View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by University of Waterloo's Institutional Repository

Feature Selection for Gene Expression Data Based on Hilbert-Schmidt Independence Criterion

by

Hadi Zarkoob

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Statistics

Waterloo, Ontario, Canada, 2010

© Hadi Zarkoob 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

DNA microarrays are capable of measuring expression levels of thousands of genes, even the whole genome, in a single experiment. Based on this, they have been widely used to extend the studies of cancerous tissues to a genomic level. One of the main goals in DNA microarray experiments is to identify a set of relevant genes such that the desired outputs of the experiment mostly depend on this set, to the exclusion of the rest of the genes. This is motivated by the fact that the biological process in cell typically involves only a subset of genes, and not the whole genome. The task of selecting a subset of relevant genes is called feature (gene) selection. Herein, we propose a feature selection algorithm for gene expression data. It is based on the Hilbert-Schmidt independence criterion, and partly motivated by Rank-One Downdate (R1D) and the Singular Value Decomposition (SVD). The algorithm is computationally very fast and scalable to large data sets, and can be applied to response variables of arbitrary type (categorical and continuous). Experimental results of the proposed technique are presented on some synthetic and well-known microarray data sets. Later, we discuss the capability of HSIC in providing a general framework which encapsulates many widely used techniques for dimensionality reduction, clustering and metric learning. We will use this framework to explain two metric learning algorithms, namely the Fisher discriminant analysis (FDA) and closed form metric learning (CFML). As a result of this framework, we are able to propose a new metric learning method. The proposed technique uses the concepts from normalized cut spectral clustering and is associated with an underlying convex optimization problem.

Acknowledgements

I'd like to extend my gratitude to my supervisors Prof. Ali Ghodsi and Prof. Mohammad Kohandel, for their support, patience and the lessons that they have taught me. Their help and support was truly beyond the traditions of academic study. I would also like to thank Prof. Siv Sivaloganathan and Prof. Mu Zhu for carefully reading my thesis and providing valuable comments. I would like to thank my fellow graduates students Vahed Maroufi, Reza Ramezan, Colin Phipps and Easwar Magesan. I enjoyed their kind support during my studies. And last, but not the least, I'd like to thank my wonderful family for the unwavering support they have consistently given to me during my life.

Dedication

To my family!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

For a long time, the fields of biology and medicine dealt with the observable traits of organisms, such as, their morphology, development and behaviors. These traits are referred to as the **phenotype** of the organism. The discovery of DNA molecules in 1940's, however, revolutionized biology by extending the studies to a genomic level. At this level, scientists study genes whose activation/deactivation lead to an organism's phenotype. In this chapter, we start by explaining the basic concepts of genes and molecular signature for an organism. We subsequently introduce the enabling technologies that are used to acquire such genomic information. As we will see, this information is usually organized in large-scale data sets, and statistical tools must be exploited to extract meaningful biological information. We conclude this chapter by introducing a number of key methodologies and algorithms from statistics and machine learning that are commonly used in the analysis of gene expression data.

1.1 Gene and Gene Expression

All the biological reactions in a cell depend on the information stored in DNA molecules. DNA molecules consist of two long strands entwined in the shape of a double helix. Each of these strands is a long polymer of simpler units called nucleotides. Each nucleotide consists of a backbone and one of the four **base**s adenine (abbreviated A), cytosine (C), guanine (G) and thymine (T). The bases, attached to the two strands, are bound to each other via hydrogen bonds. It is an important property of bases that base A bonds only to T and base C bonds only to G. As a result, the sequence of bases in a strand has a one-to-one correspondence with the bases on the complementary strand. **Gene**s are segments of DNA that contain genetic information. The information stored in genes can be used in the development and functionality of cells only after the corresponding segments in DNA are converted into **RNA** molecules and subsequently **protein**s. In this case, we say the gene is **expressed** and the process is referred to as the **central dogma of biology**. The first step, the process of building RNA molecules based on DNA molecules, is called **transcription**. RNA molecules are structurally very similar to DNA molecules with some important differences. They differ from DNA molecules in that the base uracil (U) replaces thymine (T); further, RNA has only one strand. Base A on the DNA sequence results in the incorporation of base U in the RNA sequence. Similarly, base G results in the incorporation of base C on the RNA. It is of interest to note that genes may contain both coding and non-coding regions. These regions are referred to as **exon**s and **intron**s, respectively. The non-coding regions of gene are not used in creating proteins. As a result, a large part of initial RNA molecules that correspond to introns, will be eliminated via a process called **splicing**. The resulting RNA molecule that is much shorter than the original one is called **messenger RNA** or **mRNA**. The reason for this name is that mRNA acts as a messenger leaving the nucleus and moving into the cytoplasm. It then attaches to submolecular components called **ribosome**s. Proteins are comprised of sequences of constructing units named **amino acid**s. Each triple of nucleotides in the mRNA is called a **codon** and will be mapped to a specific amino acid. The resulting sequence of amino acids is the protein corresponding to the original gene. The process of creating protein from RNA molecules is called **translation**.

The entire genetic information encoded in DNA molecules is called **genome**. All cells

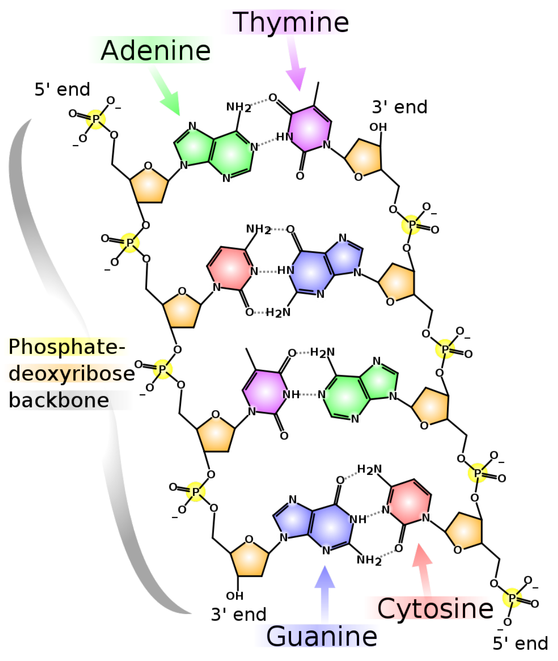


Figure 1.1: DNA chemical structure (taken from Wikipedia under the GNU Free Documentation License).

in an organism have the same genome[1]. The number of genes in the human genome was recently estimated to be 20000-25000 [9]. These genes exist in the DNA molecules of all cells in the body; however, as mentioned earlier, a gene impacts functioning and development of living cells only if it is *expressed*. Based on this, an interesting task would be to evaluate the functional genes in a cell at any time. One way to accomplish this is to examine the RNA transcripts obtained from that cell, and determine the amount of RNA corresponding to each of the genes. Due to the large number of genes present in an organism's genome, this is not an easy task and was not feasible until recently when **DNA microarray**s were developed. DNA microarrays are high-throughput assays that enable measurement of the expression levels of thousands of gene, even the whole genome, in a single experiment. In the next section we will introduce DNA microarrays and discuss their principles of operation in more detail.

[1]Genes, however, may have variations called *alleles* among different members of an organism.

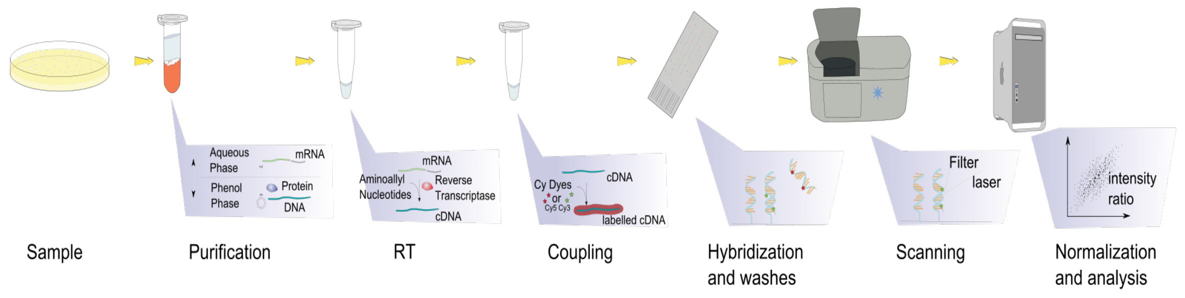


Figure 1.2: A typical DNA microarray experiment. (The image is taken from Wikipedia. It is released into the public domain by the copyright holder.)

1.2 DNA Microarrays

DNA microarray are high throughput biological assays that enable measurement of the expression levels of thousands of genes in a single experiment. The operational principle of DNA microarrays is based on the fact that DNA sequences tend to combine with each other when their bases are matched (this is called **base pairing**). On a DNA microarray chip there are a large number of spots each containing picomoles (10^{-12} moles) of a specific DNA sequences. These DNA sequences are known as **probe**s. Each probe consists of about 20-70 nucleotide in length. (The length of probes varies among different types of microarrays.) Each gene is represented by a number of probes on the chip. As mentioned earlier, to identify the expression levels of genes in a given sample, one can measure the mRNA content obtained from that sample. To hybridize with the DNA sequences available on the microarray chip, mRNA has to be transformed back into DNA sequences. This process is called **reverse transcription** and the resulting DNA sequences are referred to as **complementary DNA** or **cDNA**. Then cDNA molecules will be applied to the microarray chip. After washing off the chip, only strong bonds that correspond to paired strands will remain. The cDNA molecules are pre-labeled with fluorescent tags so that matched molecules can be detected after the hybridization process. The labels are subsequently scanned, and the strength of signal at each probe is used to determine the amount of the corresponding gene in the original sample.

1.3 Analysis of Gene Expression Data

In this section we assume that we are given the gene expression data for a number of samples and explore different methodologies that are widely used for the analysis of such data. Assume we have the expression levels of m genes given for a number of n samples. Samples might be associated with **response variable**s or **label**s. For instance, they may be labeled as cancerous or healthy samples (**categorical** response variable), or be associated with survival time of corresponding patients (**continuous** response variable). In microarray experiments, m is typically of the order of tens of thousands and n is of the order of only tens or hundreds. Let $\mathbf{X}_{m\times n}$ be a real-valued matrix containing the gene expression data.

Data mining techniques are generally divided into three different groups: **unsupervised**, **supervised** and **semi-supervised** methods. In unsupervised methods, the samples are not associated with labels and the analysis is performed on the matrix $\mathbf{X}$ introduced above. Tasks like clustering the samples or genes and dimensionality reduction belong to this group. In supervised sampling, there are some labels associated with samples, and these labels are used in the data analysis. For example, we might use the data labels to train a classifier distinguishing between different types of samples. Another important example of supervised techniques is *feature selection* (also known as *variable selection*). In feature selection, we select a small subset of genes that are most relevant with respect to a given set of labels. In semi-supervised techniques, labels are given only for a number of samples. However, samples without labels still can be used to enhance the performance of learning algorithms such as classification. In the context of DNA microarrays, we mostly deal with supervised and unsupervised methods. In the following, we explore these techniques in more detail.

1.3.1 Unsupervised Methods

Principal Component Analysis

Principal component analysis (PCA) is a very popular tool for dimension reduction and data visualization. It provides the directions of maximal variability in the data. Mathematically speaking, assume we have n data points of dimension m stacked in matrix $\mathbf{X}_{m\times n}$. We assume matrix $\mathbf{X}$ is normalized such that its rows add to zero. Assume $\mathbf{U}_{m\times d}$ is a transformation that maps the data onto a d-dimensional space, $\mathbf{U}^{\mathrm{T}}\mathbf{X}$. Columns of $\mathbf{U}$ represent the directions into which the data is mapped. PCA finds a linear transformation $\mathbf{U}$ such that:

$$\mathrm{Tr}(\,(\mathbf{U}^{\mathrm{T}}\mathbf{X})\,(\mathbf{U}^{\mathrm{T}}\mathbf{X})^{\mathrm{T}}) = \mathrm{Tr}(\mathbf{U}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{U})$$

is maximized. When $d = 1$, the above trace is proportional to the variance of data points along the direction $\mathbf{U}_{m\times 1}$. For $d > 1$, the above trace is proportional to the summation over variances of data along the d directions (vectors) stored in the columns of $\mathbf{U}$.

The above optimization problem is a standard form of trace maximization problem. Using the Rayleigh-Ritz theorem (see e.g. [31]), it can be shown that the solution under the condition that $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}_{d\times d}$ is given by setting the columns of matrix $\mathbf{U}$ (i.e., the directions of transformation) equal to the top d eigenvectors of matrix $\mathbf{X}\mathbf{X}^{\mathrm{T}}$. The value of the corresponding eigenvalue gives the variance along each eigenvector.

There is a close relationship between PCA and the singular value decomposition (SVD). SVD decomposes the matrix $\mathbf{X}_{m\times n}$ as $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}}$, where $\mathbf{U}_{m\times m}$ and $\mathbf{V}_{n\times n}$ are unitary matrices and $\boldsymbol{\Sigma}_{m\times n}$ is a diagonal matrix with nonnegative elements. It can be shown that the columns of $\mathbf{U}$ are given by eigenvectors of $\mathbf{X}\mathbf{X}^{\mathrm{T}}$. One may combine matrices $\boldsymbol{\Sigma}$ and $\mathbf{V}$ in a single matrix $\mathbf{W} = \mathbf{V}\boldsymbol{\Sigma}^{\mathrm{T}}$, and write the SVD of matrix $\mathbf{X}$ as $\mathbf{X} = \mathbf{U}\mathbf{W}^{\mathrm{T}}$. In this format, the columns of $\mathbf{W}$ are called **score**s. They can be used to provide a low-dimensional representation of $\mathbf{X}$ in terms of the bases stored in columns $\mathbf{U}$. Columns of $\mathbf{U}$ are called **loading**s in this representation.

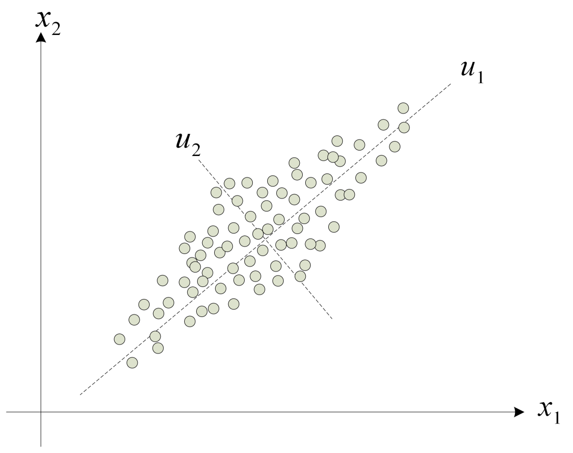


Figure 1.3: PCA identify the directions along which data has maximal variability.

