'_{k,\bullet}$
6:        for $l = 1, \ldots, L$ do
7:          $\gamma_l^{(k)} \leftarrow \frac{\left\| v_{g_l}^{(k)} \right\|}{\sqrt{|g_l|} \left\| W_{k,\bullet} \right\|^2}$
8:        end for
9:        for $l = 1, \ldots, L$ do
10:         $b_{lk} = \begin{cases} 0 & \text{if } l = \text{argmax}_s \, \gamma_s^{(k)} \text{ or } \sum_{s \in \{l' | \gamma_{l'}^{(k)} > \gamma_l^{(k)}\}} \gamma_s^{(k)} / \sum_{s=1}^{L} \gamma_s^{(k)} < r \\ 1 & \text{otherwise.} \end{cases}$
11:       end for
12:       $X_{\bullet,k} \leftarrow \text{argmin}_{X_{\bullet,k}} \frac{1}{2} \|Y - XW\|_F^2$ subject to $b_{lk} X_{g_l,k} = 0$, $X_{ik} \geq 0$ and
          $\sum_{i=1}^{m} X_{ik}^2 = 1$
13:     end for
14:   until $X$ does not change
15:   repeat
16:     for $k = 1, \ldots, K$ do
17:       $W_{k,\bullet} \leftarrow \text{argmin}_{W_{k,\bullet}} \frac{1}{2} \|Y - XW\|_F^2 + \lambda_1 |W_{k,\bullet}| + \lambda_2 \sum_{j=2}^{n} |W_{kj} - W_{k,j-1}|$
18:     end for
19:   until $W$ does not change
20: until $X$ and $W$ do not change
```

solve the sub-optimization problem to update $W_{k,\bullet}$:

$$\min_{W_{k,\bullet}} \frac{1}{2} \|Y - XW\|_F^2 + \lambda_1 |W_{k,\bullet}| + \lambda_2 \sum_{j=2}^{n} |W_{kj} - W_{k,j-1}|, \tag{5.5}$$

which is the fused lasso problem. We used the package provided by [88] in our implementation to solve Eqn. (5.5) on line 17.

5.3.6 Relation to group Lasso

[89] proposed a group sparsity regularization method to introduce the group-structured prior knowledge for nonnegative matrix factorization. The objective function of their approach is

$$\min_{W\geq 0, H\geq 0} \frac{1}{2}\left\|Y - WH\right\|_F^2 + \alpha \left\|W\right\|_F^2 + \beta \sum_{b=1}^{B} \left\|H^{(b)}\right\|_{1,q} \tag{5.6}$$

where Y is the original data matrix and the coefficient matrix H is divided into B groups as $\{H^{(1)}, \cdots, H^{(B)}\}$ by prior knowledge. The motivation of the regularization term on $\{H^{(b)}\}$ is that samples in the same group are expected to share the same sparsity patterns in their latent factor representation. Eqn. (5.6) uses a global parameter β on group lasso to enforce the group sparsity. However, it does not fit in the problem of CNV detection since the magnitude of latent features (log ratio intensities) can be in very different scales, and it is not possible to choose a global parameter suitable for all the latent feature components. Moreover, Eqn. (5.6) does not include the fused lasso penalty, which is necessary for the CNV problem.

Now we examine the following group lasso problem similar to Eqn. (5.6),

$$\min_{X_{\bullet,k}} \frac{1}{2}\left\|Y - XW\right\|_F^2 + \gamma \sum_{l=1}^{L} p_l \left\|X_{g_l,k}\right\|_2,$$

where $p_l = \sqrt{|g_l|}$ is the weight of the lth group. Note that only $X_{\bullet,k}$ are variables in this problem and all other columns of X are fixed. For this non-overlapping group lasso problem, there exists a $\gamma_l^{(k)}$ for each group g_l such that when $\gamma \geq \gamma_l^{(k)}$, the optimal solution for $X_{g_l,k}$ is zero; when $\gamma < \gamma_l^{(k)}$, the optimal solution for $X_{g_l,k}$ is nonzero [90] and actually,

$$\gamma_l^{(k)} = \frac{\left\|v_{g_l}^{(k)}\right\|}{\sqrt{|g_l|}\left\|W_{k,\bullet}\right\|^2}$$

where $v_{g_l}^{(k)}$ is defined the same as in Eqn. (5.2). Thus, it is exactly the $\gamma_l^{(k)}$ that we used to compute $\{b_{lk}\}$ in Eqn.(5.3).

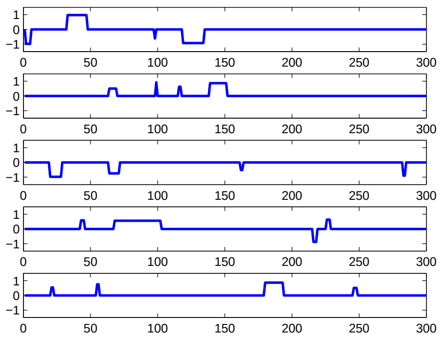
In summary, instead of using group lasso to get sparse X directly, SGS-FL first applies the group lasso setting with the parameter r to adaptively determine $\{b_{lk}\}$. Then, b_{lk}s are used to compute whether $X_{g_l,k}$ should be zero or nonzero in the optimization. Thus, r controls the sparsity of X through $\{b_{lk}\}$. Empirically, using r for adaptive sparse group selection instead of solving a group lasso problem directly for a fixed global parameter γ is more reliable and stable for CNV data analysis.

5.4 Simulation

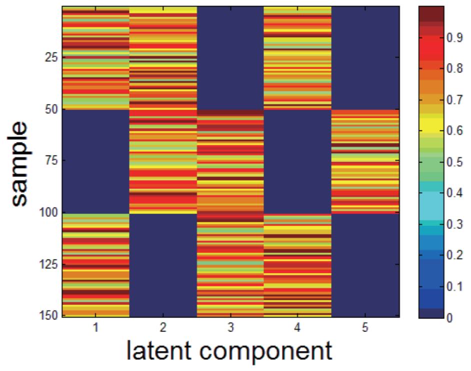
In the simulation, we compare SGS-FL with FLLat on the performance of learning the latent components and the coefficients from an artificial CNV dataset. We also tested the effect of the group sparsity parameter r and evaluated the scalability and the convergency characteristics of SGS-FL.

5.4.1 Data generation

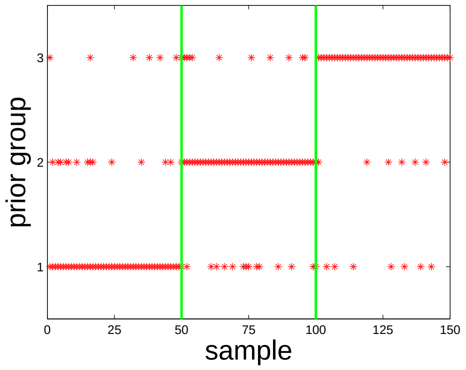
We first generated simulated latent CNV components W and coefficients X with a sparse group structure, and then constructed a CNV dataset $Y = XW + \Xi$, where Ξ are i.i.d. Gaussian noises, as illustrated in Figure 5.2. The simulated arrayCGH dataset contains 150 samples with 300 probes. There are 5 latent components, each of which contains 4 independent events of copy number gain or loss, shown in Figure 5.2(A). The 150 samples are equally divided into 3 groups with 50 samples in each group and the corresponding relation between the 3 groups and the 5 components is shown in Figure 5.2(B). The group prior is shown in Fig. 5.2(C). Errors are introduced in the prior information as a certain percentage of misplaced samples in each group. As shown in Fig. 5.2(C), each prior group contains samples from all the three true groups although majority of the samples are from only one group. Given the W and X, the dataset $Y = XW + \Xi$ is shown in Fig. 5.2(D). Y is a very noisy dataset. k-means clustering with $k = 3$ on the dataset results in error rate above 50%. The objective is to recover W and X from the noisy data Y with SGS-FL and FLLat.



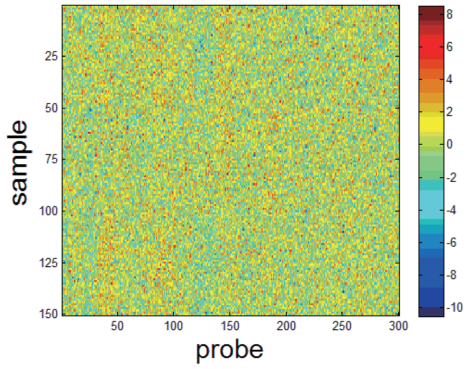
(A) Simulated W



(B) Simulated X



(C) Group information

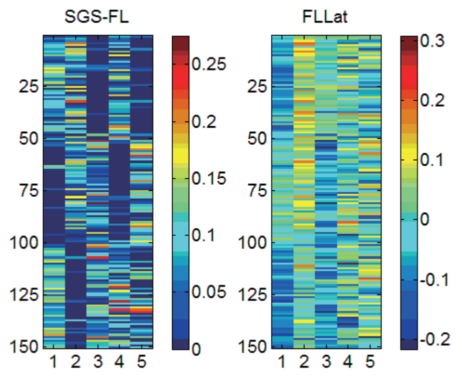


(D) Simulated Y

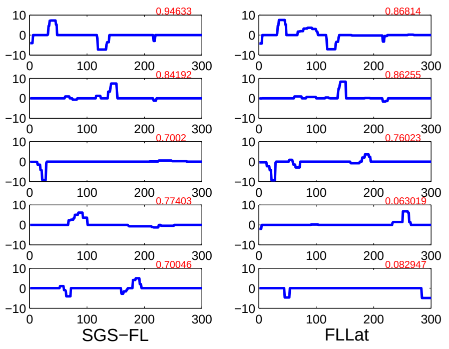
Figure 5.2: Simulation data. (A) Each latent CNV component contains four randomly generated copy number gains/losses. (B) The samples are divided into 3 groups of equal size. The coefficients are nonzero only between a group and its corresponding components. (C) Errors are introduced into the prior groups, i.e. a certain percentage of samples are misplaced into the wrong group. (D) The noisy simulated CNV dataset. There is no observable pattern although the data is constructed from the sample groups and the latent CNV components.