Non-negative Matrix Factorization

In many applications, the data matrix consists of only non-negative elements. Examples are image intensities and DNA microarray data. Non-negative matrix factorization (NMF) decomposes the non-negative data matrix $\mathbf{X}_{m\times n}$ into a product of two *non-negative* matrices $\mathbf{U}_{m\times d}$ and $\mathbf{W}^{\mathrm{T}}$ ($\mathbf{W}_{n\times d}$) such that $\mathbf{X} \approx \mathbf{U}\mathbf{W}^{\mathrm{T}}$ [30]. There are two measures commonly used in evaluating the error in this approximation. The first one is sum of squared errors (or Frobenius norm). Based on this, the non-negative factorization of matrix $\mathbf{X}$ is obtained by solving an optimization problem of the form:

$$\min_{\mathbf{U},\mathbf{W}} \|\mathbf{X} - \mathbf{U}\mathbf{W}^{\mathrm{T}}\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} \left(\mathbf{X}_{ij} - (\mathbf{U}\mathbf{W}^{\mathrm{T}})_{ij}\right)^2$$

where $\mathbf{U}$ and $\mathbf{W}$ are non-negative matrices and $\|\cdot\|_F$ denotes Frobenius norm of a matrix. The second measure is Kullback-Leibler divergence,

$$\min_{\mathbf{U},\mathbf{W}} D_{KL}(\mathbf{X}\|\mathbf{U}\mathbf{W}^{\mathrm{T}}) = \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{X}_{ij} \log \frac{\mathbf{X}_{ij}}{(\mathbf{U}\mathbf{W}^{\mathrm{T}})_{ij}} - \mathbf{X}_{ij} + (\mathbf{U}\mathbf{W}^{\mathrm{T}})_{ij}$$

where $\log(\cdot)$ denotes the natural logarithm. Matrices $\mathbf{U}$ and $\mathbf{W}$ obtained in NMF frequently happen to be sparse. Based on this, they may be used to extract rank-one submatrices of the original matrix $\mathbf{X}$. More specifically, when the non-negative matrix $\mathbf{X}_{m\times n}$ consists of, say d approximately rank-one submatrices, then these submatrices will be identified by d

terms in a d-order NMF as $\mathbf{X}_{m\times n} \approx \mathbf{U}\mathbf{W}^{\mathrm{T}} = \mathbf{u}_1\mathbf{w}_1^{\mathrm{T}} + \mathbf{u}_2\mathbf{w}_2^{\mathrm{T}} + \cdots + \mathbf{u}_d\mathbf{w}_d^{\mathrm{T}}$, where $\mathbf{u}_i$ and $\mathbf{w}_i$, $i = 1, 2, \cdots, d$, denote the columns of $\mathbf{U}$ and $\mathbf{W}$, respectively. Note that all the components here are non-negative and so they cannot cancel out each other. Instead they combine together to reconstruct the original matrix $\mathbf{X}$. Identifying such submatrices can be thought of as simultaneously clustering features and samples in the data matrix $\mathbf{X}$. The process of simultaneously extracting features and samples in a data matrix is called *bi-clustering* or *two-way clustering*. We will see an example of using NMF for bi-clustering in gene expression data in the second chapter.

Clustering

One of the other important tasks in the analysis of gene expression data is clustering. Clustering is the process of grouping objects together based on their similarities/dissimilarities. In microarray experiments, clustering may be performed both on samples and genes. Clustering of samples might provide insight into subtypes of samples and their molecular signatures. Clustering of genes, on the other hand, might help to identify the function of unknown genes based on the known genes in their group [2]. Common techniques for clustering in gene expression data are hierarchial clustering, k-means clustering and self organizing maps [29].

1.3.2 Supervised Methods

As mentioned earlier, supervised methods in machine learning use sample labels during the analysis. One important supervised task is **classification**. In classification, we train a classifier based on the samples in the *training* set, and then use this classifier to predict the class (label) of the new samples in the *test* set. The most common classification technique used for DNA microarrays is the **support vector machine (SVM)**. It finds a separating hyperplane between data points from different classes such that the distance between the hyperplane and boundary points is maximized (see Figure 1.4). A mathematical description of SVM in the simplest case when there are two classes and the classes are linearly separable is the following. Let $\mathbf{x}_1, \cdots, \mathbf{x}_n$ be n data points in $\mathbb{R}^m$ with labels $y_i \in \{\pm 1\}$, $i = 1, \cdots, n$. Since classes are assumed to be linearly separable there exists a hyperplane (and, in fact, a family of hyperplanes) in $\mathbb{R}^m$ such that data points of different classes are on different sides of this hyperplane. The SVM classification method finds the hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ such that the distances of the hyperplane to the closest member of each class is maximized. It can be shown [34, 10] that this problem may be formulated as the following minimization problem:

$$\begin{aligned} &\min\nolimits_{\mathbf{w}} \tfrac{1}{2}\|\mathbf{w}\|_2^2 \\ \text{subject to} \quad &y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, \cdots, n \end{aligned} \tag{1.1}$$

where $\|\mathbf{w}\|_2^2 = \mathbf{w} \cdot \mathbf{w}$ is the squared l_2-norm of the normal vector of the hyperplane. This optimization problem is usually transformed into its dual form and is solved using quadratic programming.

Other classification techniques used for DNA microarrays are **nearest neighbors** and **decision trees**. For more information on these algorithms see [43].

1.3.3 Feature Selection Methods

Feature (or variable) selection is one of the most important tasks in analyzing high-dimensional data. It extracts the most relevant features in a data set such that the response variable mainly depends on these features, and not the remaining ones. Feature selection helps to enhance both the efficiency and accuracy of algorithms in analyzing high dimensional data. This is specially relevant in the case of gene expression data where a large number of genes are usually unrelated to a specific biological process. In many cases, feature selection is a preprocessing task for a subsequent classification or regression problem. Based on this, the feature selection techniques are usually divided into three groups: **filters**, **wrappers** and **embedded** methods [28, 36]. In a filter method, the feature selection task is independent of the subsequent classification procedure. Most of the filter methods consider genes *individually* and evaluate them based on their dependence with the given response variable. To

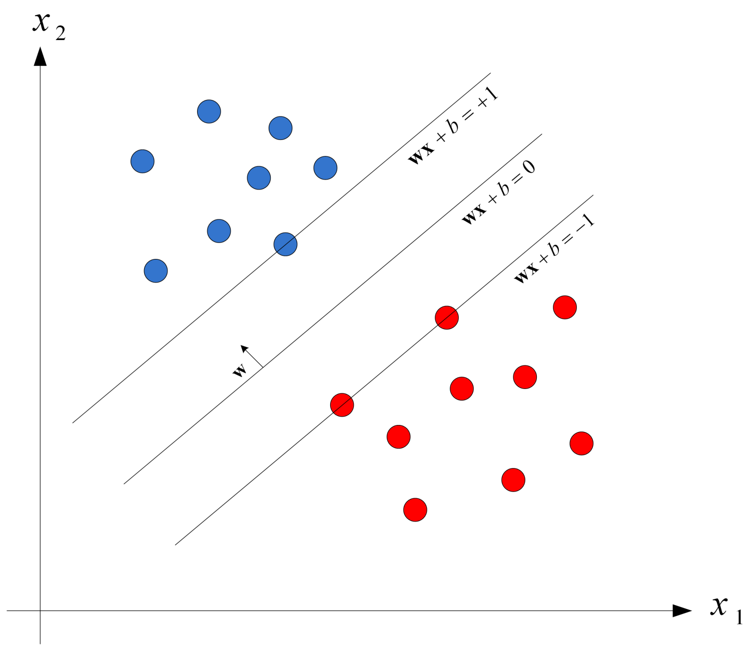


Figure 1.4: Support Vector Machine (SVM) classification technique.

this end, dependence measures based on signal-to-noise ratio, t-test and mutual information have been proposed [22, 25]. To account for interaction between genes, one way is to build a classifier using a given subset of genes and use the classification error as a measure for evaluating the quality of that subset. Having this measure of merit, different algorithms can be devised to extract an optimal subset of genes. Feature selection methods that use classification techniques in the above fashion are called wrappers (see, e.g., [54, 46]). Another class of feature selection techniques are embedded methods. In an embedded method, the intrinsic properties of classifiers are used to assign degrees of relevance to features [26, 14, 32]. For instance, if a linear classifier is built based on the data, the relevance of features can be measured using the corresponding weights in the linear classifier.

In the following, we introduce a number of common filter and embedded methods for gene selection. We do not elaborate further on wrapper methods, as they are not used commonly in the literature for gene selection. We refer the interested reader to [36].

Filter Methods

Fold change: This technique is a simple yet effective method for feature selection. It uses the ratio between the mean of the expression levels in two different classes as a measure of importance. Usually a ratio equal to 2 or more is considered to be significant. In the context of gene expression data analysis, it is common to work with $\log_2$ of the expression levels. In this way, a difference equal to one or more in $\log_2$ of expression levels is considered to be significant.

t-test: A Problem with fold change is that it does not take into account the variance of gene expression levels in each class. To see the role of variances, consider the following two synthetic scenarios. In both scenarios we assume there are two types (classes) of samples, and the goal is to determine whether a given gene is differentially expressed in the two sample types or not. In the first scenario, the ($\log_2$ of) expression levels of genes in the first and second sample types are assumed to be $\{0.99, 1, 1.01\}$ and $\{1.99, 2, 2.01\}$, respectively. In the second scenario, the expressions levels are assumed to $\{0, 1, 2\}$ and $\{1, 2, 3\}$. The mean of expression levels (and thus their difference) is the same in both scenarios; however, it is clear that the considered gene in the first scenario is much more likely to be significantly differentially expressed compared to gene studied in the the second scenario.

A t-test accounts for variance of expression levels in different classes by using the following ratio to decide about differentially expressed genes

$$T = \frac{m_1 - m_2}{\sqrt{\sigma_1^2 + \sigma_2^2}} \tag{1.2}$$

A generalization for t-test for multi-class problems is formulated under the acronym ANOVA (ANalysis Of VAriance methods). These methods use the F-test instead of t-test to evaluate the dependence between genes and class labels (see e.g. [15]).

Signal-to-noise ratio: Signal-to-noise ratio method is similar to the t-test; but it uses the following signal-to-noise ratio to decide about differentially expressed genes

$$\mathrm{SNR} = \frac{m_1 - m_2}{\sigma_1 + \sigma_2} \tag{1.3}$$

Mutual information (MI): MI provide a measure of dependence between two random variables. The mutual information between two random variables equals zero if and only if the two random variables are independent. Mathematically speaking, mutual information between random variables X and Y is defined to be

$$I(X;Y) = \int_Y \int_X p(x,y) \log\left(\frac{p(x,y)}{p_1(x)\, p_2(y)}\right) dx\, dy, \tag{1.4}$$

where $p(x,y)$ is the joint probability density function of X and Y, and $p_1(x)$ and $p_2(y)$ are the marginal probability density functions of X and Y, respectively. A difficulty with using mutual information is that in practice the joint probability density functions are not given and one needs to estimate them based on the data.

Embedded Methods

All filter methods introduced above consider genes individually when evaluating dependence on the given response variable. As a result, they are computationally fast, but they can not account for the interaction between genes. In the following we present in more detail two embedded methods, namely **s**upport **v**ector **m**achine-**r**ecursive **f**eature **e**limination (SVM-RFE) [26] and **a**pproximation of the ze**ro**-norm **m**inimization-**s**upport **v**ector **m**achine (AROM-SVM) [53]. As previously mentioned, embedded methods use the intrinsic properties of classifiers to evaluate the degree of relevance for each features. This enables them to somehow take into account the interactions between genes.

SVM-RFE: The SVM-RFE is a widely used method for feature selection from gene expression data. SVM-RFE is an iterative algorithm that works based on the SVM classifier. To explain this method, we assume there are only two classes. At each step, SVM-RFE method builds an SVM classifier based on the data and uses the absolute value of the weights in the normal vector of the classifier as a measure of relevance for the features. (By normal vector of the classifier we mean vector $\mathbf{w}$ in the classifier $\text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$, see Section 1.3.2.) To obtain a small subset of features, SVM-RFE use *Recursive Feature Elimination* method to eliminate irrelevant features at each step. That means SVM-RFE modifies data at each step by eliminating the least relevant feature. This feature corresponds to the lowest weight in $\mathbf{w}$. The procedure is repeated until the desired number of features is obtained.

Since SVM-RFE requires the training if an SVM classifier at each step, it would be computationally expensive to eliminate only one feature at each step. One can increase the speed of this algorithm by reducing the number of features by a certain percentage at each iteration instead of eliminating the features one by one.

AROM-SVM: To explain the AROM-SVM feature selection method, we recall from Section 1.3.2 that SVM can be formulated as follows:

$$\begin{aligned} &\min\nolimits_{\mathbf{w}} \|\mathbf{w}\|_2^2 \\ \text{subject to} \quad &y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, \cdots, n \end{aligned} \tag{1.5}$$

where $\|\mathbf{w}\|_2^2 = \mathbf{w} \cdot \mathbf{w}$ is the squared l_2-norm of $\mathbf{w}$. Induced by this formulation, Weston *et. al* [53] proposed minimizing the zero norm [2] of $\mathbf{w}$ instead of the second norm to obtain a sparse $\mathbf{w}$, as follows:

$$\begin{aligned} &\min\nolimits_{\mathbf{w}} \|\mathbf{w}\|_0 \\ \text{subject to} \quad &y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, \cdots, n \end{aligned} \tag{1.6}$$

Solving this optimization problem provides us with the separating hyperplane with the fewest nonzero elements in the normal vector $\mathbf{w}$. The features corresponding to these nonzero elements would be the important features obtained from feature selection process. The optimization problem presented in 1.6 is shown to be computationally hard [4]. Instead,

[2]Zero norm of a vector is defined as the number of non-zero elements in that vector.

Weston *et. al* [53] proposed an approximate method to solve this problem. They show that one can solve the following optimization problem to approximate the solution to the original one:

$$\begin{aligned} & \min_{\mathbf{w}} \textstyle\sum_{j=1}^{n} \ln(\epsilon + |w_j|) \\ \text{subject to} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, \cdots, n \end{aligned} \tag{1.7}$$

where ϵ is used to avoid zero arguments in $\ln(\cdot)$ when the elements of $\mathbf{w}$ are zero. The optimization problem presented in 1.7 is tractable in this form and can be solved using an iterative gradient-based method [53].

In chapter 3, we will use a sparse matrix factorization method, namely Rank-one Downdate (R1D), to obtain sparse solutions to the proposed optimization problem. We will see that this method is much simpler and faster than optimizing the zero-norm of the vectors.

Chapter 2

Analysis of Brain Tumors CD133+/− Expression Data

In the previous chapter, we introduced a number of widely used methodologies for the analysis of gene expression data. In this chapter, we present the result of two important techniques, namely Principal Component Analysis (PCA) and Non-negative Matrix Factorization (NMF), applied to a data set of real-world cancerous and normal samples. We will see how these techniques can be effectively used to extract information from gene expression data. This data set we use was obtained from the Singh Lab at McMaster University.

2.1 Introduction

In recent years, the stem cell hypothesis for brain tumors [39] has received an increasing amount of support. Based on this hypothesis, only a small fraction of brain tumor cells are responsible for initiating and maintaining brain tumors. This stem-like subset of cells are called **Brain Tumor Initiating Cell**s (**BTIC**s). Identification of biomarkers that are able to identify BTICs is one of the most important aspects of research on brain tumors. The surface protein prominin 1 (CD133) has been widely used in the literature for this purpose.

In [39], Singh *et. al* assert that the injection of only about 100 brain tumor cells which expressed the CD133 protein resulted in a brain tumor with the same phenotype as the original tumor *in vivo*. However, the injection of as many as 10'000 CD133- cells did not result in initiation of any brain tumor. Following these original experiments, the literature on brain tumor stem cells has witnessed a consistent series of works reporting on the stemness properties of CD133+ cell as opposed to their CD133- counterparts.

In spite of the existing evidence supporting the stem cell hypothesis for brain tumors, there are still many important questions that need to be addressed. Is CD133 a perfect biomarker for BTICs? What is the relation between expression of CD133 and stemness properties of cell such as proliferation and self-renewal? In addition, it is not clear yet if environmental factors such as hypoxia can affect the expression of CD133 in a cell. DNA microarrays are powerful tools capable of simultaneously measuring the expression levels of whole genes in the genome. Based on this, they can be used to unravel important information about genomic signatures of different brain tumor cells, and address these types of questions at a genomic level. DNA microarrays have been effectively used in a variety of applications ranging from classification and clustering of cancerous and normal sample data to functional analysis of new genes.

We have recently received experimental data containing the gene expression levels of 18 samples of cancerous and normal cells. These samples are organized in 9 pairs. Each pair consists of CD133+ and $CD133-$ samples of the same tissue. More specifically, the data consists of 3 normal pairs (obtained from normal brain tissue), 2 glioblastoma (GBM) pairs, 2 medulloblastoma pairs and 2 metastasized pairs. Glioblastoma and medulloblastoma are two malignant types of brain tumors. The metastasis samples correspond to tumors that originally developed in other tissues in the body and which have then metastasized to the brain. The samples in this data set are labeled as follows:

- Normal: hNC1+, hNC1-, hNC2+, hNC2-, hNC3+ and hNC3-
- GBM: BT47_GBM+, BT47_GBM-, BT63_GBM+, BT63_GBM-

- Medulloblastoma: BT88_Medullo+, BT88_Medullo-, Medullo+, Medullo-
- Metastasis: BT84_Lung Mets+, BT84_Lung Mets-, BT53_Mets+, BT53_Mets-

The signs + and - after sample names indicate the state of being CD133 positive or negative. A heatmap of expression levels of a limited number of genes given for these samples is shown in Figure 2.1.

As a first step, we have used the DNA microarrays chips Affymetrix© GeneChip Human Genome U133 Plus 2.0 arrays which are capable of measuring the expression levels of the whole human genome on a single array. These chips have 54675 probe sets, which results in 54675 features for each sample data point. The reason why the number of features is greater than the total number of genes is that some genes are represented by more than one probe on the array. We have successfully used a variety of well-known dimensionality reduction and clustering algorithms to distinguish between different tissues in our data, based solely on their gene expression profiles. Since this data is still not published, we are not able to name the individual genes; however, here we will show the capability of a number of techniques introduced in the previous chapter to effectively extract information from real-world gene expression data.

2.2 Application of Principal Component Analysis

We start our analysis by performing PCA on this data. It is common to work with $\log_2$ of expression levels instead of the original values. In this way, the range and distribution of expression levels will have a more reasonable behavior. Let $\mathbf{X}$ be the 54675×18 data matrix. Since rank of $\mathbf{X}$ and thus $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ is 18, PCP provides up to 18 directions of maximum variability in the data. As mentioned in subsection 1.3.1, the variances of projected points along these directions are given by the corresponding eigenvalues. Figure 2.2 plots 18 eigenvalues obtained from PCA. As shown, there is a gap between the first 9 eigenvalues and the rest of them. The last small eigenvalues (variances) might be due to noise only.

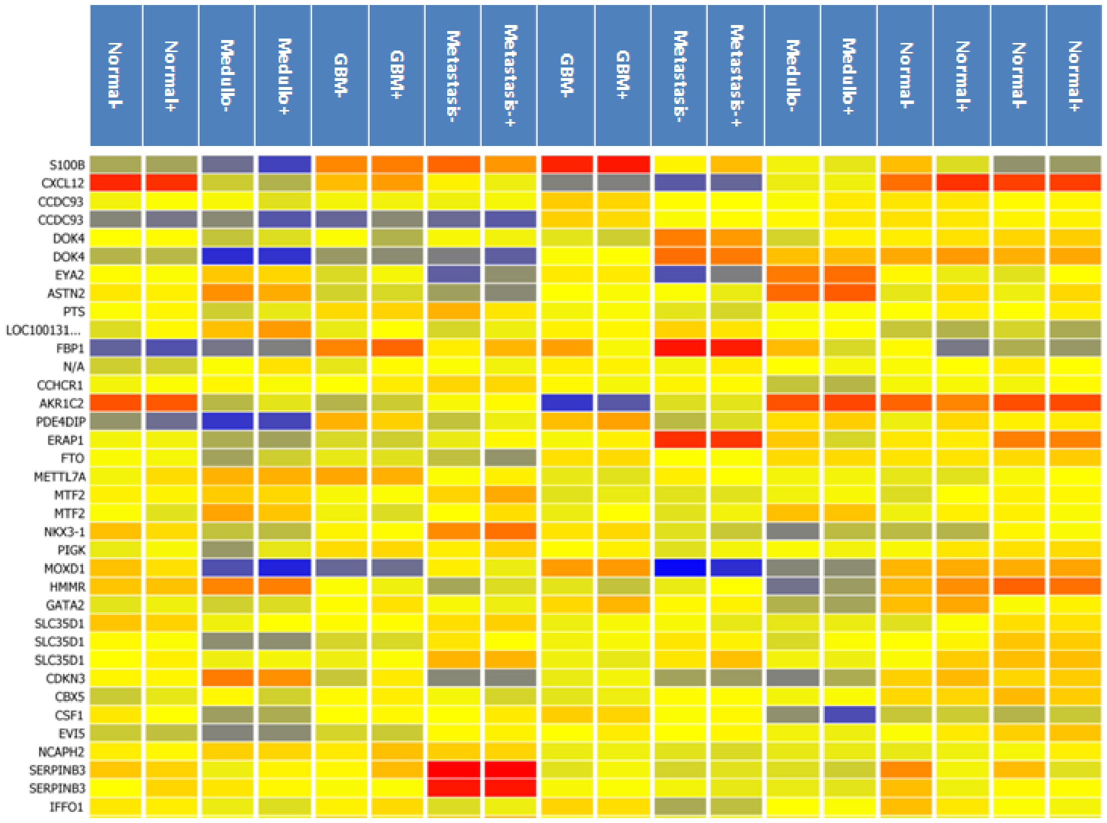


Figure 2.1: Heatmap of gene expression data.

We noted that the projection of points over the first principal component (PC) corresponds to the *date* in which data was created. More specifically, we received the data in two stages, and the 8 points of the first stage and 10 points of the second stage lie far apart after projection on the first PC, which gives the direction of maximal variability. However, we are not interested in this source of variation. Figure 2.3 depicts the projection of data points on the second principal component. One may see that the position of points on this direction well describes the type of the samples. Specifically, data points from each of the groups normal, GBM, Medulloblastoma and metastasis samples lie close to each other. The direction demonstrates the second maximum source of variation in data. One may use this direction as a classifier for a new unknown sample. It is interesting to note that the medulloblastoma sample lying at the very left of the diagram was first introduced as a GBM sample to us. However, after further investigations deducted from our mathematical analysis, we noted

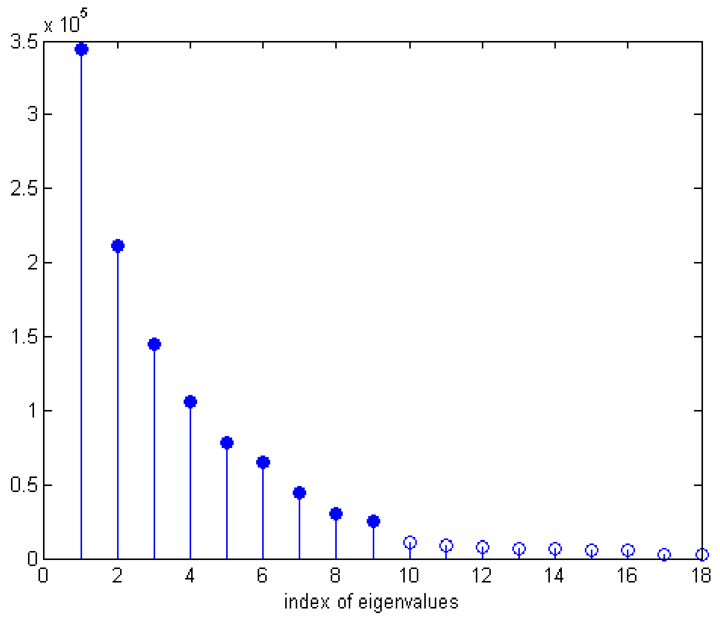


Figure 2.2: Eigenvalues obtained from PCA sorted in descending order. Each eigenvalue provides the variance of data projected on the corresponding eigenvector.

that the sample in fact corresponds to a medulloblastoma sample.

We normally expect one of the extracted directions to correspond to the CD133 *positive* or *negative* property of the sample, as this is another known source giving rise variation in the data. We noted that the ninth PC in fact does so. Figure 2.4 depicts the projection of data points on this principal component. Remember this direction is the last direction that was expected to reflect significant variation in the data. It is seen, however, that the separation between positive and negative samples is not perfect. This might be due to the noise in the DNA microarray experiments or due to the labeling process of the samples. Assigning CD133 positive/negative labels to samples requires performing a process called **cell sorting**, and there might be some experimental errors associated with this process.

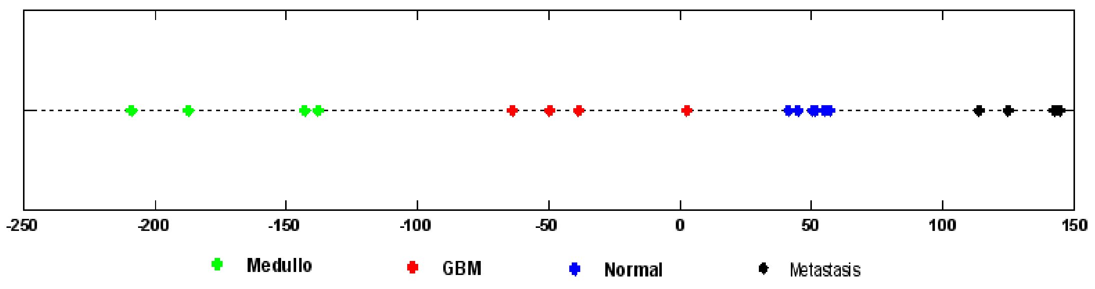


Figure 2.3: Projection of data points on the second principal component.

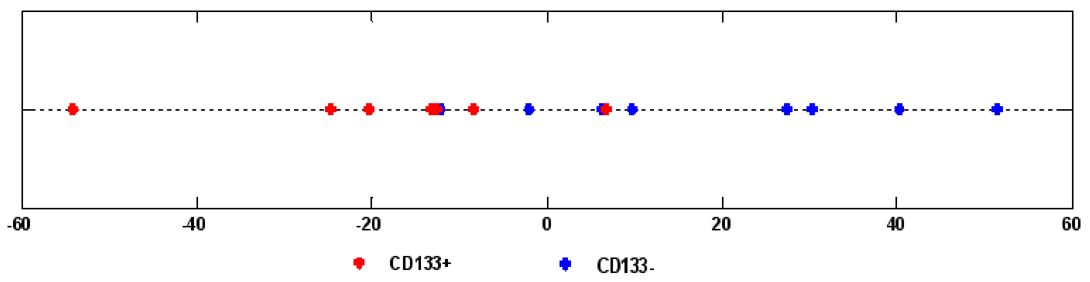


Figure 2.4: Projection of data points on the ninth principal component.

2.3 Application of Non-negative Matrix Factorization

The second method we study in this chapter is Non-negative Matrix Factorization (NMF). As mentioned in subsection 1.3.1, both PCA and NMF provide a decomposition of the data matrix of the form $\mathbf{X} \approx \mathbf{U}\mathbf{W}^{\mathrm{T}}$. However, in PCA there is no restriction on the sign of the elements in $\mathbf{X}$, $\mathbf{U}$ and $\mathbf{W}$. In contrast, NMF is applied to non-negative matrices and the elements of $\mathbf{U}$ and $\mathbf{W}$ are restricted to be non-negative. As mentioned in subsection 1.3.1, this property allows NMF to identify approximate rank-one submatrices in the data matrix $\mathbf{X}$, and in this way, simultaneously cluster samples and features. Figure 2.5 depicts the results of 3-order NMF applied to 12 samples, namely, BT47_GBM+, BT47_GBM-, BT63_GBM+, BT63_GBM-, BT88_Medullo+, BT88_Medullo-, hNC1+, hNC1-, hNC2+, hNC2-, hNC3+ and hNC3-. Note that NMF must be applied to the original gene expression data (that are always positive) and not to the $\log_2$ of expression levels. As shown in the bottom diagram, columns of matrix W are sparse and distinguish between normal, GBM and medulloblastoma samples.

It is instructive to compare the *score*s (see subsection 1.3.1) obtained from PCA with those obtained from NMF. Figure 2.6 depicts scores obtained from both NMF and PCA. One can see that the scores obtained from NMF are sparse, while those obtained from PCA are not. As a result, scores of NMF clearly distinguish between different samples types. Note also that, as mentioned in the previous part of this chapter, the scores of the second principal component in PCA distinguish between different sample types, but this distinguishing behavior is less apparent than that obtained from NMF.