5.4.2 Performance of recovering W and X

Bayesian Information Criterion (BIC) [87] is used to determine the hyper-parameters λ_1 and λ_2 for applying SGS-FL and FLLat. The group sparsity parameter r is set to 0.5.



(A) Coefficients



(B) Latent components

Figure 5.3: Performance of recovering the latent components and the weight coefficients with SGS-FL and FLLat. (A) The learned coefficient matrices. (B) The learned latent components. The red numbers above each component is the Pearson correlation coefficient between the component and its corresponding original latent component.

Figure 5.3(A) shows the learned hidden components X. Clearly, the X learned by SGS-FL preserves the group structure and is more similar to the original X, compared with the X learned by FLLat. Guided by the prior group information, for each component the coefficients are learned only for the samples in the selected groups. The coefficients in the unselected groups are all zero. For example, for first latent component, only group 1 and 3 are chosen. Figure 5.3(B) shows the learned latent components. SGS-FL successfully recovered the 5 latent components with a lowest correlation 0.70 with the original component, while FLLat detected two wrong latent components that are completely different. In the FLLat method, the mistakes in the components can be matched with the wrong weights in X: column 4 and column 5 in the X in Figure 5.3(A) do not capture any group relations and thus, the corresponding components are derived to support the wrong samples. On the contrary, the X learned by SGS-FL preserves the group structures and the correct samples are used to derive the components. Note that since we introduced noise in Y and errors in the prior group, the X by SGS-FL is not perfectly sparse in the coefficients of the samples in the unselected groups.

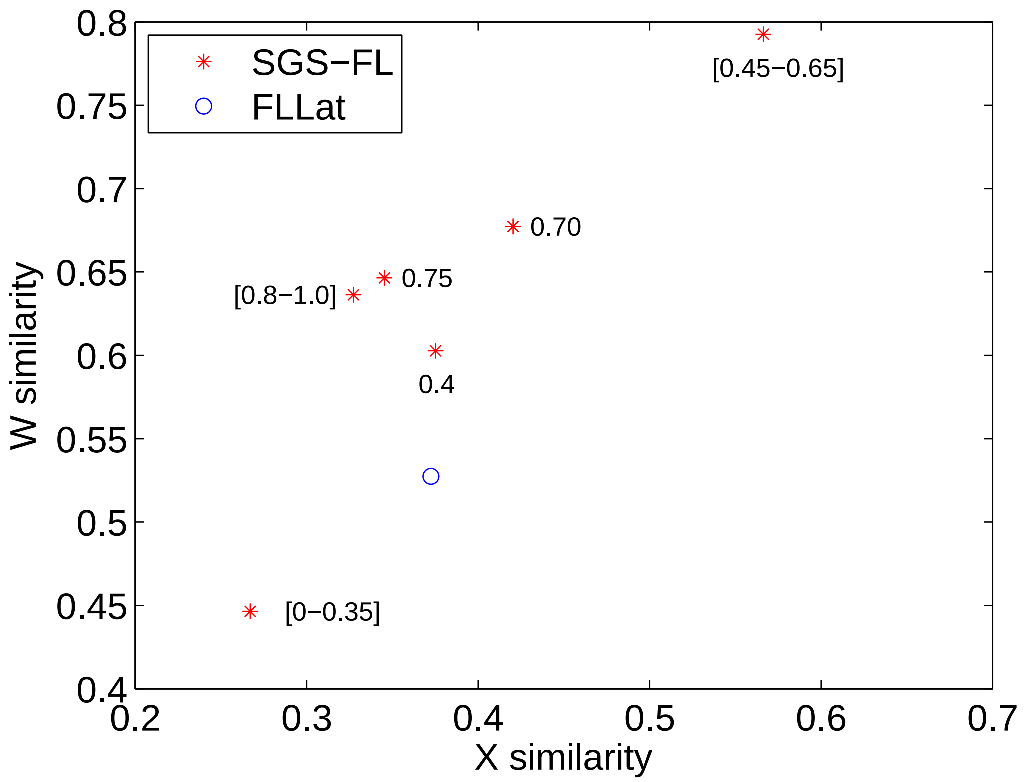
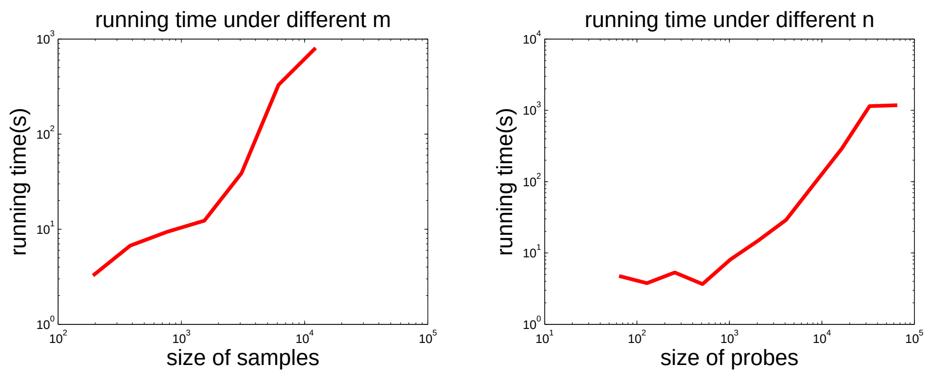


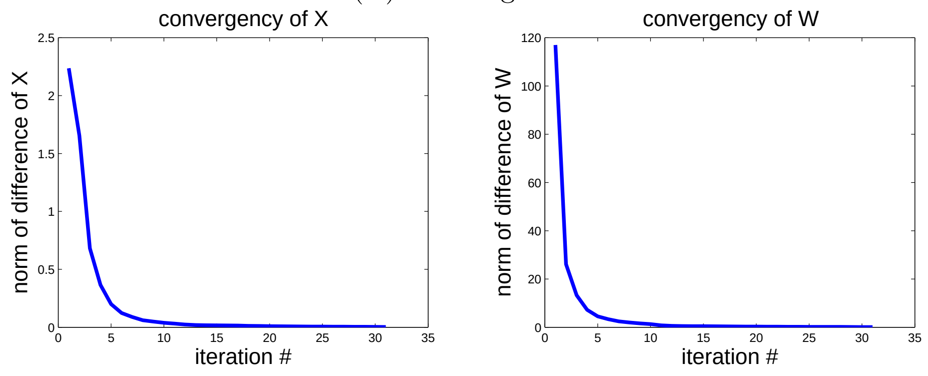
Figure 5.4: Accuracy of the learned X and W at different group sparsities. Different group sparsity parameter r is tested. The x-axis is the Pearson correlation between the learned X and the original X, and the y-axis is the correlation between the learned W and the original W. Each star represents the accuracy of X and W, labeled by the corresponding r parameter.

5.4.3 Controlling group sparsity by parameter r

Selecting appropriate group sparsity with r is important for the performance of SGS-FL. We tested SGS-FL with different $r \in [0, 1]$ with step size 0.05 and plot the accuracy of the learned X and W by calculating the Pearson correlation with the original ones in Figure 5.4. As expected that the group selection changes by steps (Eqn. 5.3) and the performance of SGS-FL only changes when group selection changes. Thus, SGS-FL performs the same in a certain range of r until reaching an increase or a decrease in the number of selected groups. It is interesting that in the range $r \in [0.45, 0.65]$, X and



(A) Running time



(B) Convergency

Figure 5.5: Running time and convergency of SGS-FL. (A) Running time under different m (# of samples) and n (# of probes). (B) Convergency of X and W.

W are most accurately recovered, and when r is too small or too large, the correlations are lower. This is consistent with our hypothesis: a small r leads to insufficient group selection for the components and a large r may lead to unnecessary group selection and thus an overly dense X that overfits Y. It is clear that in a reasonable range of sparsity, SGS-FL performs well and SGS-FL also performs better than FLLat in most of the choices of r.

5.4.4 Scalability and convergence

In real arrayCGH datasets, the number of probes can be as many as several millions. In Figure 5.5, we analyze the running time and the convergency of SGS-FL on simulated datasets of different sample sizes and different numbers of probes. In the left plot of

Figure 5.5(A), we fixed the number of probes to be 300 and vary the sample size; in the right plot, we fixed the sample size to be 150 and vary the number of probes. In both cases, SGS-FL scales linearly with the log of the sizes. In Figure 5.5(B), SGS-FL clearly converges within tens of iterations. The results suggest a good scalability to large datasets by SGS-FL.