Finally, we compare NMF and PCA in terms of the sparsity of the *loading* vectors stacked in matrix $\mathbf{U}$. Figure 2.7 depicts the loading corresponding to the first principal component obtained from the two methods. It is seen that the loadings are much sparser in NMF. It is a very helpful property; as it indicates that not only does NMF correctly cluster GBM samples together, but that it also extracts a small subset of genes that have similar expression levels among these samples and are grouped together. Currently, we are collaborating with researchers from the Singh lab at McMaster university to study the biological relevance of the extracted genes. This may shed important light on how genetic information influences the phenotype of cancerous and normal cells.

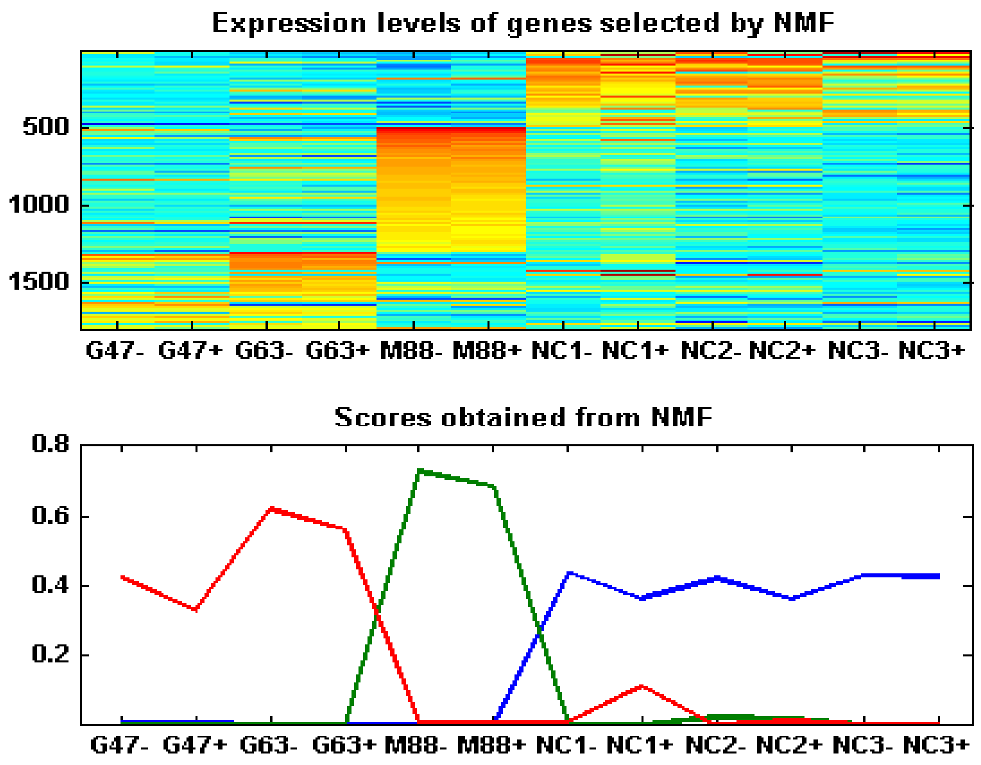
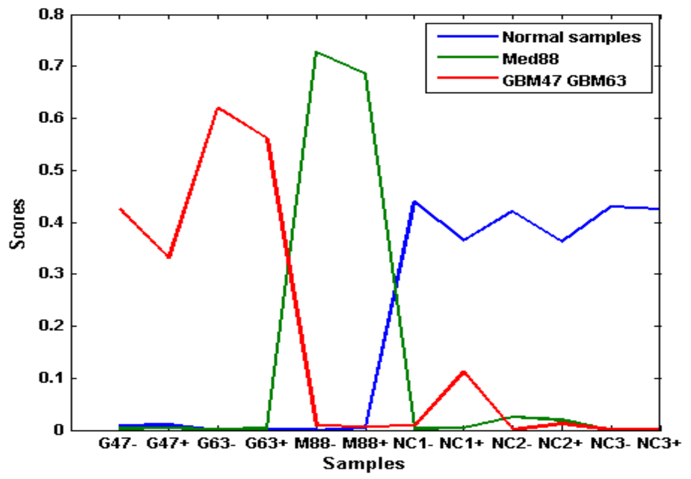


Figure 2.5: Non-negative matrix factorization of order 3 applied to cancer gene expression data. The top diagram shows the identified sample clusters and the corresponding extracted genes. The bottom diagram shows the columns of matrix **W** from which clustering of samples is inferred. In the above two diagrams, G47 stands for BT47_GBM, G63 stands for BT63_GBM, M88 stands for BT88_Medullo, NC1 stands for hNC1, NC2 stands for hNC2 and NC3 stands for hNC3.



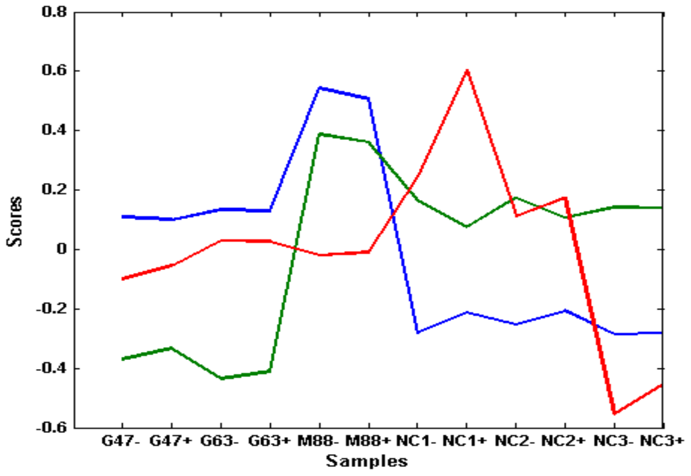


Figure 2.6: Scores obtained from NMF versus scores obtained from PCA. In the above two diagrams, G47 stands for BT47_GBM, G63 stands for BT63_GBM, M88 stands for BT88_Medullo, NC1 stands for hNC1, NC2 stands for hNC2 and NC3 stands for hNC3.

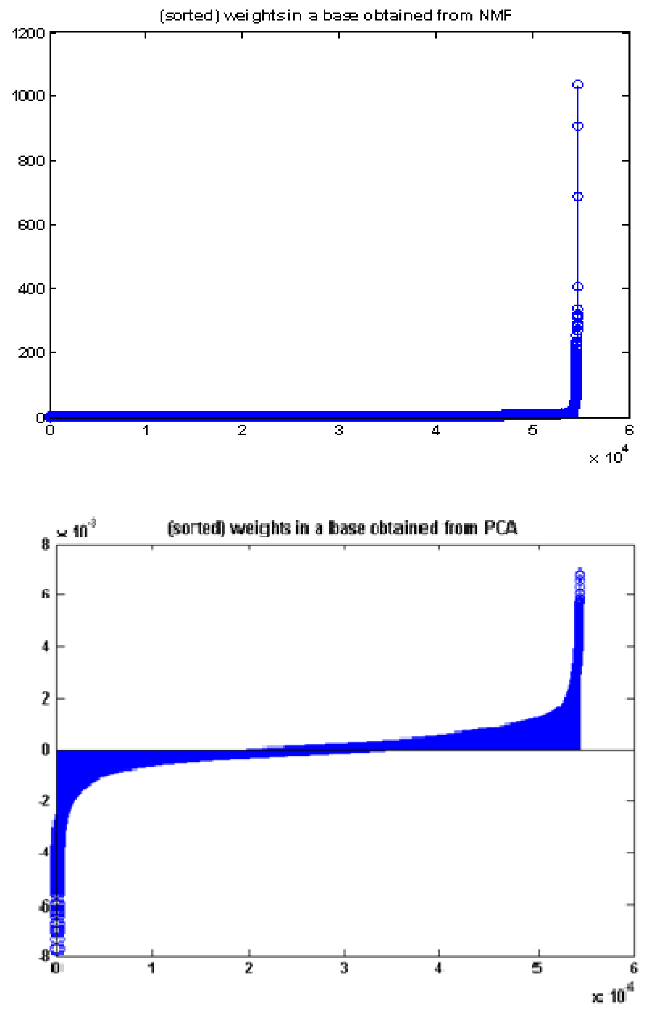


Figure 2.7: Loadings obtained from NMF versus scores obtained from PCA.

Chapter 3

Feature Selection Based on Hilbert-Schmidt Independence Criterion

3.1 Introduction

In the previous chapter, we introduced feature selection as an important task in the analysis of high dimensional data. We saw that a challenge in designing feature selection algorithms is the trade-off between accuracy and speed of the underlying algorithms. The wrapper and embedded algorithms are more accurate than the filter methods, at the expense of being time-consuming. The filter methods, on the other hand, are fast but they cannot take into account gene-to-gene interactions when extracting the important genes.

In this chapter, we propose a fast multivariate feature selection method for DNA microarrays based on the Hilbert-Schmidt Independence Criterion (HSIC) [23, 24]. HSIC provides a measure of dependence between two random variables. Once a set of observations (realizations) of the two random variables is given, one can estimate the value of HSIC based on the observations. This is discussed in subsection 3.2.1. HSIC has been already used to propose a family of gene selection algorithms, namely, Backward elimination HSIC (BAHSIC) family of algorithms, which encapsulates a number of well-known methods such as fold change, signal-to-noise ratio, Shrunken centroid and ridge regression ([42, 40]). The methods in BAHSIC family are mostly univariate techniques and thus does not take into account the interaction between features.

We use a completely different approach to accomplish feature selection based on HSIC, which is capable of takeing into account the interaction between features. We use HSIC in association with a fast technique for the sparse decomposition of matrices, namely Rank-one downdate (R1D) [7], to identify a sparse projection of the DNA microarray features that well represents the underlying response variables. Only a small subset of genes will have non-zero weights in the extracted projection vector and are thus recognized as the relevant genes with respect to the given response variables. The proposed algorithm is computationally very fast and scalable as storage of the whole microarray data in memory, is not required. Thus it can be easily applied to real-world large data sets.

3.2 Background of the Hilbert-Schmidt Independence Criterion

The Hilbert-Schmidt norm of the cross-covariance operator [23] in reproducing kernel Hilbert spaces (RKHS) has been proposed as an independence criterion. This measure will be referred to as the Hilbert-Schmidt independence criterion (HSIC). HSIC uses the fact that two random variables $\mathbf{x}$ and $\mathbf{y}$ are independent if and only if any bounded continuous function of the two random variables is uncorrelated. Consider two multivariate random variables $\mathbf{x}$ and $\mathbf{y}$ with joint probability distribution $p_{\mathbf{xy}}$. Let $\mathcal{X}$ and $\mathcal{Y}$ be the support (the set of possible values) of the random variables $\mathbf{x}$ and $\mathbf{y}$ respectively. Let $\mathcal{F}$ be a separable RKHS of real-valued functions from $\mathcal{X}$ to $\mathbb{R}$ with *universal*[1] kernel $k(\cdot,\cdot)$. Similarly, define $\mathcal{G}$ to be

[1]It is known that there is a unique correspondence between any kernel and the RKHS it reproduces. A kernel k is called *universal* if the corresponding RKHS, $\mathcal{F}$, includes all continuous functions on domain $\mathcal{X}$.

a separable RKHS of real-valued functions from $\mathcal{Y}$ to $\mathbb{R}$ with universal kernel $b(\cdot,\cdot)$. We are interested in the cross-covariance between elements of $\mathcal{F}$ and $\mathcal{G}$:

$$\operatorname{cov}(f(\mathbf{x}), g(\mathbf{y})) = \mathbb{E}_{\mathbf{x},\mathbf{y}}[f(\mathbf{x})g(\mathbf{y})] - \mathbb{E}_{\mathbf{x}}[f(\mathbf{x})]\mathbb{E}_{\mathbf{y}}[g(\mathbf{y})].$$

There exists a unique operator $C_{\mathbf{x},\mathbf{y}} : \mathcal{G} \to \mathcal{F}$ mapping elements of $\mathcal{G}$ to elements of $\mathcal{F}$ such that: $\langle f, C_{\mathbf{x},\mathbf{y}}(g)\rangle_{\mathcal{F}} = \operatorname{cov}(f, g)$ for all $f \in \mathcal{F}$ and $g \in \mathcal{G}$. This operator is called the cross-covariance operator [23].

The measure of dependence between two random variables can be defined as the squared Hilbert-Schmidt norm of the cross-covariance operator:

$$\operatorname{HSIC}(p_{\mathbf{xy}}, \mathcal{F}, \mathcal{G}) := \|C_{\mathbf{xy}}\|_{HS}^2.$$

Note that if $\|C_{\mathbf{xy}}\|_{HS}^2$ is zero, then the value of $\langle f, C_{\mathbf{x},\mathbf{y}}(g)\rangle$, i.e., $\operatorname{cov}(f, g)$, will always be zero for any $f \in \mathcal{F}$ and $g \in \mathcal{G}$, and thus the random variables $\mathbf{x}$ and $\mathbf{y}$ are independent. Now it becomes clear why kernels k and b need to be universal. That is because the corresponding RKHS, $\mathcal{F}$ and $\mathcal{G}$, have to include all continuous functions defined on supports $\mathcal{X}$ and $\mathcal{Y}$, respectively.

3.2.1 Empirical HSIC

To compute the HSIC we need to express it in terms of kernel functions. This can be achieved via the following identity [23]:

$$\begin{aligned} \operatorname{HSIC}(p_{\mathbf{xy}}, \mathcal{F}, \mathcal{G}) &= \mathbb{E}_{\mathbf{x},\mathbf{x}',\mathbf{y},\mathbf{y}'}[k(\mathbf{x},\mathbf{x}')b(\mathbf{y},\mathbf{y}')] + \mathbb{E}_{\mathbf{x},\mathbf{x}'}[k(\mathbf{x},\mathbf{x}')]\mathbb{E}_{\mathbf{y},\mathbf{y}'}[b(\mathbf{y},\mathbf{y}')] \\ &\quad -2\mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\mathbb{E}_{\mathbf{x}'}[k(\mathbf{x},\mathbf{x}')]\mathbb{E}_{\mathbf{y}'}[b(\mathbf{y},\mathbf{y}')]\right] \end{aligned}$$

where $\mathbb{E}_{\mathbf{x},\mathbf{x}',\mathbf{y},\mathbf{y}'}$ is expectation over $(\mathbf{x},\mathbf{y})$ and $(\mathbf{x}',\mathbf{y}')$ with $(\mathbf{x},\mathbf{y})$ and $(\mathbf{x}',\mathbf{y}')$ being random variables taken independently from $p_{\mathbf{xy}}$. Now let $\mathcal{Z} := \{(\mathbf{x}_1,\mathbf{y}_1),\cdots,(\mathbf{x}_n,\mathbf{y}_n)\} \subseteq \mathcal{X}\times\mathcal{Y}$ be a collection of n independent observations drawn from $p_{\mathbf{xy}}$. An estimator of HSIC is given by [23]:

$$\operatorname{HSIC}(\mathcal{Z}, \mathcal{F}, \mathcal{G}) := (n-1)^{-2}\operatorname{Tr}(\mathbf{KHBH}) \tag{3.1}$$

where $\mathbf{H}, \mathbf{K}, \mathbf{B} \in \mathbb{R}^{n \times n}, \mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j), \mathbf{B}_{ij} = b(\mathbf{y}_i, \mathbf{y}_j)$, $\mathbf{H} = \mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^{\mathrm{T}}$, k and b are positive semidefinite kernel functions, and $\mathbf{1}$ is a vector of ones. Based on this result, in order to maximize the dependence between two random variables $\mathbf{x}$ and $\mathbf{y}$, we need to increase the value of the empirical estimate, i.e. $\mathrm{Tr}(\mathbf{KHBH})$.