5.5 Experiments on Breast Cancer Data

To directly compare SGS-FL with FLLat, we followed the experiment setup in [87]. SGS-FL and FLLat were applied to chromosome 8 and 17 of a breast cancer arrayCGH data from [91] to identify CNV regions for cancer relevance.

5.5.1 Breast cancer data

The breast cancer data contains profiles of 44 predominantly advanced primary breast tumors with 241 mapped human genes from chromosome 8 and 382 mapped human genes from chromosome 17. Among the 44 profiles, 5 are in tumor grade 1, 21 in grade 2 and 17 in grade 3. This prior clinical information was used in our model to define three groups of samples. There is one additional sample missing the clinical information of tumor grade which was also included in the study. Note that SGS-FL allows additional samples that are not assigned to any group. The number of latent feature K was chosen as the number of principle components that explain at least 80% variation of the data. The hyper-parameters λ_1 and λ_2 were chosen by Bayesian Information Criterion (BIC) suggested by [87]. The r parameter for SGS-FL was set to 0.5 to learn a coefficient matrix with moderate sparsity.

5.5.2 Analysis of the coefficient matrices

We compare the coefficient matrices learned from FLLat and SGS-FL for chromosome 17 in Figure 5.6. The samples are ordered by tumor grade and the three groups are separated by red horizontal lines. In the coefficient matrix learned by FLLat, there is hardly any group structure of samples and the relation between the samples and the latent features seems arbitrary. In other words, there is no subset of samples with similar tumor grade by which a latent feature is shared. The coefficients learned by SGS-FL

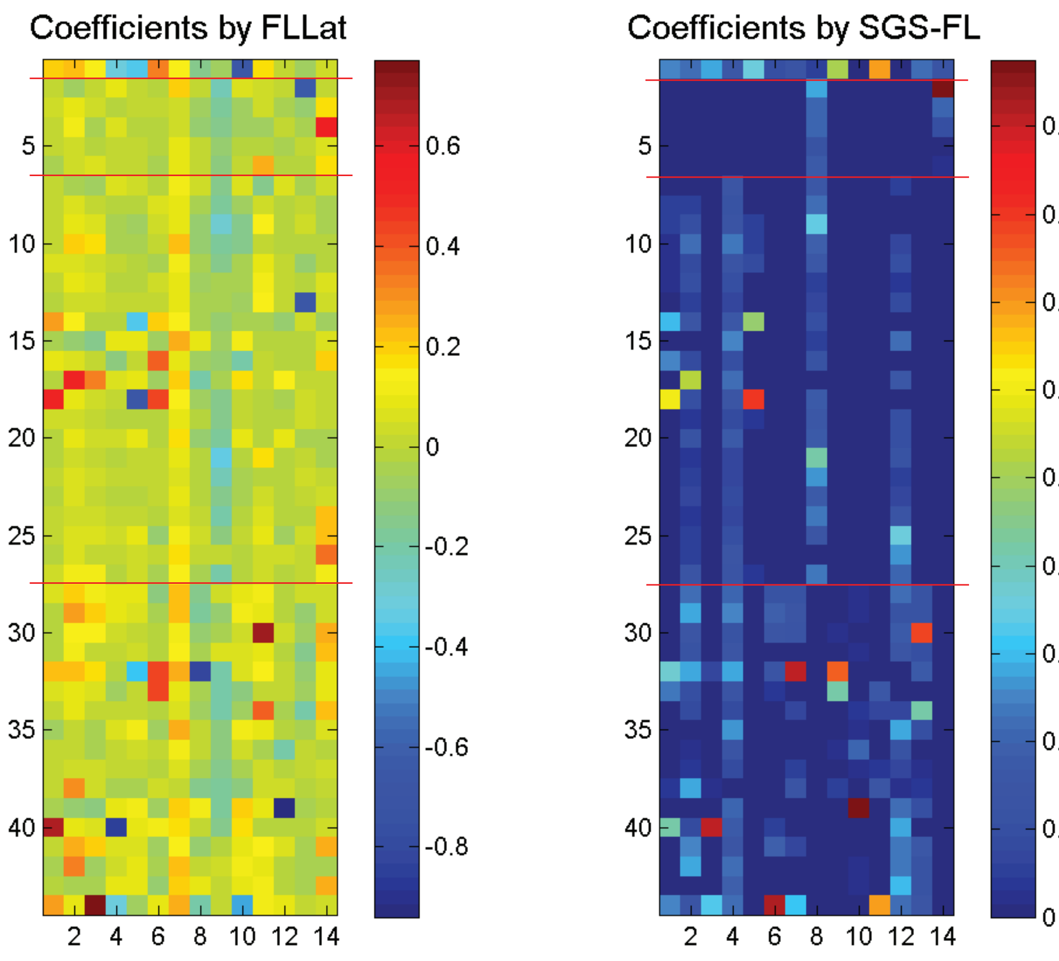


Figure 5.6: The coefficient matrices learned from the breast cancer data by FLLat and SGS-FL. The sample profile missing the tumor grade information is put in the first row and the other samples are ordered by tumor grade 1-3 from top to bottom. The groups are separated by red horizontal lines. The K columns of X are sorted in descending order by the magnitude of the corresponding latent features (i.e. $\|W_{i,\bullet}\|_2$).

show clear group structures. When the coefficients are all zeros in one group, it implies that the CNVs identified from the corresponding latent feature are not associated with that group. For example, the first two latent features are only shared by the groups of tumor grade 2 and 3; the third latent feature is only shared by the group of tumor grade 3 and the last latent feature is only shared by the group of tumor grade 1. The

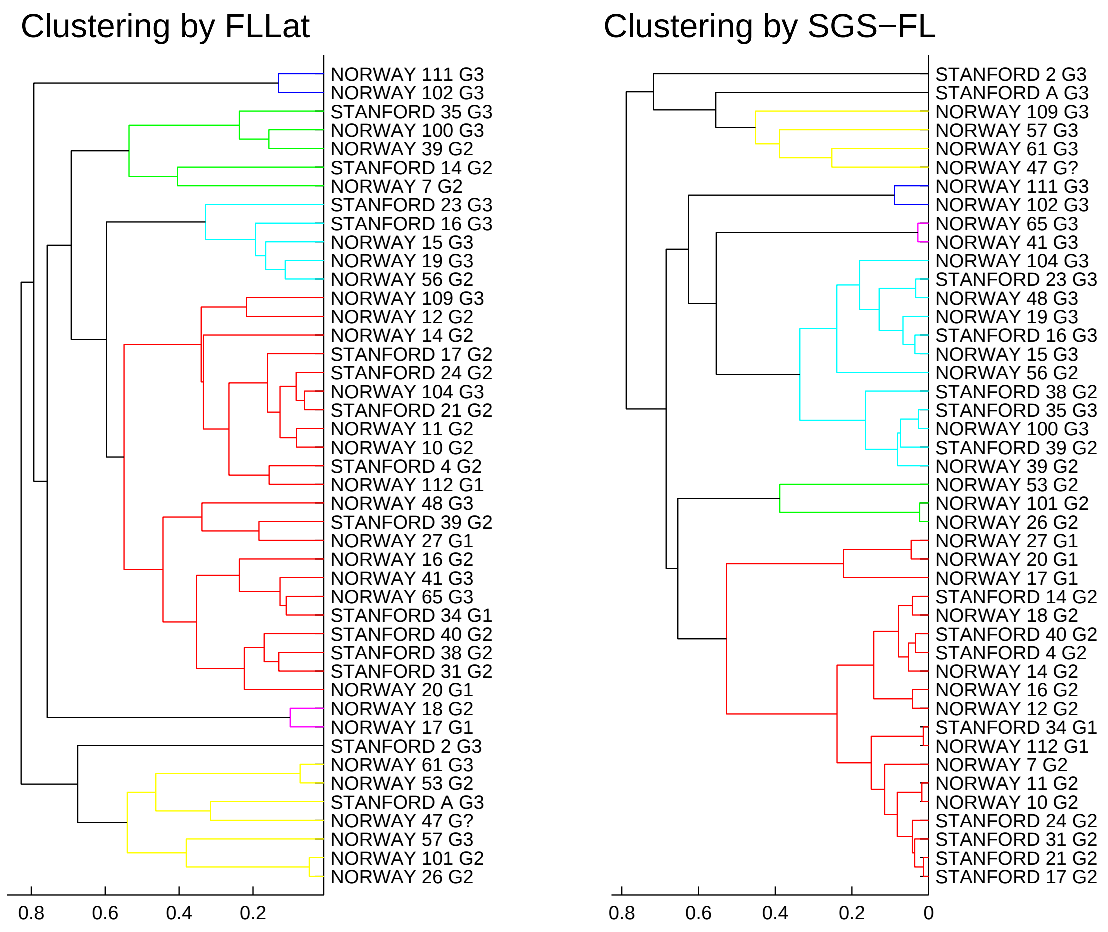


Figure 5.7: Hierarchical clustering results of breast cancer samples. The samples are labeled by their tumor grade, G1, G2 or G3.

submatrix of the last group is denser than those of the first and the second groups, which implies that the samples with tumor grade 3 are sharing more CNVs than the samples in the other two groups.