3.3 Feature selection via HSIC

Suppose $\mathbf{X} \in \mathbb{R}^{m \times n}$ represents genomic microarray data with m genes and n samples. Also assume $\mathbf{y}$ is a discrete or continuous response variable. For example, $\mathbf{y}$ could be labels of patients with high risk cancer (discrete) or their survival time (continuous) [2]. The goal is to select a small feature set that contains as much predictive information about the response $\mathbf{y}$ as possible. In other words, we are looking for a subset of features, such that $\mathbf{y}$ depends mainly on this subset and not on the rest of the features.

We address this problem as follows. Suppose there are n samples and n response values $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$ where $\mathbf{x}_i$ denotes the i^{th} column of matrix $\mathbf{X}$. We are looking for a projection $\mathbf{S} = \mathbf{u}^{\mathrm{T}}\mathbf{X}$ such that $\mathbf{y}$ depends mainly on $\mathbf{S}$. $\mathbf{u}^{\mathrm{T}}\mathbf{X}$ is a linear combination of all features where elements of $\mathbf{u}$ determine the importance, or the weight, of each feature. If $\mathbf{u}$ is a *sparse* vector, then the weight of some features are zero and the subset of features with corresponding nonzero weights will be the desired subset with maximum predictive information.

The dependence between the projection $\mathbf{u}^{\mathrm{T}}\mathbf{X}$ and the response variable $\mathbf{y}$ can be measured using the empirical estimation of HSIC given in (3.1). If $\mathbf{x}_1 \ldots \mathbf{x}_n$ are projected to $\mathbf{S} = [\mathbf{u}^{\mathrm{T}}\mathbf{x}_1 \ldots \mathbf{u}^{\mathrm{T}}\mathbf{x}_n] = \mathbf{u}^{\mathrm{T}}\mathbf{X}$, a linear kernel on subspace $\mathbf{S}$ can be computed as $\mathbf{X}^{\mathrm{T}}\mathbf{u}\mathbf{u}^{\mathrm{T}}\mathbf{X}$. Then $\mathbf{B}$ is a kernel of $\mathbf{y}$ and can be used to capture different types of prior information in the

[2]Here we assume the response variable is univariate. However, the results can be directly extended to the case of multivariate response variables. In such cases, the observations of the response variable is captured in matrix $\mathbf{Y}$ rather than the vector $\mathbf{y}$.

problem:

$$\begin{aligned}\mathrm{Tr}(\mathbf{HKHB}) &= \mathrm{Tr}(\mathbf{HX}^{\mathrm{T}}\mathbf{uu}^{\mathrm{T}}\mathbf{XHB}) \\ &= \mathrm{Tr}(\mathbf{u}^{\mathrm{T}}\mathbf{XHBHX}^{\mathrm{T}}\mathbf{u}) \qquad (3.2)\end{aligned}$$

We can make (3.2) arbitrarily large by increasing the magnitude of **u**. To make the problem well-posed we choose **u** to maximize (3.2) while constraining **u** to have unit length. On the other hand, to accomplish the feature selection task, **u** is required to be sparse. Based on these two constraints the optimization problem becomes

$$\begin{aligned}\max_{\mathbf{u}} \quad & \mathrm{Tr}(\mathbf{u}^{\mathrm{T}}\mathbf{XHBHX}^{\mathrm{T}}\mathbf{u}) \qquad (3.3)\\ \text{subject to} \quad & \mathbf{u}^{\mathrm{T}}\mathbf{u} = 1 \\ \text{subject to} \quad & \mathbf{u} \text{ is sparse}\end{aligned}$$

If we relax the sparsity constraint, this problem can be solved in closed-form. If the symmetric and real matrix $\mathbf{Q} = \mathbf{XHBHX}^{\mathrm{T}}$ has eigenvalues $\lambda_1 \leq \cdots \leq \lambda_n$ and eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$, then the maximum value of the cost function is λ_n and the optimal solution is $\mathbf{u} = \mathbf{v}_n$ [31].

Note that both **Q** and **B** are positive semidefinite matrices and thus we can define:

$$\begin{aligned}\mathbf{Q} &= \mathbf{AA}^{\mathrm{T}} \\ \mathbf{B} &= \boldsymbol{\Delta}^{\mathrm{T}}\boldsymbol{\Delta} \\ \mathbf{A} &= \mathbf{XH}\boldsymbol{\Delta}^{\mathrm{T}}\end{aligned}$$

Clearly the solution for **u** can be expressed as the first singular vector of **A**, since the singular vectors of **A** are the eigenvectors of $\mathbf{AA}^{\mathrm{T}} = \mathbf{XHBHX}^{\mathrm{T}}$.

3.3.1 Sparsity constraint

As noted earlier, **u** needs to be sparse. Otherwise we will not be able to locate important genes. A classical approach to add sparsity to **u** is to follow an approach similar to Lasso

[49] by adding the L_1 penalty $\sum_{i=1}^{n} |u_i|$ to the objective function. However, this leads to a rather computationally intensive problem which cannot be solved in closed form. Another approach is to set a threshold and keep only those elements in $\mathbf{u}$ that exceed the threshold. This is a common approach in the area of text mining and has been used extensively by methods such as latent semantic indexing (LSI) [12]. However, we believe this will not lead to an appropriate solution in our case. This can be seen in the following example:

Consider the following matrix $\mathbf{A}$, which is the sum of a completely separable matrix $\mathbf{F}$ and a noise matrix $\mathbf{E}$:

$$\begin{aligned} \mathbf{A} &= \mathbf{F} + \mathbf{E} \\ &= \begin{pmatrix} 1.01 & 1.01 & 0 & 0 \\ 1.01 & 1.01 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} + \begin{pmatrix} -0.02 & -0.02 & 0.02 & 0.02 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \end{aligned}$$

It should be clear that there are two separate sets of columns in $\mathbf{A}$, given by the two diagonal blocks, and a reasonable algorithm ought to be able to identify the two blocks. In other words, one would expect an answer close to $[1100]^{\mathrm{T}}$. Perhaps unexpectedly, the dominant right singular vector of $\mathbf{A}$ is very close to being proportional to $[1111]^{\mathrm{T}}$, which is different from what is normally expected. The reason for this behavior is that the matrix $\mathbf{F}$ has two nearly equal singular values, so its singular vectors are highly sensitive to small perturbations (such as the matrix $\mathbf{E}$). This pitfall can be avoided by computing a sparse singular vector that is used to factorize a submatrix of the original $\mathbf{A}$ instead of the whole matrix. Inspired by Rank-One Downdate (R1D) [7], we propose a new algorithm that finds the largest rank one submatrix of $\mathbf{A}$ and its singular vectors simultaneously. It therefore computes $[1100]^{\mathrm{T}}$ as the sparse singular vector of $\mathbf{A}$, as desired.

One of the classical algorithms for computing the first singular vectors of a matrix is the Power method [44]. Here we first review this algorithm and then show how it can be modified so as to extract a submatrix along with the optimal singular vectors. As presented below, the Power method takes an $m \times n$ matrix $\mathbf{A}$ as input and returns its first singular vectors $\mathbf{u} \in \mathbb{R}^{m\times 1}$ and $\mathbf{v} \in \mathbb{R}^{n\times 1}$ as well as the corresponding singular value $\sigma \in \mathbb{R}$. It is known that the factorization $\mathbf{u}\sigma\mathbf{v}^{\mathrm{T}}$ provides the best rank-one approximation to $\mathbf{A}$ in either Frobenius norm or 2-norm.

```
Algorithm $[\mathbf{u}, \sigma, \mathbf{v}] = \texttt{PowerMethod}(\mathbf{A})$
Input: $\mathbf{A} \in \mathbb{R}^{m\times n}$.
Output: $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{v} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}$
1.  Select a random nonzero $\bar{\mathbf{u}} \in \mathbb{R}^m$.
2.  $\sigma = \|\bar{\mathbf{u}}\|$.
3.  $\mathbf{u} = \bar{\mathbf{u}}/\sigma$.
4.  repeat
5.      $\bar{\mathbf{v}} = \mathbf{A}^{\mathrm{T}}\mathbf{u}$.
6.      $\mathbf{v} = \bar{\mathbf{v}}/\|\bar{\mathbf{v}}\|$.
7.      $\bar{\mathbf{u}} = \mathbf{A}\mathbf{v}$.
8.      $\sigma = \|\bar{\mathbf{u}}\|$.
9.      $\mathbf{u} = \bar{\mathbf{u}}/\sigma$.
10. until stagnation in $\mathbf{u}, \sigma, \mathbf{v}$.
```

Our proposed method, which we call the Sparse Power Method (`SparsePM`), is similar to the classical Power method. It provides a rank one approximation to the original matrix $\mathbf{A}$. The output of our algorithm, however, includes a submatrix of the original matrix which is approximately rank-one as well as the corresponding optimal singular vectors used to factorize this submatrix. The `SparsePM` algorithm and its parameters may be briefly described as follows.

$[M, N, \mathbf{u}, \sigma, \mathbf{v}] = \texttt{SparsePM}(\mathbf{A})$

Inputs: $\mathbf{A} \in \mathbb{R}^{m\times n}$.

Outputs: M, N, $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{v} \in \mathbb{R}^n$ and $\sigma \in \mathbb{R}$.

Here, $M \subset \{1, \ldots, m\}$ and $N \subset \{1, \ldots, n\}$ specify the indices of the extracted submatrix. In addition, $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{v} \in \mathbb{R}^n$ are sparse unit vectors whose nonzero elements are indexed by the subsets M and N, respectively. In this chapter, we use subscripts to denote submatrices (sub-vectors) of a matrix (a vector). For example, $\mathbf{A}_{M,N}$ denotes an $|M| \times |N|$ submatrix of $\mathbf{A}$ with rows and columns in sets M and N, respectively. Also, $\mathbf{u}_M$ denotes an $|M| \times 1$ vector consisting of elements of $\mathbf{u}$ indexed by M. Here, $|\cdot|$ denotes cardinality of a set.

The function `SparsePM` selects the submatrix $\mathbf{A}_{M,N}$ such that it is approximately rank one, and in particular approximately equal to $\mathbf{u}_M \sigma \mathbf{v}_N^{\mathrm{T}}$. In this way, it provides a sparse rank-one approximation to the original matrix $\mathbf{A}$. Specifically, the function `SparsePM` is de signed to maximize the following objective function:

$$f(M, N, \mathbf{u}, \sigma, \mathbf{v}) = \|\mathbf{A}_{M,N}\|_F^2 - \gamma \|\mathbf{A}_{M,N} - \mathbf{u}_M \sigma \mathbf{v}_N^{\mathrm{T}}\|_F^2 - \rho |M|\,|N| \tag{3.4}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix, and γ and ρ are penalty parameters. The first term in the above objective function favors submatrices with large norm. Such submatrices provide better approximations to the original matrix $\mathbf{A}$, and are less likely to be produced by noise. The second term in 3.4 penalizes deviations from being rank-one, and the third term ensures that the size of the extracted submatrix is not too large. This is particularly relevant in the feature selection applications where we do not want the number of selected features to be too large.

Maximization of the objective function (3.4) is conjectured to be NP-hard [7], however it can be optimized using a heuristic procedure. To this end, we update the values of parameters $\mathbf{u}$, M, $\mathbf{v}$ and N in a periodic manner. First assume the values of $\mathbf{v}$, and thus N, is given. It can be shown that the objective function is separable in terms of rows (as well as columns) of matrix $\mathbf{A}$. In particular, the contribution of the i^{th} row is given by:

$$r_i = \|\mathbf{A}_{i,N}\|^2 - \gamma \|\mathbf{A}_{i,N} - \beta_i \mathbf{v}_N^{\mathrm{T}}\|^2 - \rho |N| \tag{3.5}$$

where $\beta_i = u_i \sigma$ and u_i is the i^{th} element of $\mathbf{u}$. Due to this property, one can consider the contribution of the rows separately and choose only those rows that can potentially make positive contribution to the objective function. The values of the first and third terms in 3.5 is fixed when a row is given. One can solve a simple least squares problem to show that the value of the second term is maximized (the values of $\|\mathbf{A}_{i,N} - \beta_i \mathbf{v}_N^{\mathrm{T}}\|^2$ is minimized) when β_i is set to $\mathbf{A}_{i,N}\mathbf{v}_N$. Based on this the contribution of the i^{th} row would be:

$$r_i = \|\mathbf{A}_{i,N}\|^2 - \gamma\|\mathbf{A}_{i,N} - \mathbf{A}_{i,N}\,\mathbf{v}_N\mathbf{v}_N^{\mathrm{T}}\|^2 - \rho|N|. \tag{3.6}$$

Now one can evaluate the above expression for all rows and select only those whose contribution could be positive. This provides us with the set M. The nonzero elements of vector $\mathbf{u}$ and the values of σ can be obtained simply by normalizing the vector consisting of β_i, $i \in M$.