We also performed hierarchical clustering on the two coefficient matrices. Cosine similarity was used as the similarity metric in the clustering to obtain similar clustering results reported in [87]. The clustering results are shown in Figure 5.7. Since the tumor grade information is incorporated in SGS-FL, the generated hierarchical structures are more biologically meaningful. For example, there are three large clusters: the first cluster contains samples with tumor grade 1 and 2, the second cluster contains samples with tumor grade 2 and 3, and the third cluster contains samples with tumor grade 3 except the additional sample missing the tumor grade information. Since tumor grade 2 is an intermediate state between tumor grade 1 and 3, it is reasonable to assume that some samples with tumor grade 2 are more similar to samples with tumor grade 1 and some are more similar to samples with tumor grade 3. The results can be easily explained by the coefficient matrix in Figure 5.6. The components can only be shared by samples in group 1 and group 2, or by samples in group 2 and group 3, and never shared by samples in group 1 and group 3. This result strongly support the hypothesis that CNVs correlate with the tumor grade. The clustering result generated by FLLat in Figure 5.7 and [87] also showed there are three distinct groups of samples. However, it is not clear why these samples were clustered together since their tumor grades are different.

5.5.3 Sample classification

To check whether the coefficient matrix X is also consistent with other clinical information, we also designed a binary classification problem of separating samples into two groups with another clinical variable 'Tumor size' (T1&T2 vs. T3&T4). Tumor grades were used as prior group information by SGS-FL and the 'Tumor size' variable is the target variable for classification. We run a leave-one-out cross-validation with k-nearest neighbor (KNN) classifier on the coefficient matrices learned by PCA, FLLat and SGS-FL from chromosome 8 and 17. The number of latent features are fixed to be the number of principle components that explain 80% variance for all the three methods. The classification accuracies by the three methods with different KNN parameters are reported in Table 5.1. It is not surprising that PCA achieved the best performance since PCA preserves the most variance of the data without constraints on obtaining interpretable coefficients. Nevertheless, in the table the result by SGS-FL is comparable to PCA and much better than the result by FLLat. It suggests that by using relevant prior information, SGS-FL can obtain both interpretable CNV components and informative coefficients for classification. It is worth noting that if only the tumor grade information is used by KNN in the leave-one-out cross-validation, the accuracy is 0.727 for any choice of k for the KNN classifier. This result further implies that the better classification performance of SGS-FL is not solely due to the relevant prior information

		k=1	k=3	k=5	k=7
Chromosome 8	PCA	0.727	**0.795**	**0.795**	0.682
	FLLat	**0.659**	**0.659**	0.614	0.636
	SGS-FL	0.705	**0.795**	0.750	0.773
Chromosome 17	PCA	0.750	0.795	**0.818**	0.750
	FLLat	0.591	0.682	0.659	**0.750**
	SGS-FL	**0.773**	0.750	0.705	0.750
Tumor grade		0.727	0.727	0.727	0.727

Table 5.1: Classification accuracies in leave-one-out cross-validation with best results for each method bold.

in tumor grade.

5.5.4 Analysis of CNV components

We next compared the latent features learned by FLLat and SGS-FL. We ranked the identified probes by the sum of their magnitude in all latent features (i.e. $\sum_k |W_{kj}|$). The probes without gene names were excluded in this analysis. We took the top-50 genes ranked by FLLat and SGS-FL, and plot their ranks in Figure 5.8. FLLat and SGS-FL have consensus on the ranks of many of the genes. We focus on the genes which have a difference larger than 3 in the ranks by the two methods. There are several interesting examples. NGFR was demonstrated as a marker to identify myoepithelial cells in preinvasive lesions and myoepithelial differentiation in breast carcinomas [92]. SPOP can mediate the Breast cancer metastasis suppressor 1 (BRMS1) and is important for breast cancer progression [93]. PIP5K2B (PIP4K2B) is a known amplified gene in breast cancer [94]. DLX4 (BP1) negatively regulates BRCA1 in sporadic breast cancer [95]. NR1D1 is a survival factor for breast cancer [96]. ITGB4 is a prognostic marker for breast cancer [97]. All the genes ranked better by SGS-FL seem to be relevant to breast cancer. However, for the genes ranked higher by FLLat such as ZNF207, PCTP and SCYA3L1, there is no literature suggesting associations between the genes and breast cancer. Possibly, these genes might be involved in some frequent CNVs instead of CNVs specific to breast cancer.

Finally, we also compared the ranking of the known cancer genes in Cancer Gene

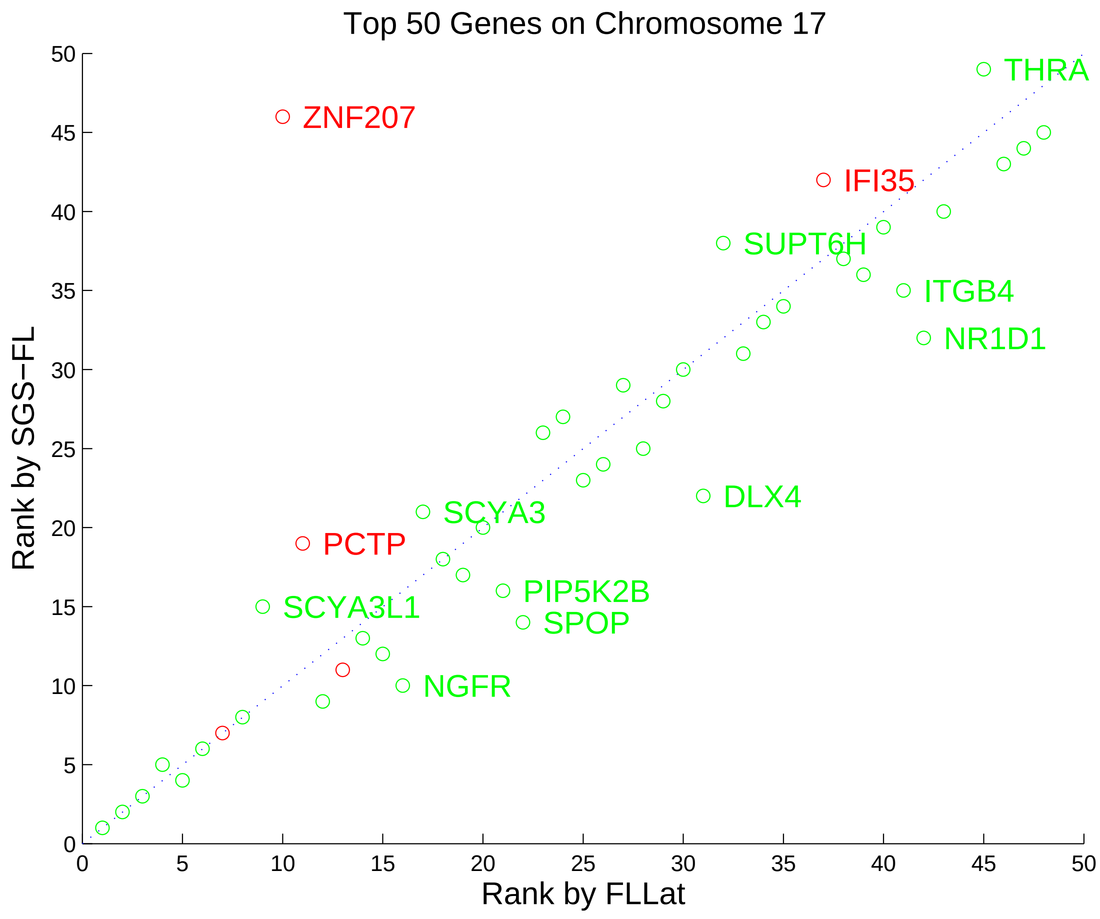


Figure 5.8: Top ranked 50 genes by FLLat and SGS-FL on chromosome 17. Green denotes 'gain' status and red denotes 'loss' status. Genes with most different ranks by FLLat and SGS-FL are label by their gene symbols.

Census[1] on chromosome 8 and 17 in Figure 5.9. Overall, most of the known cancer genes were ranked better by SGS-FL. The result implies that the identified CNVs by SGS-FL are more likely to be associated with breast cancer.

[1] `http://www.sanger.ac.uk/genetics/CGP/Census/`

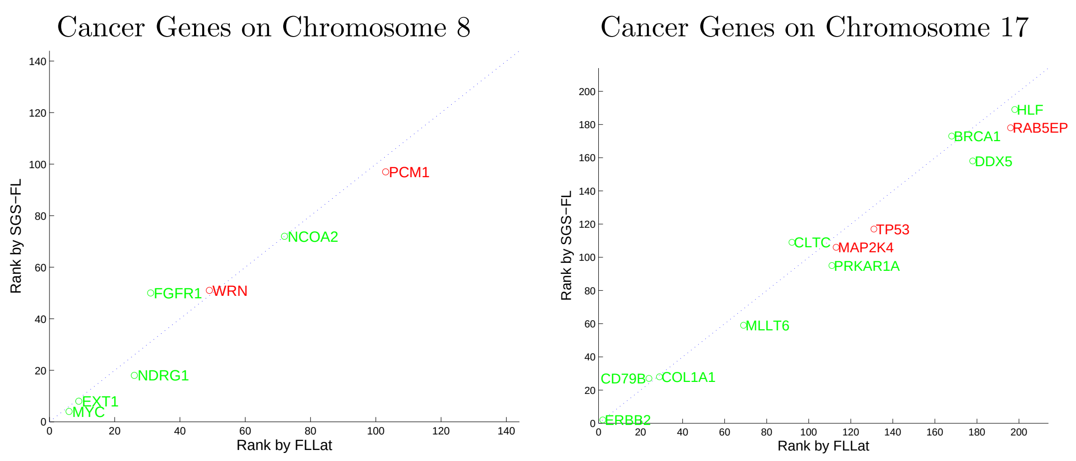


Figure 5.9: Ranking of known cancer genes on chromosome 8 and 17. Green denotes 'gain' status and red denotes 'loss' status. On average, SGS-FL ranked the cancer genes on chromosome 8 and 17 better than FLLat.

5.6 Experiments on Bladder Cancer Data

We also applied SGS-FL and FLLat to test a bladder cancer arrayCGH data from [55] to identify CNVs relevant to bladder cancer. This dataset contains 38 urothelial carcinomas with whole-genome tiling resolution array-CGH and high density expression profiling. We still used tumor grade as the prior information to separate samples into 3 groups {G1,G2,G3} and set $r = 0.5$ for SGS-FL. The parameters k, λ_1 and λ_2 were selected in the same way as in the previous experiment. [55] reported genomic amplifications of 47 genes at regions 2p25, 6p22 and 8q22 in "Additional file 4", so we focused our study on these chromosomes. There are 1938 probes on chromosome 2; 1801 probes on chromosome 6; and 1091 probes on chromosome 8. 33 of the 47 genes are annotated in the dataset. Both FLLat and SGS-FL identified the 33 known amplified genes and ranked them in the top 100 probes. We also compared the methods with the naïve approach which ranks the genes simply based on the sum of their magnitude in the original data (i.e. $\sum_i |Y_{ij}|$). The 33 genes and their corresponding ranks are listed in Table 5.2. Among the 33 genes, SGS-FL has higher or equal ranks on 21 of them. Compared with FLLat, SGS-FL ranked 16 genes better and 6 genes worse. The result suggests that the prior information in tumor grade helps rank the cancer relevant CNVs higher. Compared with the naïve method, SGS-FL ranked 25 genes better and 7 genes

Chromosome	Gene	Naïve	FLLat	SGS-FL
2	CPSF3	3	**1**	**1**
	ADAM17	3	**1**	**1**
	YWHAQ	3	**1**	**1**
	TAF1B	8	**4**	**4**
	UNQ5830	13	**5**	**5**
	KLF11	**1**	5	5
	RRM2	**1**	8	8
6	CAP2	53	59	**52**
	FAM8A1	**28**	61	53
	NUP153	**28**	61	53
	KIF13A	**28**	62	54
	NHLRC1	79	66	**58**
	AOF1	117	67	**59**
	DEK	117	67	**59**
	IBRDC2	117	67	**59**
	ID4	66	31	**30**
	OACT1	23	**13**	14
	E2F3	10	**1**	**1**
	CDKAL1	**3**	**3**	**3**
	SOX4	20	**16**	**16**
	PRL	**15**	22	22
8	COX6C	65	49	**45**
	POLR2K	69	51	**47**
	SPAG1	69	51	**47**
	RNF19	69	51	**47**
	MGC39715	76	53	**49**
	NCALD	**6**	7	37
	RRM2B	52	**10**	39
	ODF1	138	**63**	71
	KLF10	185	**64**	76
	FLJ45248	162	**65**	77
	ATP6V1C1	46	58	**35**
	BAALC	68	60	**41**

Table 5.2: Ranking of the 33 known amplified genes in the bladder cancer data. The best rank of each gene is bold.

worse. The result suggests that the learned latent features is more reliable than the original data for identifying cancer relevant CNVs.

5.7 Conclusions

In general, discovering CNVs across multiple samples is more accurate than single sample analysis. To analyze multiple samples of probe series, it is important to consider both the similarity and the heterogeneity among the samples. Existing methods such as FLLat ignore the fact that patient samples with different phenotypes show different frequencies and patterns of CNVs in their genotyping. These methods tend to miss the CNVs specific to subsets of samples. To the best of our knowledge, SGS-FL is the first model that considers the prior information on sample groups in CNV identification. SGS-FL constructs a latent feature model to identify CNVs and learn the sample groups sharing the CNVs simultaneously by integrating fused lasso to smooth CNV patterns and adaptive sparse group selection to identify the group specificity of the CNVs. The simulations and experiments on real cancer arrayCGH datasets suggest that with the relevant sample group information, SGS-FL can more accurately identify cancer relevant CNV regions and a more informative representation of CNV data as coefficients on the CNV components.

Chapter 6

Conclusion and Discussion

6.1 Conclusion

Finding associations between CNVs and human diseases has been a hot topic in Bioinformatics since the role of CNVs in human disease was first revealed. The development of new techniques such as arrayCGH made it possible to detect genomic copy number variations at a high resolution level. Numerous arrayCGH datasets and methods were proposed aiming to identify CNVs that are associated with diseases. However, there are still several challenges as described in section 1.1, which make general machine learning methods or statistical approaches fail on these datasets. Several models have been proposed in this thesis to address these challenges and improve the performance of sample classification and CNV identification on arrayCGH datasets.

First, we proposed *HyperPrior* to incorporate biological prior knowledge into predictive models for data integration. We applied *HyperPrior* to test two arrayCGH datasets and two gene expression datasets for both cancer classification and biomarker identification. On all the datasets, *HyperPrior* achieved competitive classification performance, compared with SVMs and the other baselines utilizing the same prior knowledge. *HyperPrior* also identified several discriminative regions on chromosomes and discriminative subnetworks in the PPI, both of which contain cancerrelated genomic elements. Our results suggest that *HyperPrior* is promising in utilizing biological prior knowledge to achieve better classification performance and more biologically interpretable findings in gene expression and arrayCGH data.

Then, we proposed an alignment based framework for integrated sample classification and cross-dataset CNV analysis. The probe alignment kernel and the multiple probe alignment algorithm were experimented to integrate three bladder cancer datasets as well as artificial datasets. In the experiments, by integrating arrayCGH samples from multiple datasets, the probe alignment kernel used with support vector machines significantly improved patient sample classification accuracy over other baseline kernels. The experiments also demonstrated that the multiple probe alignment algorithm can find common DNA aberrations that cannot be identified with the standard interpolation method. Furthermore, the multiple probe alignment algorithm also identified many known bladder cancer DNA aberrations containing four known bladder cancer genes, three of which cannot be detected by interpolation.

We also proposed *GLNP* as a general algorithm to learn a low-rank graph and perform more accurate label prorogation for sample classification. Large scale simulations and experiments on UCI datasets and high-dimensional datasets (arrayCGH and gene expression datasets) showed that label propagation based on global linear neighborhoods captures the global cluster structures better and achieved more accurate classification results. The experiment results suggested that *GLNP* is a promising semi-supervised learning algorithm for sample classification problems on arrayCGH datasets.

Finally, we proposed *SGS-FL* to integrate phenotype information to identify disease-associated CNVs from arrayCGH datasets. *SGS-FL* is a latent feature model designed to identify group(phenotype)-specific CNVs instead of common CNVs. Simulation results show that *SGS-FL* can more accurately identify the latent CNV components when there is a reliable underlying group structure in the samples. In the experiments on arrayCGH breast cancer and bladder cancer datasets, *SGS-FL* detected CNV regions that are more relevant to cancer, and provided latent feature weights that can be used for better sample classification.

In summary, all models proposed in this thesis showed promising results in both simulations and experiments on real arrayCGH datasets. Algorithms implemented for these models also provide computational tools for oncologists to discover more disease-relevant CNV candidates in various studies of human disease.

6.2 Future Work

The models proposed in this thesis can be improved in several ways. We will discuss each model separately in the following subsections.

6.2.1 *HyperPrior* algorithm

For the *HyperPrior* algorithm, the incidence matrix H has to be nonnegative. Thus for each feature with both positive and negative values, *HyperPrior* has to use two hyperedges to encode that feature. This might cause scalability problems on some high-resolution arrayCGH datasets or next-generation sequencing data. However, defining hyperedges based on the sign of feature values is just one way to model the data with a hypergraph. Actually, the relation between the original features and the hyperedges can be more general. As we have already discussed in Chapter 2, *HyperPrior* can be explained as a relaxed wrapper-feature-selection method. Thus we can define each hyperedge based on multiple features in the original data as long as it is meaningful. For example, one possible way is to use pathways to define hyperedges and the relations between the pathways are used as prior knowledge. We believe if we properly define biologically relevant hyperedges, much less number of hyperedges are necessary to encode high-dimensional biomedical data, and *HyperPrior* will be scalable to next-generation sequencing data.

In the thesis, we have shown *HyperPrior* worked well with the prior knowledge we defined. It will be interesting to also try other types of prior knowledge. For example, we can use the gene expression value to define the prior knowledge on the corresponding probes in arrayCGH data.

6.2.2 Probe alignment kernel

For the probe alignment kernel, one limitation is that the defined scoring function is not guaranteed to output a positive semi-definite matrix. Thus a small positive value has to be added to the diagonal to make it a valid kernel. However, this small positive value is not known until the alignment score from all pairs of samples are computed and it might not be efficient for large scoring matrices. To solve this problem, we may try the methods proposed in [98, 99], which propose to train SVM with indefinite kernels.