The expression presented in 3.6 can be further simplified to:

$$\begin{aligned} r_i &= \mathbf{A}_{i,N}\mathbf{A}_{i,N}^{\mathrm{T}} \\ &\quad -\gamma\left(\mathbf{A}_{i,N} - \mathbf{A}_{i,N}\,\mathbf{v}_N\mathbf{v}_N^{\mathrm{T}}\right)\left(\mathbf{A}_{i,N} - \mathbf{A}_{i,N}\,\mathbf{v}_N\mathbf{v}_N^{\mathrm{T}}\right)^{\mathrm{T}} - \rho|N| \\ &= -(\gamma-1)\mathbf{A}_{i,N}\mathbf{A}_{i,N}^{\mathrm{T}} + \gamma(\mathbf{A}_{i,N}\mathbf{v}_N)^2 - \rho|N|. \end{aligned}$$

A similar analysis may be applied to the columns. The result is that for given values of M and $\mathbf{u}$, the column j should be accepted provided that $c_j > 0$, where,

$$c_j = -(\gamma-1)\mathbf{A}_{M,j}^{\mathrm{T}}\mathbf{A}_{M,j} + \gamma(\mathbf{A}_{M,j}^{\mathrm{T}}\mathbf{u}_M)^2 - \rho|M|.$$

The separability characteristics of the objective function (3.4) (in terms of rows and columns) not only provides an easy way to implement the above algorithm but also leads the proposed technique to be scalable to large data sets. That means one does not need to store the whole microarray data in memory when running the algorithm; we only need to deal with the one row (or column) of data which is being used in th operation. This feature proves particularly important when dealing with large real-world data sets.

Finally, we need to decide about initial values of parameters $M, N, \mathbf{u}, \sigma, \mathbf{v}$. These starting values should be chosen such that the initial value of the objective function (3.4) is positive. Otherwise, it is possible that no row (or column) will be selected in the subsequent steps.

We select the row which has the greatest norm as the initial value of $\mathbf{v}$, and start iterations to find M. Alternatively, one could select the column with the largest norm as the initial value of $\mathbf{u}$ and proceed to find N and $\mathbf{v}$. Thus we have derived the following algorithm for the subroutine `SparsePM`:

```
**Algorithm** $[M, N, \mathbf{u}, \mathbf{v}, \sigma] = \texttt{SparsePM}(\mathbf{A})$
**Input:** $A \in \mathbb{R}^{m\times n}$, parameter $\gamma$.
**Output:** $M \subset \{1,\dots,m\}$, $N \subset \{1,\dots,n\}$, $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{v} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}$.
1.  Select $i_0 \in \{1,\dots,m\}$ to maximize $\|\mathbf{A}_{i_0,:}\|$.
2.  $N = \{1,\dots,n\}$.
3.  $M = \{i_0\}$.
4.  $\sigma = \|\mathbf{A}_{i_0,:}\|$.
5.  $\mathbf{v} = \mathbf{A}_{i_0,:}/\sigma$.
6.  **repeat**
7.      Let $\bar{\mathbf{u}} = \mathbf{A}_{:,N}\mathbf{v}_N$.
8.      $M = \{i \ : \ -(\gamma-1)\mathbf{A}_{i,N}\mathbf{A}_{i,N}^{\mathrm{T}} + \gamma(\mathbf{A}_{i,N}\mathbf{v}_N)^2 - \rho|N| > 0\}$.
9.      $\mathbf{u}_M = \bar{\mathbf{u}}_M/\|\bar{\mathbf{u}}_M\|$.
10.     Let $\bar{\mathbf{v}} = \mathbf{A}_{M,:}^{\mathrm{T}}\mathbf{u}_M$.
11.     $N = \{j \ : \ -(\gamma-1)\mathbf{A}_{M,j}^{\mathrm{T}}\mathbf{A}_{M,j} + \gamma(\mathbf{A}_{M,j}^{\mathrm{T}}\mathbf{u}_M)^2 - \rho|M| > 0\}$.
12.     $\sigma = \|\bar{\mathbf{v}}_N\|$.
13.     $\mathbf{v}_N = \bar{\mathbf{v}}_N/\sigma$.
14. **until** stagnation in $M, N, \mathbf{u}, \sigma, \mathbf{v}$.
```

The value of the objective function is increased at each iteration in the above algorithm; so the convergence of the algorithm is guaranteed. In practice, we observe the algorithm converges within a few iterations, leading the underlying feature selection method to be fast. The entire feature selection algorithm can be expressed as follows:

```
Algorithm $M =$ FeatureSelection$(\mathbf{X}, \mathbf{y})$
Input: $\mathbf{X} \in \mathbb{R}^{m \times n}$ and $\mathbf{y} \in \mathbb{R}^n$.
Output: $M \subset \{1, 2, \cdots, m\}$.
1.  Compute $\mathbf{B} = b\{y_i, y_j\} = \mathbf{\Delta}^{\mathrm{T}}\mathbf{\Delta}$
2.  Compute $\mathbf{A} = \mathbf{XH\Delta}^{\mathrm{T}}$
3.  $[M, N, \mathbf{u}, \mathbf{v}, \sigma] =$ SparsePM$(\mathbf{A})$
4.  return $M$
```

3.4 Experimental results

3.4.1 Synthetic data

To evaluate the performance of our algorithm, we first consider an experiment on synthetic data. Assume we have a set of 50 data points $\{\mathbf{x}_i\}_{i=1}^{50}$ each consisting of 60 features, stacked in a 60×50 matrix $\mathbf{X}$. We construct a univariate response variable $\mathbf{y}$ which depends only on a specific subset of the features, as follows:

$$\mathbf{y} = \sin(\mathbf{X}_{5,:}) + \sin(\mathbf{X}_{10,:}) + \mathbf{X}_{15,:} \odot \mathbf{X}_{15,:} + \epsilon$$

where $\epsilon \sim N(0, 0.01)$ is Normally-distributed additive i.i.d. noise and $\mathbf{X}_{i:}$ denotes the ith feature (the ith row of $\mathbf{X}$), and $\odot$ stands for the elementwise product between two vectors (also known as Hadamard product). Note that there is a relatively complicated relationship between the response variable $\mathbf{y}$ and the features of $\mathbf{X}$. We produce elements of the data points $\mathbf{X}$ according to the unit uniform distribution. A linear kernel is used for the response variable $\mathbf{y}$ in this example. We have arbitrarily used the value $\rho \approx 0.015$ which results in a reasonable number of selected features. One may see that the presented results are not very sensitive to this parameter. Furthermore, the value of the parameter γ is always set to a default value of 1.1 and this is kept constant throughout this chapter. We repeat the process 100 times to explore different possibilities for the variables $\mathbf{X}$ and $\mathbf{y}$. We observed that the true features (5, 10 and 15) were selected by our method in all 100 trials. However, all of the other features occurred in at most 3% of the trials.

3.4.2 Gene expression data

In this section, we present the experimental results on a number of well-known DNA microarray data sets. For all the classification problems in this work we use the Leave One Out (LOO) classification correction rate to evaluate the performance of feature selection methods. To apply our proposed algorithm, the label matrix $\boldsymbol{\Delta}$ is chosen to be a linear indicator matrix whose elements are normalized by the cardinal of the corresponding data classes. For the classification task, we apply a linear SVM classifier subsequent to our feature selection algorithm. For other feature selection techniques we use the same classifiers as those used in the original references. The results are presented both on the two-class and multi-class data sets. For the two-class case we study four data sets, namely, Leukemia [22], Colon cancer [3], Lymphoma [2] and Prostate [38]. The results for the two-class data sets are summarized in Table 3.1. In this case, we compare our feature selection method with four standard techniques, as presented in Table 3.1. In this table our method is represented by SHS-SVM which stands for Sparse Hilbert-Schmidt independence criterion based feature selection technique followed by an SVM classifier. In addition, Golub-Golub denotes the feature selection and classification methods introduced in [22]. RFE-SVM and L0-SVM represents the RFE-SVM and AROM-SVM feature selection techniques introduced in the first chapter followed by an SVM classifier. Finally, FC-SVM uses a simple Fold Change criterion to filter important genes and then the extracted genes would be fed into an SVM classifier. We have used the implementations of RFE and L0 feature selection techniques presented in the SPIDER toolbox [52].

To evaluate the performance of our proposed feature selection algorithm on multi-class problems, we perform experiments on three data sets: Lung cancer [6], SRBCT [27], and 11 tumor [45]. We follow our proposed feature selection method with a linear one-vs-one SVM classifier taken from LIBSVM [8]. We compare our method with the baseline BSS/WSS

	Data Set			
Method	Leuk.	Colon	Lymph.	Prostate
SHS-SVM	98.61%	85%	96.88%	95.1%
Golub-Golub	98.61%	87.1%	92.71%	95.1%
RFE-SVM	98.61%	85.48%	96.88%	95.1%
L0-SVM	98.61%	82.26%	96.88%	95.1%
FC-SVM	98.61%	85.48%	96.88%	95.1%

Table 3.1: Experimental results on two-class data sets. The best LOO classification correction rate is presented for each feature selection technique.

	Data Set		
Method	Lung cancer	SRBCT	11 tumors
SHS-SVM	94.09%	100.0%	90.3%
BSS/WSS-SVM	94.09%	100.0%	93.1%

Table 3.2: Experimental results on multi-class data sets. Again, the best LOO classification correction rate is presented for each feature selection method.

filtering method [16]. This is a univariate technique that uses the ratio of Between-class to Within-class Sum of Squares to rank the genes. We apply the same multi-class SVM classifier to this feature selection technique as was proposed in the SHS-SVM method. The results of the experiment on multi-class data sets are presented in Table 3.2. Recall that our proposed feature selection method does not use a fixed number of genes for each separation of the test/training data, as opposed to other feature selection methods. Because of this, a comparison between these techniques cannot be made in any straightforward way. To make a fair comparison, for each data set, we report the best LOO correction rate obtained by each method on each of the data sets. To do so, in our method, we change the value of the parameter ρ so that the number of genes vary between 1 and the total number of genes. On the other hand, for other feature selection techniques, we manually vary the number of genes between 1 and the total number of genes. In both cases, we report the best classification correction rate that was obtained. In all tests performed here (both two-class and multi-class cases) the optimal (varying or fixed) number of genes was only a small fraction (always less than half) of the total number of genes, which is an indicator of the feature selection task. One may note that the performance of our proposed method is comparable to the well-established feature selection techniques used in the literature. Indeed, considering the fact that the variance of the classification correction rates is relatively high in DNA microarray experiments (we observed an average of 10% variance in our experiments), the difference between the reported classification rates are considered to be statistically insignificant based on the standard statistical significant tests. On the other hand, unlike other multi-gene feature selection techniques, our method has the virtues of being fast and scalable to large data sets.

Chapter 4

A Unified View of Learning Algorithms Based on Hilbert-Schmidt Independence Criterion

In the first chapter, we introduced a number of widely used techniques for the analysis of gene expression data. They include methods for dimensionality reduction, clustering and classification of data. Later in chapter three, we proposed a feature selection method based on an independence criterion, namely, the Hilbert-Schmidt independence criterion (HSIC). In this chapter, we will see that HSIC can be indeed used to explain a wide range of techniques in machine learning including dimensionality reduction, clustering and metric learning methods.

There are some works in the literature that aim to provide unified frameworks for some learning algorithms. Ham, et al. [33] suggested that a number of well-known dimensionality reduction techniques may be thought of as a special case of the Kernel Principle Component Analysis (KPCA). Dhillon, et al. [13] showed that different clustering algorithms, including k-means, Ratio Cut and Normalized Cut Spectral Clustering, can be cast as a special case of weighted kernel k-means. In [41], Song, et al. presented a dependence view of k-means clustering based on HSIC. In this chapter, we describe a framework that can encompass all these descriptions. Also, we show the capability of this framework in explaining a number of new machine learning techniques. In particular, we show that two successful metric learning techniques, namely Fisher's Discriminant Analysis (FDA) [17] and Closed Form Metric Learning (CFML) [1], can be formulated in the context of our proposed framework. In addition, inspired by the introduced general framework, we derive a new method for distant metric learning. The new technique uses the concepts from normalized cut spectral clustering and can be formulated in terms of a convex optimization problem.

4.1 General Framework

Let $\mathbf{x}$ and $\mathbf{y}$ be two random variables with joint distribution $p_{\mathbf{xy}}$ and supports $\mathcal{X}$ and $\mathcal{Y}$, respectively. Suppose $\mathcal{Z} := \{(\mathbf{x}_1, \mathbf{y}_1), \cdots, (\mathbf{x}_n, \mathbf{y}_n)\} \subseteq \mathcal{X} \times \mathcal{Y}$ is a collection of n independent observations drawn from $p_{\mathbf{xy}}$. As stated in Section 3.2.1, an empirical estimator of HSIC is given by

$$\mathrm{HSIC}(\mathcal{Z}) := (n-1)^{-2} \operatorname{Tr}(\mathbf{KHBH}) \tag{4.1}$$

where $\mathbf{H}, \mathbf{K}, \mathbf{B} \in \mathbb{R}^{n \times n}, \mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j), \mathbf{B}_{ij} = b(\mathbf{y}_i, \mathbf{y}_j),\ \mathbf{H} = \mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^{\mathrm{T}},\ k$ and b are positive semidefinite kernel functions, and $\mathbf{1}$ is a vector of ones. In order to maximize the dependence between two random variables $\mathbf{x}$ and $\mathbf{y}$, we need to increase the value of the empirical estimate, i.e. $\operatorname{Tr}(\mathbf{KHBH})$. As mentioned in the previous chapter, matrices $\mathbf{K}$ and $\mathbf{B}$ usually represent kernels of data and labels, respectively.

In some applications, the labels are already given as some sort of side-information about the data (e.g. metric learning). In other applications (e.g. clustering and dimensionality reduction), the labels are unknown and the goal is to learn labels based on data. In the latter case, the unknown labels might be categorical (clustering) or continuous (dimensionality reduction). In the following sections, we consider each of the above techniques in more detail.

4.2 Dimensionality Reduction Techniques

Let $\mathbf{X}_{m\times n} = [\mathbf{x}_1, \dots, \mathbf{x}_n]$ be a data matrix consisting of n points in the $\mathbb{R}^m$ space. Also let matrix $\mathbf{G}_{d\times n}$=$[\mathbf{g}_1, \dots, \mathbf{g}_n]$ be an alternative representation of the data consisting of n points in $\mathbb{R}^d$. In dimensionality reduction techniques, d is assumed to be much less than m, and the goal is to find a real-valued matrix $\mathbf{G}$ such that it is an appropriate *low-dimensional* representation of the data.