Another possible extension is to distinguish regions with different probe density (gene-rich vs. non-coding) with different normalization for further improvement of classification or common CNV detection. The scoring function can also be different in gene-rich regions and non-coding regions since the alignment in gene-rich regions is expected to be more biologically significant. Although it will introduce additional complexity to the model, the alignment results might be more biologically relevant if the alignment kernel is designed in a sophisticated way.

Last but not least, segmentation methods such as [100] are also widely used to detect the actual CNV intervals in each probe series. These segmentation methods output CNV intervals of very different lengths for each sample. Integrating these segmentation results is more challenging than integrating probe series since both the number of intervals and the length of intervals can vary very much among all samples. However, we believe the idea of alignment kernel can still be used to develop a new alignment approach to compare CNV intervals. It will be promising to combine segmentation approaches with alignment to get more interesting results.

6.2.3 *GLNP* algorithm

In *GLNP* algorithm, the low-rank graph is learned from data in an unsupervised manner and then label prorogation is performed for semi-supervised learning. However, it can be easily extended to perform semi-supervised learning directly as in [101]. Suppose there are m samples and m_l of them are used for training. We need to predict the labels for the remaining $m_u = m - m_l$ samples. Let Y_l be the labels for training set, Y_u be the labels to predict and $Y = \begin{bmatrix} Y_l \\ Y_u \end{bmatrix}$. Then, the low-rank graph and the labels on the test set can be learned from

$$\min \mathcal{Q}(F, Y_u) = \frac{1-\alpha}{2} \left\| X - FF^T X \right\|_F^2 + \frac{\alpha}{2} \left\| Y - FF^T Y \right\|_F^2$$

subject to $F_{ij} \geq 0$ where α is a parameter between 0 and 1. It can be solved similarly with multiplicative update rule.

Another possible better model to learn a low-rank graph is

$$\min \mathcal{Q}(F) = \left\| X - (FF^T - I)X \right\|_F^2$$

subject to $F_{ij} \geq 0$ and $\sum_j F_{ij}^2 = 1$. The low-rank graph learned from this model should be more suitable for label prorogation. However, this optimization problem is harder since it requires each row of F to be normalized.

Last, the multiplicative update rule requires the data to be nonnegative. If we want to use *GLNP* to handle data with both positive and negative values, we can use the additive update rule with nonnegative constraints. However, more efficient optimization techniques based on gradient descent may be required to get better solutions in real problems.

6.2.4 *SGS-FL* model

Identifying CNVs from multiple samples is expected to be more accurate than from single sample since it helps to remove false positives caused by the noise in the data. However, if only CNVs from all samples are reported, we may only get common CNVs while still missing disease-specific CNVs. By considering both the similarity and the heterogeneity among the samples, *SGS-FL* identified cancer relevant CNV regions more accurately in experiments in the thesis. However, tumor grade of patients is just one type of prior knowledge we can use. The next step is to investigate different types of clinical information to find what prior knowledge is more helpful in identifying disease-specific CNVs.

In fused lasso model, λ_1 and λ_2 control the number of identified CNVs. The same λ_1 and λ_2 were used in *SGS-FL* for all latent components. Since it is more reasonable to assume both the number of CNVs and the length of CNVs can vary for each latent component, it is important to derive a better parameter tuning method to generate informative CNVs in all latent components.

References

[1] E. E. Eichler. Copy number variation and human disease. *Nature Education*, 1(3), 2008.

[2] L Feuk, AR Carson, and SW Scherer. Structural variation in the human genome. *Nature Reviews Genetics*, 7(2):85–97, FEB 2006.

[3] Richard Redon, Shumpei Ishikawa, Karen R. Fitch, Lars Feuk, George H. Perry, T. Daniel Andrews, Heike Fiegler, Michael H. Shapero, Andrew R. Carson, Wenwei Chen, Eun Kyung Cho, Stephanie Dallaire, Jennifer L. Freeman, Juan R. Gonzalez, Monica Gratacos, Jing Huang, Dimitrios Kalaitzopoulos, Daisuke Komura, Jeffrey R. MacDonald, Christian R. Marshall, Rui Mei, Lyndal Montgomery, Kunihiro Nishimura, Kohji Okamura, Fan Shen, Martin J. Somerville, Joelle Tchinda, Armand Valsesia, Cara Woodwark, Fengtang Yang, Junjun Zhang, Tatiana Zerjal, Jane Zhang, Lluis Armengol, Donald F. Conrad, Xavier Estivill, Chris Tyler-Smith, Nigel P. Carter, Hiroyuki Aburatani, Charles Lee, Keith W. Jones, Stephen W. Scherer, and Matthew E. Hurles. Global variation in copy number in the human genome. *Nature*, 444(7118):444–454, NOV 23 2006.

[4] Adam Shlien and David Malkin. Copy number variations and cancer. *Genome Med*, 1(6):62, 2009.

[5] Feng Zhang, Wenli Gu, Matthew E. Hurles, and James R. Lupski. Copy Number Variation in Human Health, Disease, and Evolution. *Annual Review of Genomics and Human Genetics*, 10:451–481, 2009.

[6] Nigel P. Carter. Methods and strategies for analyzing copy number variation using DNA microarrays. *Nature Genetics*, 39(7):S16–S21, JUL 2007.

[7] Nuala H. Sykes, Claudio Toma, Natalie Wilson, Emanuela V. Volpi, Ines Sousa, Alistair T. Pagnamenta, Raffaella Tancredi, Agatino Battaglia, Elena Maestrini, Anthony J. Bailey, Anthony P. Monaco, and IMGSAC. Copy number variation and association analysis of SHANK3 as a candidate gene for autism in the IMGSAC collection. *European Journal of Human Genetics*, 17(10):1347–1353, OCT 2009.

[8] TaeHyun Hwang, Hugues Sicotte, Ze Tian, Baolin Wu, Jean-Pierre Kocher, Dennis A. Wigle, Vipin Kumar, and Rui Kuang. Robust and efficient identification of biomarkers by classifying features on graphs. *Bioinformatics*, 24(18):2023–2029, SEP 15 2008.

[9] LJ van't Veer, HY Dai, MJ van de Vijver, YDD He, AAM Hart, M Mao, HL Peterse, K van der Kooy, MJ Marton, AT Witteveen, GJ Schreiber, RM Kerkhoven, C Roberts, PS Linsley, R Bernards, and SH Friend. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415(6871):530–536, JAN 31 2002.

[10] YX Wang, JGM Klijn, Y Zhang, A Sieuwerts, MP Look, F Yang, D Talantov, M Timmermans, ME Meijer-van Gelder, J Yu, T Jatkoe, EMJJ Berns, D Atkins, and JA Foekens. Gene-expression pro-files to predict distant metastasis of lymph-node-negative primary breast cancer. *Lancet*, 365(9460):671–679, FEB 19 2005.

[11] Ze Tian, TaeHyun Hwang, and Rui Kuang. A hypergraph-based learning algorithm for classifying gene expression and arrayCGH data with prior knowledge. *Bioinformatics*, 25(21):2831–2838, NOV 1 2009.

[12] Ze Tian, TaeHyun Hwang, and Rui Kuang. A hypergraph-based learning algorithm for classifying arraycgh data with spatial prior. In *Genomic Signal Processing and Statistics, 2009. GENSIPS 2009. IEEE International Workshop on*, pages 1 –4, May 2009.

[13] Ze Tian and Rui Kuang. Integrative classification and analysis of multiple arrayCGH datasets with probe alignment. *Bioinformatics*, 26(18):2313–2320, SEP 2010.

[14] Ze Tian and Rui Kuang. Global linear neighborhoods for efficient label propagation. In *Proceedings of the 12th SIAM International Conference on Data Mining*, page 863, 2012.

[15] Charles L. Sawyers. The cancer biomarker problem. *Nature*, 452(7187):548–552, APR 3 2008.

[16] GV Glinsky, AB Glinskii, AJ Stephenson, RM Hoffman, and WL Gerald. Gene expression profiling predicts clinical outcome of prostate cancer. *Journal of Clinical Investigation*, 113(6):913–923, MAR 2004.

[17] J. C. M. Pole, C. Courtay-Cahen, M. J. Garcia, K. A. Blood, S. L. Cooke, A. E. Alsop, D. M. L. Tse, C. Caldas, and P. A. W. Edwards. High-resolution analysis of chromosome rearrangements on 8p in breast, colon and pancreatic cancer reveals a complex pattern of loss, gain and translocation. *Oncogene*, 25(41):5693–5706, SEP 14 2006.

[18] G Tonon, KK Wong, G Maulik, C Brennan, B Feng, YY Zhang, DB Khatry, A Protopopov, MJ You, AJ Aguirre, ES Martin, ZH Yang, HB Ji, L Chin, and RA DePinho. High-resolution genomic profiles of human lung cancer. *Proceedings of the National Academy of Sciences of the United States of America*, 102(27):9625–9630, JUL 5 2005.

[19] Alain Dupuy and Richard M. Simon. Critical review of published microarray studies for cancer outcome and guidelines on statistical analysis and reporting. *Journal of the National Cancer Institute*, 99(2):147–157, JAN 17 2007.

[20] Han-Yu Chuang, Eunjung Lee, Yu-Tsueng Liu, Doheon Lee, and Trey Ideker. Network-based classification of breast cancer metastasis. *Molecular Systems Biology*, 3, OCT 2007.

[21] Ramon Aragues, Chris Sander, and Baldo Oliva. Predicting cancer involvement of genes from heterogeneous data. *BMC Bioinformatics*, 9, MAR 27 2008.

[22] Franck Rapaport, Andrei Zinovyev, Marie Dutreix, Emmanuel Barillot, and Jean-Philippe Vert. Classification of microarray data using gene networks. *BMC Bioinformatics*, 8, FEB 1 2007.

[23] Franck Rapaport, Emmanuel Barillot, and Jean-Philippe Vert. Classification of arrayCGH data using fused SVM. *Bioinformatics*, 24(13):I375–I382, JUL 1 2008. 16th ISMB Conference on Intelligent Systems for Molecular Biology, Toronto, CANADA, JUL 19-23, 2008.

[24] Sameer Agarwal, Kristin Branson, and Serge Belongie. Higher order learning with graphs. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 17–24, New York, NY, USA, 2006. ACM.

[25] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1601–1608. MIT Press, Cambridge, MA, 2007.

[26] Z Barutcuoglu, RE Schapire, and OG Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, APR 1 2006.

[27] K Tsuda, HJ Shin, and B Scholkopf. Fast protein classification with multiple networks. *Bioinformatics*, 21(2):59–65, SEP 2005. Joint Meeting of the 4th European Conference on Computational Biology/6th Meeting of the Spanish-Bioinformatics-Network, Madrid, SPAIN, SEP 28-OCT 01, 2005.

[28] Jiexun Li, Xin Li, Hua Su, Hsinchun Chen, and David W. Galbraith. A framework of integrating gene relations from heterogeneous data sources: an experiment on Arabidopsis thaliana. *Bioinformatics*, 22(16):2037–2043, AUG 15 2006.

[29] Zheng Zhao, Jiangxin Wang, Huan Liu, Jieping Ye, and Yung Chang. Identifying biologically relevant genes via multiple heterogeneous data sources. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 839–847, New York, NY, USA, 2008. ACM.

[30] C Jones, E Ford, C Gillett, K Ryder, S Merrett, JS Reis, LG Fulford, A Hanby, and SR Lakhani. Molecular cytogenetic identification of subgroups of grade III invasive ductal breast carcinomas with different clinical outcomes. *Clinical Cancer Research*, 10(18, Part 1):5988–5997, SEP 15 2004.

[31] S-F Chin, Y. Wang, N. P. Thorne, A. E. Teschendorff, S. E. Pinder, M. Vias, A. Naderi, I. Roberts, N. L. Barbosa-Morais, M. J. Garcia, N. G. Iyer, T. Kranjac, J. F. R. Robertson, S. Aparicio, S. Tavare, I. Ellis, J. D. Brenton, and C. Caldas. Using array-comparative genomic hybridization to define molecular portraits of primary breast cancers. *Oncogene*, 26(13):1959–1970, MAR 22 2007.

[32] Caiyan Li and Hongzhe Li. Network-constrained regularization and variable selection for analysis of genomic data. *Bioinformatics*, 24(9):1175–1182, MAY 1 2008.

[33] Ted Sandler, John Blitzer, Partha Pratim Talukdar, and Lyle H. Ungar. Regularized learning with networks of features. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1401–1408. 2009.

[34] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003.

[35] James C. Bezdek and Richard J. Hathaway. Convergence of alternating optimization. *Neural, Parallel Sci. Comput.*, 11:351–368, December 2003.

[36] TaeHyun Hwang, Ze Tian, Rui Kuang, and J.-P. Kocher. Learning on weighted hypergraphs to integrate protein interactions and gene expressions for cancer outcome prediction. In *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, pages 293 –302, Dec. 2008.

[37] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006.

[38] M Gribskov and NL Robinson. Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Computers & Chemistry*, 20(1):25–33, MAR 1996.

[39] E Blaveri, JL Brewer, R Roydasgupta, J Fridlyand, S DeVries, T Koppie, S Pejavar, K Mehta, P Carroll, JP Simko, and FM Waldman. Bladder cancer stage and outcome by array-based comparative genomic hybridization. *Clinical Cancer Research*, 11(19, Part 1):7012–7022, OCT 1 2005.

[40] MD Onken, JP Ehlers, LA Worley, J Makita, Y Yokota, and JW Harbour. Functional gene expression analysis uncovers phenotypic switch in aggressive uveal melanomas. *Cancer Research*, 66(9):4602–4609, MAY 1 2006.

[41] Marcia R. Saban, Helen L. Hellmich, Cindy Simpson, Carole A. Davis, Mark L. Lang, Michael A. Ihnat, Michael A. O'Donnell, Xue-Ru Wu, and Ricardo Saban. Repeated BCG treatment of mouse bladder selectively stimulates small GTPases and HLA antigens and inhibits single-spanning uroplakins. *BMC Cancer*, 7, NOV 2 2007.

[42] Panagiotis A. Konstantinopoulos, Michalis V. Karamouzis, and Athanasios G. Papavassiliou. Post-translational modifications and regulation of the RAS superfamily of GTPases as anticancer targets. *Nature Reviews Drug Discovery*, 6(7):540–555, JUL 2007.

[43] Steven Christopher Smith, Brian Nicholson, Matthew Nitz, Henry F. Frierson, Jr., Mark Smolkin, Garret Hampton, Wael El-Rifai, and Dan Theodorescu. Profiling Bladder Cancer Organ Site-Specific Metastasis Identifies LAMC2 as a Novel Biomarker of Hematogenous Dissemination. *American Journal of Pathology*, 174(2):371–379, FEB 2009.

[44] D Hanahan and RA Weinberg. The hallmarks of cancer. *Cell*, 100(1):57–70, JAN 7 2000.

[45] MJ van de Vijver, YD He, LJ van 't Veer, H Dai, AAM Hart, DW Voskuil, GJ Schreiber, JL Peterse, C Roberts, MJ Marton, M Parrish, D Atsma, A Witteveen, A Glas, L Delahaye, T van der Velde, H Bartelink, S Rodenhuis, ET Rutgers, SH Friend, and R Bernards. A gene-expression signature as a predictor of survival in breast cancer. *New England Journal of Medicine*, 347(25):1999–2009, DEC 19 2002.

[46] Kwang-Il Goh, Michael E. Cusick, David Valle, Barton Childs, Marc Vidal, and Albert-Laszlo Barabasi. The human disease network. *Proceedings of the National Academy of Sciences of the United States of America*, 104(21):8685–8690, MAY 22 2007.

[47] C Sotiriou, P Wirapati, S Loi, A Harris, S Fox, J Smeds, H Nordgren, P Farmer, V Praz, B Haibe-Kains, C Desmedt, D Larsimont, F Cardoso, H Peterse, D Nuyten, M Buyse, MJ Van de Vijver, J Bergh, MT Piccart, and M Delorenzi. Gene expression profiling in breast cancer: Understanding the molecular basis of histologic grade to improve prognosis. *Journal of the National Cancer Institute*, 98(4):262–272, FEB 15 2006.

[48] Richard Durbin, Sean Eddy, Anders Krogh, and Graeme Mitchison. Biological sequence analysis: probabilistic models of proteins and nucleic acids. cambridge univ, 1998.

[49] SF Altschul, TL Madden, AA Schaffer, JH Zhang, Z Zhang, W Miller, and DJ Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, SEP 1 1997.

[50] J Aach and GM Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495–508, JUN 2001.

[51] Subharup Guha, Yi Li, and Donna Neuberg. Bayesian hidden Markov Modeling of array CGH data. *Journal of the American Statistical Association*, 103(482):485–497, JUN 2008.

[52] L Liao and WS Noble. Combining pairwise-sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. *Journal of Computational Biology*, 10(6):857–868, 2003.

[53] JD THOMPSON, DG HIGGINS, and TJ GIBSON. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, NOV 11 1994.

[54] Nicolas Stransky, Celine Vallot, Fabien Reyal, Isabelle Bernard-Pierrot, Sixtina Gil Diez de Medina, Rick Segraves, Yann de Rycke, Paul Elvin, Andrew Cassidy, Carolyn Spraggon, Alexander Graham, Jennifer Southgate, Bernard Asselain, Yves Allory, Claude C. Abbou, Donna G. Albertson, Jean Paul Thiery, Dominique K. Chopin, Daniel Pinkel, and Francois Radvanyi. Regional copy

number-independent deregulation of transcription in cancer. *Nature Genetics*, 38(12):1386–1396, DEC 2006.

[55] Markus Heidenblad, David Lindgren, Tord Jonson, Fredrik Liedberg, Srinivas Veerla, Gunilla Chebil, Sigurdur Gudjonsson, Ake Borg, Wiking Mansson, and Mattias Hoglund. Tiling resolution array CGH and high density expression profiling of urothelial carcinomas delineate genomic amplicons and candidate target genes specific for advanced tumors. *BMC Medical Genomics*, 1, JAN 31 2008.

[56] Jun Liu, Sanjay Ranka, and Tamer Kahveci. Classification and feature selection algorithms for multi-class CGH data. *Bioinformatics*, 24(13):I86–I95, JUL 1 2008. 16th ISMB Conference on Intelligent Systems for Molecular Biology, Toronto, CANADA, JUL 19-23, 2008.

[57] M. Oeggerli, P. Schraml, C. Ruiz, M. Bloch, H. Novotny, M. Mirlacher, G. Sauter, and R. Simon. E2F3 is the main target gene of the 6p22 amplicon with high specificity for human bladder cancer. *Oncogene*, 25(49):6538–6543, OCT 19 2006.