As mentioned earlier, it is known that many widely used dimensionality reduction algorithms can be represented using the framework of kernel PCA [33]. The covered methods include metric multidimensional scaling (MDS) [11], Isomap [47, 48], locally linear embedding (LLE) [35, 37] and Laplacian eigenmap (LEM) [5]. On the other hand, it is known [41] that kernel PCA can be reproduced by maximizing the empirical estimate of HSIC between a kernel of $\mathbf{X}$ and a *linear* kernel of a low-dimensional representation $\mathbf{G}$. Based on this, many dimensionality reduction techniques can be recovered using the formulation given in 4.1. The formulation, in this special case, reproduces kernel PCA as follows:

$$\max \ \mathrm{Tr}\left[\mathbf{HKH}\,\mathbf{G}^{\mathrm{T}}\mathbf{G}\right] = \mathrm{Tr}\left[\mathbf{G}(\mathbf{HKH})\mathbf{G}^{\mathrm{T}}\right] \tag{4.2}$$

In kernel PCA, the optimization problem presented in 4.2 is solved subject to $\mathbf{G}\mathbf{G}^{\mathrm{T}} = \mathbf{I}$, where $\mathbf{G}$ could be any real-valued matrix of size $d \times n$ [1]. No other constraint is imposed on $\mathbf{G}$ in this formulation. As will be seen in the subsequent sections, one needs to impose some extra constraints on the structure of $\mathbf{G}$ when reproducing clustering and metric learning techniques.

By appropriately choosing the value of the kernel matrix $\mathbf{K}$ in 4.2, one can recover different dimensionality reduction techniques. In the following, we briefly introduce the aforementioned dimensionality reduction methods, namely, MDS, Isomap, LLE and LEM; and present the kernels that can be used to reproduce these methods based on the formulation given in 4.2.

[1]If the linear kernel $\mathbf{K} = \mathbf{X}^{\mathrm{T}}\mathbf{X}$ is used in 4.2, the low-dimensional representation represented by $\mathbf{G}$ would be identical to the low-dimensional representation given by direct PCA up to scaling factors [19].

LLE: In nonlinear dimensionality reduction methods, including LLE, the goal is to learn a low-dimensional neighborhood preserving embedding (manifold) of high-dimensional data. Suppose the data in the high-dimensional space lie on or are close to a (generally nonlinear) manifold of dimensional d. In LLE, we assume that each point and its nearest neighbors lie on, or are close to, a locally linear patch of the manifold. To determine the nearest neighbors of a given data point, we can choose a fixed number, say k, of its nearest neighbors in Euclidian space; or choose all points within some fixed radius ϵ of that given data point. At the next step, we obtain the optimal weights by which each data point can be reconstructed from its nearest neighbors in the original space. This can be done by solving the following optimization problem:

$$\min_{\mathbf{W}} \sum_{i=1}^{n} \|\mathbf{x}_i - \sum_{j=1}^{k} W_{ij}\, \mathbf{x}_{N_i(j)}\|^2 \qquad \text{s.t.} \;\; \sum_{j=1}^{k} W_{ij} = 1, \;\; 1 \leq i \leq n.$$

where $N_i(j)$ denotes the index of the j^{th} neighbor of the i^{th} point, and W_{ij} denoted the contribution of the j^{th} data point in reconstructing the i^{th} data point. Of course, the (i, j)-th element of the weight matrix $\mathbf{W}$ equals zero whenever the j^{th} data point is not a neighbor of the i^{th} data point. At the last step, LLE selects data points $\mathbf{g}_1, \cdots, \mathbf{g}_n$ in the low-dimensional space so as to preserve the reconstruction weights obtained above. Mathematically speaking, it solves

$$\min_{\mathbf{G}} \sum_{i=1}^{n} \|\mathbf{g}_i - \sum_{j=1}^{k} W_{ij}\mathbf{g}_{N_i(j)}\|^2$$

This objective function can be expressed as [35]:

$$\min_{\mathbf{G}} \operatorname{Tr}(\mathbf{G}\mathbf{L}\mathbf{G}^{\mathrm{T}}) \tag{4.3}$$

where $\mathbf{L} = (\mathbf{I}-\mathbf{W})^{\mathrm{T}}(\mathbf{I}-\mathbf{W})$. The solution for $\mathbf{G}$ can have arbitrary origin and orientation. To make the problem well-posed, the optimization problem in 4.3 is solved subject to constraints $\sum_{i=1}^{n} \mathbf{g}_i = \mathbf{0}$ and $\mathbf{G}\mathbf{G}^T = \mathbf{I}$. Matrix $\mathbf{L}$ in 4.3 always has an eigenvalue equal to zero, and the corresponding eigenvector is a vector of all ones, denoted by $\mathbf{1}$. Using the Rayleigh-Ritz theorem (see e.g. [31]), one can see that the solution to the optimization problem in 4.3 subject to the constraints proposed above is given by the eigenvectors corresponding to the lowest eigenvalues of $\mathbf{L}$, excluding the trivial eigenvector $\mathbf{1}$. To express LLE in terms of the formulation given in 4.2 (which, as mentioned before, is the formulation of kernel PCA), one needs to express the optimization problem in 4.3 as a maximization problem. This can be done in two ways. The first one is by using the pseudo inverse of matrix $\mathbf{L}$, which we denote by $\mathbf{L}^{\dagger}$. Using the Rayleigh-Ritz theorem, and the fact that the eigenvalues of the pseudo inverse of a given matrix are equal to the inverse of the eigenvalues of the original matrix, one can show that the solution to LLE can be reproduced by solving

$$\max_{\mathbf{G}} \mathrm{Tr}(\mathbf{G}\mathbf{L}^{\dagger}\mathbf{G}^{\mathrm{T}}) \tag{4.4}$$

Note that matrix $\mathbf{L}$, and thus its pseudo inverse $\mathbf{L}^{\dagger}$, are positive semidefinite matrices. So we can set the value of the kernel matrix $\mathbf{K}$ in 4.2 to $\mathbf{L}^{\dagger}$, and solve the resulting optimization problem subject to $\mathbf{G}\mathbf{G}^{\mathrm{T}} = \mathbf{I}$. This provides us with the desired equivalent optimization problem expressed in terms of the formulation given in 4.2.

Another way to express the optimization problem presented in 4.3 as an equivalent maximization problem, is the following:

$$\max_{\mathbf{G}} \mathrm{Tr}(\mathbf{G}(\lambda_{\max}\mathbf{I} - \mathbf{L})\mathbf{G}^{\mathrm{T}}) \tag{4.5}$$

Here, λ_{max} is the absolute value of the largest negative eigenvalue of $\mathbf{L}$. The term $\lambda_{max}\mathbf{I}$ in 4.5 is added to make the expression $\lambda_{max}\mathbf{I} - \mathbf{L}$ positive semidefinite. Note that:

$$\mathrm{Tr}[\mathbf{G}(\lambda_{max}\mathbf{I} - \mathbf{L}^{\dagger})\mathbf{G}^{\mathrm{T}}] = \mathrm{Tr}[\mathbf{G}(\lambda_{max}\mathbf{I})\mathbf{G}^{\mathrm{T}}] + \mathrm{Tr}[\mathbf{G}\mathbf{L}^{\dagger}\mathbf{G}^{\mathrm{T}}] = d\lambda_{max} + \mathrm{Tr}[\mathbf{G}\mathbf{L}^{\dagger}\mathbf{G}^{\mathrm{T}}]$$

and thus the solution to 4.5 is identical to the solution to LLE in 4.3.

MDS: In MDS, the idea is to choose points in the low-dimensional space such that the pairwise distances between the points in this space are as close as possible to the pairwise distances between the points in the original space. Mathematically speaking, MDS aims to minimize the following objective function:

$$\min_{\mathbf{G}} \sum_{i=1}^{n}\sum_{j=1}^{n}(d_{ij}^{(X)} - d_{ij}^{(G)})^2$$

where $d_{ij}^{(X)} = \|\mathbf{x}_i - \mathbf{x}_j\|$ and $d_{ij}^{(G)} = \|\mathbf{g}_i - \mathbf{g}_j\|$ are Euclidian distances between points in the original and the low-dimensional space, respectively. Let $\mathbf{D}^{(X)}$ denote the distance matrix whose (i, j)-th element is equal to $d_{ij}^{(X)}$. It can be shown [11] that the solution to the above optimization problem is given by $\mathbf{G} = \mathbf{\Lambda}^{1/2}\mathbf{V}^{\mathrm{T}}$ where $\mathbf{V}$ represents the eigenvectors of matrix $-\frac{1}{2}\mathbf{H}\mathbf{D}^{(X)}\mathbf{H}$ corresponding to the top d eigenvalues, and $\mathbf{\Lambda}$ is a diagonal matrix consisting of the top d eigenvalues of $-\frac{1}{2}\mathbf{H}\mathbf{D}^{(X)}\mathbf{H}$. So MDS can be reproduced using the formulation given in 4.2 up to scaling factors $\mathbf{\Lambda}^{1/2}$.

Isomap: Isomap is very similar to MDS. The only difference is that in evaluating the distance matrix $\mathbf{D}^{(X)}$ it uses geodesic distance between points instead of the Euclidian distance. The geodesic distance between points is obtained based on an auxiliary graph. The auxiliary graph has n vertices corresponding to the n data points. Each data point is connected to its nearest data points via an edge. The geodesic distances between two data points is given by the shortest path between the corresponding vertices on the graph. Again, to determine the nearest neighbors of a given data point one can choose a fixed number, say k, of its nearest neighbors in the Euclidian space; or choose all points within some fixed radius ϵ of that data point.

LEM: In LEM, we first need to have a similarity matrix $\mathbf{W}$ whose (i, j)-th entry captures a measure of similarity between the j^{th} and i^{th}data points. A widely used measure is the Gaussian measure defined as $W_{ij} = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2})$. Then LEM selects points $\mathbf{g}_1, \cdots, \mathbf{g}_n$ in the low-dimensional space such that the following objective function is minimized:

$$\min_{\mathbf{G}} \sum_{i=1}^{n} \sum_{j=1}^{n} \|\mathbf{g}_i - \mathbf{g}_j\|^2 W_{ij} \tag{4.6}$$

Optimizing the above objective function results in the data points with high similarity lying close to each other. One may see that this objective function can be expressed as follows:

$$\min_{\mathbf{G}} \mathrm{Tr}(\mathbf{G}\mathbf{L}\mathbf{G}^{\mathrm{T}}) \tag{4.7}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$, $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_{j=1}^{n} W_{ij}$. Matrix $\mathbf{L}$ defined in this way is called the *Laplacian* matrix. Again, to account for the degrees of freedom due to location and orientation of data, one can solve the optimization problem in 4.7 subject to $\mathbf{G}\mathbf{G}^{\mathrm{T}} = \mathbf{I}$. Again like LLE, one can use either $\mathbf{L}^{\dagger}$ or $\lambda_{max}\mathbf{I} - \mathbf{L}$ to obtain a maximization problem. This allows us to express LEM in terms of the formulation given in 4.2.

A *normalized* form of LEM may also be proposed as follows. In this form, the goal is to minimize the following objective function:

$$\min_{\mathbf{G}} \sum_{i=1}^{n} \sum_{j=1}^{n} \| \frac{\mathbf{g}_i}{\sqrt{D_{ii}}} - \frac{\mathbf{g}_j}{\sqrt{D_{jj}}} \|^2 W_{ij} \tag{4.8}$$

Similar to normal LEM the above objective function can be expressed in the trace form. To do so, we can replace $\mathbf{G}$ by $\mathbf{G}\mathbf{D}^{-\frac{1}{2}}$ in 4.7 to arrive at:

$$\min_{\mathbf{G}} \ \mathrm{Tr}[\mathbf{G}\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\mathbf{G}^{\mathrm{T}}]$$

The above objective function can be simplified as follows:

$$\begin{aligned}
\mathrm{Tr}[\mathbf{G}\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\mathbf{G}^{\mathrm{T}}] &= \mathrm{Tr}[\mathbf{G}\mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}}\mathbf{G}^{\mathrm{T}}] \\
&= \mathrm{Tr}[\mathbf{G}(\mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})\mathbf{G}^{\mathrm{T}}] \\
&= \mathrm{Tr}[\mathbf{G}\mathbf{G}^{\mathrm{T}}] - \mathrm{Tr}[\mathbf{G}(-\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})\mathbf{G}^{\mathrm{T}}] \\
&= d - \mathrm{Tr}[\mathbf{G}(\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})\mathbf{G}^{\mathrm{T}}]
\end{aligned}$$

Based on this, one can write the optimization problem in 4.8 as a maximization problem as follows:

$$\max_{\mathbf{G}} \ \mathrm{Tr}[\mathbf{G}(\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})\mathbf{G}^{\mathrm{T}}]$$

Now we can use a kernel of the form $\lambda_{max}\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$ to recover normalized LEM in terms of the formulation given in 4.2. In this formulation, λ_{max} is the absolute value of the largest negative eigenvalue of $\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$, and helps to obtain a positive semidefinite kernel in formulation 4.2.

The above results are summarized in Table 4.1. It is interesting to note that the kernels introduced for MDS, Isomap and kernel $\mathbf{L}^{\dagger}$ for LLE and LEM are already double centered and the application of centering matrices $\mathbf{H}$ in equation (4.2) leaves them unchanged. Also, the centering effect of $\mathbf{H}$ on the kernels $\lambda_{max}\mathbf{I}-\mathbf{L}$ for LLE and LEM does not affect the final result. To see this, let $\alpha_1, \cdots, \alpha_n$ be eigenvalues of the involved matrix $\mathbf{L}$, and $\mathbf{u}_1, \cdots, \mathbf{u}_n$ be the corresponding *orthogonal* eigenvectors [2]. It can be easily seen that matrix $\mathbf{L}$ in both LLE and LEM has an eigenvalue equal to zero, and the eigenvector corresponding to this eigenvalue is $\mathbf{1}$. Without loss of generality, we may assume $\alpha_1 = 0$ and $\mathbf{u}_1 = \mathbf{1}$. Due to orthogonality of eigenvectors of $\mathbf{L}$, we have $\mathbf{1}^{\mathrm{T}}\mathbf{u}_i = 0$, for all $2 \leq i \leq n$. It immediately follows that:

$$\mathbf{H}\mathbf{u}_i = (\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^{\mathrm{T}})\mathbf{u}_i = \mathbf{u}_i - \frac{1}{n}\mathbf{1}\mathbf{1}^{\mathrm{T}}\mathbf{u}_i = \mathbf{u}_i \qquad \text{for all} \quad 2 \leq i \leq n$$

Now we note that the eigenvectors of the matrix $\lambda_{max}\mathbf{I}-\mathbf{L}$ are the same as the eigenvectors of $\mathbf{L}$, and the corresponding eigenvalues are given by $\lambda_{max}-\alpha_i$. Based on this, the eigenvalue decomposition of matrix $\lambda_{max}\mathbf{I}-\mathbf{L}$ would be the following:

$$\lambda_{max}\mathbf{I} - \mathbf{L} = (\lambda_{max} - \alpha_1)\mathbf{u}_1\mathbf{u}_1^{\mathrm{T}} + (\lambda_{max} - \alpha_2)\mathbf{u}_2\mathbf{u}_2^{\mathrm{T}} + \cdots + (\lambda_{max} - \alpha_n)\mathbf{u}_n\mathbf{u}_n^{\mathrm{T}}$$

Double multiplying both sides of the above equation by $\mathbf{H}$, and noting that $\mathbf{H}\mathbf{u}_1 = 0$, and $\mathbf{H}\mathbf{u}_i = \mathbf{u}_i$ for $2 \leq i \leq n$, we obtain:

$$\begin{aligned}\mathbf{H}(\lambda_{max}\mathbf{I} - \mathbf{L})\mathbf{H} &= (\lambda_{max} - \alpha_1)(\mathbf{H}\mathbf{u}_1)(\mathbf{u}_1^{\mathrm{T}}\mathbf{H}) + \\ &\qquad (\lambda_{max} - \alpha_2)(\mathbf{H}\mathbf{u}_2)(\mathbf{u}_2^{\mathrm{T}}\mathbf{H}) + \cdots + (\lambda_{max} - \alpha_n)(\mathbf{H}\mathbf{u}_n)(\mathbf{u}_n^{\mathrm{T}}\mathbf{H}) \\ &= (\lambda_{max} - \alpha_2)\mathbf{u}_2\mathbf{u}_2^{\mathrm{T}} + \cdots + (\lambda_{max} - \alpha_n)\mathbf{u}_n\mathbf{u}_n^{\mathrm{T}}\end{aligned}$$

As seen above, the eigenvectors $\mathbf{u}_2, \cdots, \mathbf{u}_n$ remain unchanged. Now we note that the solution to the optimization problem presented in 4.2, in this special case, is given by the eigenvectors corresponding to the largest eigenvalues $\lambda_{max}-\alpha_i$, $2 \leq i \leq n$. This is exactly identical to the solution of LLE and LEM.