[58] NJ HIGHAM. Computing a Nearest Symmetric Positive Semidefinite Matrix. *Linear Algebra And Its Applications*, 103:103–118, MAY 1988.

[59] AB Olshen, ES Venkatraman, R Lucito, and M Wigler. Circular binary segmentation for the analysis of array-based DNA copy number data. *Biostatistics*, 5(4):557–572, OCT 2004.

[60] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems 14*, pages 945–952. MIT Press, 2001.

[61] Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.

[62] Mikhail Belkin and Partha Niyogi. Using manifold stucture for partially labeled classification. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 929–936. MIT Press, Cambridge, MA, 2003.

[63] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[64] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. Label Propagation and Quadratic Criterion. In Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors, *Semi-Supervised Learning*, pages 193–216. MIT Press, 2006.

[65] Fei Wang and Changshui Zhang. Label propagation through linear Neighborhoods. *IEEE Transactions on Knowledge and Data Engineering*, 20(1):55–67, JAN 2008.

[66] Hong Cheng, Zicheng Liu, and Jie Yang. Sparsity induced similarity measure for label propagation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 317 –324, 29 2009-Oct. 2 2009.

[67] Nanhai Yang, Yuanyuan Sang, Ran He, and Xiukun Wang. Label propagation algorithm based on non-negative sparse representation. In Kang Li, Li Jia, Xin Sun, Minrui Fei, and George Irwin, editors, *Life System Modeling and Intelligent Computing*, volume 6330 of *Lecture Notes in Computer Science*, pages 348–357. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-15615-1_42.

[68] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press, 2001.

[69] ST Roweis and LK Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323+, DEC 22 2000.

[70] Shuicheng Yan and Huan Wang. Semi-supervised learning by sparse representation. In *SIAM International Conference on Data Mining*, pages 792–801, 2009.

[71] H. Sebastian Seung Daniel D. Lee. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13*, pages 556–562. MIT Press, 2000.

[72] Dijun Luo, C. Ding, Heng Huang, and Tao Li. Non-negative laplacian embedding. In *Data Mining, 2009. ICDM '09. Ninth IEEE International Conference on*, pages 337 –346, Dec. 2009.

[73] JB Shi and J Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, AUG 2000.

[74] M Belkin and P Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, JUN 2003.

[75] AS Georghiades, PN Belhumeur, and DJ Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, JUN 2001.

[76] Markus Breitenbach and Gregory Z. Grudic. Clustering through ranking on manifolds. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 73–80, New York, NY, USA, 2005. ACM.

[77] Maria Teresa Landi, Tatiana Dracheva, Melissa Rotunno, Jonine D. Figueroa, Huaitian Liu, Abhijit Dasgupta, Felecia E. Mann, Junya Fukuoka, Megan Hames, Andrew W. Bergen, Sharon E. Murphy, Ping Yang, Angela C. Pesatori, Dario Consonni, Pier Alberto Bertazzi, Sholom Wacholder, Joanna H. Shih, Neil E. Caporaso, and Jin Jen. Gene Expression Signature of Cigarette Smoking and Its Role in Lung Adenocarcinoma Development and Survival. *PLoS ONE*, 3(2), FEB 20 2008.

[78] Li-Jen Su, Ching-Wei Chang, Yu-Chung Wu, Kuang-Chi Chen, Chien-Ju Lin, Shu-Ching Liang, Chi-Hung Lin, Jacqueline Whang-Peng, Shih-Lan Hsu, Chen-Hsin Chen, and Chi-Ying F. Huang. Selection of DDX5 as a novel internal control for Q-RT-PCR from microarray data using a block bootstrap re-sampling scheme. *BMC Genomics*, 8, JUN 1 2007.

[79] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning with application to clustering with side-information. In S. Thrun

S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, Cambridge, MA, 2003.

[80] Inderjit S. Dhillon, Subramanyam Mallela, and Rahul Kumar. A divisive information theoretic feature clustering algorithm for text classification. *J. Mach. Learn. Res.*, 3:1265–1287, March 2003.

[81] Wei Liu, Shiqian Ma, Dacheng Tao, Jianzhuang Liu, and Peng Liu. Semi-supervised sparse metric learning using alternating linearization optimization. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 1139–1148, New York, NY, USA, 2010. ACM.

[82] Wei Liu, Junfeng He, and Shih-Fu Chang. Large graph construction for scalable semi-supervised learning. In *ICML*, pages 679–686, 2010.

[83] E. S. Venkatraman and Adam B. Olshen. A faster circular binary segmentation algorithm for the analysis of array CGH data. *Bioinformatics*, 23(6):657–663, MAR 15 2007.

[84] Robert Tibshirani and Pei Wang. Spatial smoothing and hot spot detection for CGH data using the fused lasso. *Biostatistics*, 9(1):18–29, JAN 2008.

[85] Jean-Philippe Vert and Kevin Bleakley. Fast detection of multiple change-points shared by many signals using group lars. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2343–2351. 2010.

[86] Kevin Bleakley and Jean-Philippe Vert. The group fused lasso for multiple change-point detection, 2011, arXiv:1106.4199.

[87] Gen Nowak, Trevor Hastie, Jonathan R. Pollack, and Robert Tibshirani. A fused lasso latent feature model for analyzing multi-sample aCGH data. *Biostatistics*, 12(4):776–791, OCT 2011.

[88] J. Liu, S. Ji, and J. Ye. *SLEP: Sparse Learning with Efficient Projections*. Arizona State University, 2009.

[89] Jingu Kim, Renato Monteiro, and Haesun Park. Group sparsity in nonnegative matrix factorization. In *Proceedings of the 12th SIAM International Conference on Data Mining*, page 851, 2012.

[90] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Convex optimization with sparsity-inducing norms.

[91] JR Pollack, T Sorlie, CM Perou, CA Rees, SS Jeffrey, PE Lonning, R Tibshirani, D Botstein, AL Borresen-Dale, and PO Brown. Microarray analysis reveals a major direct role of DNA copy number alteration in the transcriptional program of human breast tumors. *Proceedings of the National Academy of Sciences of the United States of America*, 99(20):12963–12968, OCT 1 2002.

[92] JS Reis, D Steele, S Di Palma, RL Jones, K Savage, M James, F Milanezi, FC Schmitt, and A Ashworth. Distribution and significance of nerve growth factor receptor (NGFR/p75(NTR)) in normal, benign and malignant breast tissue. *Modern Pathology*, 19(2):307–319, FEB 2006.

[93] Bogyou Kim, Hye Jin Nam, Ki Eun Pyo, Min Jung Jang, Ik Soo Kim, Dongha Kim, Kyungjin Boo, Seung Hoon Lee, Jong-Bok Yoon, Sung Hee Baek, and Jung Hwa Kim. Breast cancer metastasis suppressor 1 (BRMS1) is destabilized by the Cul3-SPOP E3 ubiquitin ligase complex. *Biochemical and Biophysical Research Communications*, 415(4):720–726, DEC 2 2011.

[94] SW Luoh, N Venkatesan, and R Tripathi. Overexpression of the amplified Pip4k2 beta gene from 17q11-12 in breast cancer cells confers proliferation advantage. *Oncogene*, 23(7):1354–1363, FEB 19 2004.

[95] Brian J. Kluk, Yebo Fu, Trina A. Formolo, Lei Zhang, Anne K. Hindle, Yan-gao Man, Robert S. Siegel, Patricia E. Berg, Chuxia Deng, Timothy A. McCaffrey, and Sidney W. Fu. BP1, an Isoform of DLX4 Homeoprotein, Negatively Regulates BRCA1 in Sporadic Breast Cancer. *International Journal of Biological Sciences*, 6(5):513–524, 2010.

[96] Antonis Kourtidis, Ritu Jain, Richard D. Carkner, Cheryl Eifert, M. Julia Brosnan, and Douglas S. Conklin. An RNA Interference Screen Identifies Metabolic

Regulators NR1D1 and PBP as Novel Survival Factors for Breast Cancer Cells with the ERBB2 Signature. *Cancer Research*, 70(5):1783–1792, MAR 1 2010.

[97] Annika Brendle, Haixin Lei, Andreas Brandt, Robert Johansson, Kerstin Enquist, Roger Henriksson, Kari Hemminki, Per Lenner, and Asta Foersti. Polymorphisms in predicted microRNA-binding sites in integrin genes and breast cancer: ITGB4 as prognostic marker. *Carcinogenesis*, 29(7):1394–1399, JUL 2008.

[98] Ronny Luss and Alexandre D'Aspremont. Support vector machine classification with indefinite kernels. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 953–960. MIT Press, Cambridge, MA, 2008.

[99] Jianhui Chen and Jieping Ye. Training svm with indefinite kernels. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 136–143, New York, NY, USA, 2008. ACM.

[100] Felix Sanchez-Garcia, Uri David Akavia, Eyal Mozes, and Dana Pe'er. JISTIC: Identification of Significant Targets in Cancer. *BMC BIOINFORMATICS*, 11, APR 14 2010.

[101] Buyue Qian and Ian Davidson. Semi-supervised dimension reduction for multi-label classification. In *AAAI Conference on Artificial Intelligence*, 2010.