[2]Note that matrix $\mathbf{L}$ is real and symmetric in both LLE and LEM methods, and thus has orthogonal eigenvectors given by eigenvalue decomposition.

	Kernel K	**Kernel B**	**Comments**
PCA	$\mathbf{X}^{\mathrm{T}}\mathbf{X}$	$\mathbf{G}^{\mathrm{T}}\mathbf{G}$	—
MDS	$-\frac{1}{2}\mathbf{HDH}$	$\mathbf{G}^{\mathrm{T}}\mathbf{G}$	$\mathbf{D}$ is the matrix of pair-wise Euclidian distances.
Isomap	$-\frac{1}{2}\mathbf{HD}^{(\mathcal{G})}\mathbf{H}$	$\mathbf{G}^{\mathrm{T}}\mathbf{G}$	$\mathbf{D}^{(\mathcal{G})}$ is the matrix of pair-wise geodesic distances.
LLE	$\mathbf{L}^{\dagger}$ or $\lambda_{max}\mathbf{I}-\mathbf{L}$	$\mathbf{G}^{\mathrm{T}}\mathbf{G}$	$\mathbf{L}{=}(\mathbf{I}{-}\mathbf{V})^{\mathrm{T}}(\mathbf{I}{-}\mathbf{V})$ $\mathbf{V}$ is the matrix of locally embedded weights
LEM	$\mathbf{L}^{\dagger}$ or $\lambda_{max}\mathbf{I}-\mathbf{L}$	$\mathbf{G}^{\mathrm{T}}\mathbf{G}$	$\mathbf{L}{=}\mathbf{D}{-}\mathbf{W}$ $\mathbf{W}$ is a positive symmetric affinity matrix, $\mathbf{D}$ is the corresponding degree matrix.
Normalized LEM	$\lambda_{max}\mathbf{I}{+}\mathbf{D}^{-\frac{1}{2}}\ \mathbf{W}\mathbf{D}^{-\frac{1}{2}}$	$\mathbf{G}^{\mathrm{T}}\mathbf{G}$	$\mathbf{W}$ is a positive symmetric affinity matrix, $\mathbf{D}$ is the corresponding degree matrix.

Table 4.1: Different Dimensionality Reduction techniques formulated in terms of the proposed general framework. In all techniques, the constraint $\mathbf{GG^T{=}I}$ is used.

4.3 Clustering algorithms

In a clustering application, the goal is to group n given data points into d clusters. (The number of clusters is assumed to be given.) An indicator matrix $\mathbf{F}_{d\times n}$ is usually used to describe the members in each cluster. Indicator matrix $\mathbf{F}$ is defined as follows: $\mathbf{F}_{ij} = 1$ if and only if the j^{th} data point belongs to the i^{th} cluster. In [41], Song *et. al* provide a maximal dependence view of k-means clustering based on HSIC. In this section, we take a different approach from that presented in [41] which allows us to recast a number of well-known clustering algorithm using the formulation given in 4.2. In our approach, matrix $\mathbf{G}$ in 4.2 plays the role of a *normalized* indicator matrix and therefore is the unknown variable in the underlying clustering problem. As presented in Table 4.2, matrix $\mathbf{G}$ may be normalized in different ways depending on the clustering technique. However, in all cases, it satisfies the constraints $\mathbf{G} \geq \mathbf{0}$ and $\mathbf{G}\mathbf{G}^{\mathrm{T}}{=}\mathbf{I}$, by construction.

K-means. We first briefly review the k-means clustering method. Suppose we are given n data points $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$ in $\mathbb{R}^m$. The goal is to cluster the data into d groups indexed by $1, 2, \cdots, d$. To accomplish this task, k-means finds d centroids $\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_d$, and assigns each data point to one (and only one) of them. Let $\mathcal{C}_i$ be the set of indices of data points that are assigned to the i^{th} centroid (cluster), $i = 1, 2, \cdots, d$. The selection of centroid locations and the assignment of data points to clusters in k-means clustering is such that the following objective function is minimized:

$$\sum_{i=1}^{d} \sum_{j \in \mathcal{C}_i} \|\mathbf{x}_j - \mathbf{c}_i\|_2^2 \tag{4.9}$$

This objective function can be written in matrix form as follows:

$$\min_{\mathbf{C},\mathbf{F}} \|\mathbf{X} - \mathbf{C}\mathbf{F}\|_F^2 \tag{4.10}$$

where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$ is the data matrix and $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_d]$ is a matrix consisting of cluster centroids. Matrix $\mathbf{F}$ denotes the indicator matrix, as stated before. To investigate the above optimization problem, we first assume that the indicator matrix $\mathbf{F}$ is fixed and obtain the optimum matrix for $\mathbf{C}$ as a function of $\mathbf{X}$ and $\mathbf{F}$. Taking the derivative of the above objective function with respect to matrix $\mathbf{C}$, we obtain:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{C}}\mathrm{Tr}[(\mathbf{X}-\mathbf{CF})^{\mathrm{T}}(\mathbf{X}-\mathbf{CF})] &= \frac{\partial}{\partial \mathbf{C}}(\mathrm{Tr}[\mathbf{X}^{\mathrm{T}}\mathbf{X}] - 2\mathrm{Tr}[\mathbf{X}^{\mathrm{T}}\mathbf{CF}] + \mathrm{Tr}[\mathbf{FF}^{\mathrm{T}}\mathbf{C}^{\mathrm{T}}\mathbf{C}]) \\ &= -2\mathbf{FX}^{\mathrm{T}} + 2\mathbf{C}\,(\mathbf{FF}^{\mathrm{T}})\end{aligned}$$

Setting the above derivative equal to zero, we obtain:

$$\mathbf{C} = \mathbf{XF}^{\mathrm{T}}(\mathbf{FF}^{\mathrm{T}})^{-1} \tag{4.11}$$

This equation indicates that, for a fixed clustering of data points, the corresponding optimal centroids are given by the cluster means. We may proceed by plugging $\mathbf{C}$ from equation 4.11 into 4.10 to obtain:

$$\max_{\mathbf{F}} \mathrm{Tr}[\mathbf{X}^{\mathrm{T}}\mathbf{XF}^{\mathrm{T}}(\mathbf{FF}^{\mathrm{T}})^{-1}\mathbf{F}]$$

Defining $\mathbf{G} = (\mathbf{FF}^{\mathrm{T}})^{-\frac{1}{2}}\mathbf{F}$, we can express the above optimization problem as follows:

$$\max_{\mathbf{G}} \mathrm{Tr}[\mathbf{G}\,\mathbf{X}^{\mathrm{T}}\mathbf{X}\,\mathbf{G}^{\mathrm{T}}] \tag{4.12}$$

Note that in the above formulation we have $\mathbf{GG}^{\mathrm{T}} = \mathbf{I}$. An example of $\mathbf{G}$ in the case where we have 5 data points and 2 clusters, is presented below. Here it is assumed that the first 2 data points belong to the first cluster and the last 3 data points belong to the second cluster.

$$\mathbf{G} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{pmatrix}$$

One may note that the optimization problem presented in 4.12 can be written as:

$$\max_{\mathbf{G}} \mathrm{Tr}\left[\mathbf{G}(\mathbf{HX}^{\mathrm{T}}\mathbf{XH})\mathbf{G}^{\mathrm{T}}\right]$$

To see this, we note that the double application of matrix $\mathbf{H}$ is equivalent to replacing matrix $\mathbf{X}$ by $\mathbf{XH}$, and this is equivalent to centering the data points around the origin. It is not hard to show that the solution to the original k-means problem presented in 4.9 does not change by shifting the data points by a constant vector. The above results, taken together, imply that the method of k-means clustering can be cast using the formulation presented in 4.2. Note that, here, we need to impose two constraints on matrix $\mathbf{G}$ when solving the optimization problem in 4.2. First, matrix $\mathbf{G}$ is required to be to be orthogonal (which is the same condition as that presented for kernel PCA). In addition, the elements in each row of $\mathbf{G}$ are required to be nonnegative and equal. The second constraint restricts the acceptable values for matrix $\mathbf{G}$ compared to the case of kernel PCA.

Weighted k-means. Weighted k-means clustering is similar to k-means clustering; but it assigns a weight (degree of importance) to each data point. Here we denote the weight of the i^{th} point by M_i. The method of weighted k-means minimizes the following objective function:

$$\sum_{i=1}^{n} M_i \|\mathbf{x}_i - \mathbf{c}_{\pi(i)}\|_2^2 \tag{4.13}$$

where $\pi(i) \in \{1, 2, \cdots, d\}$ is the index of the cluster centroid to which the i^{th} data point is assigned. One can easily see that the weighted k-means clustering with equal weights is equivalent to k-means clustering. The objective function presented in 4.13 may be written in terms of matrices as follows:

$$\min_{\mathbf{C},\mathbf{F}} \|(\mathbf{X} - \mathbf{CF})\mathbf{M}^{\frac{1}{2}}\|_F^2$$

where $\mathbf{M}$ is a diagonal matrix with the i^{th} diagonal entry being equal to M_i. Using matrix manipulations similar to that presented above for k-means clustering, we arrive at the following maximization problem for weighted k-means:

$$\min_{\mathbf{G}} \operatorname{Tr}\left[\mathbf{G}\ \mathbf{M}^{\frac{1}{2}}\ \mathbf{X}^{\mathrm{T}}\mathbf{X}\ \mathbf{M}^{\frac{1}{2}}\ \mathbf{G}^{\mathrm{T}}\right] \tag{4.14}$$

where $\mathbf{G}$ is defined as: $\mathbf{G} = \left(\mathbf{FMF}^{\mathrm{T}}\right)^{-\frac{1}{2}}\mathbf{FM}^{\frac{1}{2}}$. This representation may be used to obtain the formulation given in table (4.2). An example of $\mathbf{G}$ in the case where we have 5 data points and 2 clusters, is presented below. Here it is assumed that the first 2 data points belong to the first cluster and the last 3 data points belong to the second cluster. Weights of the data points are denoted by M_i, $i = 1, \cdots, 5$.

$$\mathbf{G} = \begin{pmatrix} \frac{\sqrt{M_1}}{\sqrt{M_1+M_2}} & \frac{\sqrt{M_2}}{\sqrt{M_1+M_2}} & 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{M_3}}{\sqrt{M_3+M_4+M_5}} & \frac{\sqrt{M_4}}{\sqrt{M_3+M_4+M_5}} & \frac{\sqrt{M_5}}{\sqrt{M_3+M_4+M_5}} \end{pmatrix}$$

Spectral Clustering. In Spectral clustering, the goal again is to cluster n data points $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$ in $\mathbb{R}^m$ into d clusters. Let $\mathcal{C}_i$, $i = 1, \cdots, d$, denote the set of indices of points that are assigned to the i^{th} cluster. In spectral clustering, the clustering task is based on a positive symmetric similarity matrix $\mathbf{W}$ whose (i, j)-th entry provides a measure of similarity between the i^{th} and the j^{th} data point. There are two forms of spectral clustering known as *ratio cut* spectral clustering and *normalized cut* spectral clustering [51]. In ratio cut spectral clustering the goal is to minimize the following objective function:

$$\text{RatioCut}(\mathcal{C}_1, \cdots, \mathcal{C}_n) = \sum_{i=1}^{d} \frac{\text{cut}(\mathcal{C}_i, \bar{\mathcal{C}}_i)}{|\mathcal{C}_i|} \tag{4.15}$$

Here, $|\cdot|$ denotes cardinality of a set, and $\text{cut}(\mathcal{C}_i, \bar{\mathcal{C}}_i)$ is the summation over similarities of every point whose index is in set $\mathcal{C}_i$ with each and every point whose index is out of this set. Mathematically speaking, let $\bar{\mathcal{C}}_i$ be the complement of set $\mathcal{C}_i$ defined as $\bar{\mathcal{C}}_i = \{1, 2, \cdots, n\} - \mathcal{C}_i$. Then $\text{cut}(\mathcal{C}_i, \bar{\mathcal{C}}_i)$ is:

$$\text{cut}(\mathcal{C}_i, \bar{\mathcal{C}}_i) = \sum_{r \in \mathcal{C}_i, s \in \bar{\mathcal{C}}_i} \mathbf{W}_{r,s} = \sum_{r \in \mathcal{C}_i, s \notin \mathcal{C}_i} \mathbf{W}_{r,s}$$

Two important matrices in spectral clustering are the *degree matrix* $\mathbf{D}$ and the *Laplacian matrix* $\mathbf{L}$. The degree matrix is an $n \times n$ diagonal matrix with the i^{th} diagonal entry defined as $\mathbf{D}_{i,i} = \sum_{j=1}^{n} \mathbf{W}_{i,j}$. The Laplacian matrix $\mathbf{L}$ is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$. One may show that the value of $\text{cut}(\mathcal{C}_i, \bar{\mathcal{C}}_i)$ can be reproduced by $\mathbf{F}_i \mathbf{L} \mathbf{F}_i^{\mathrm{T}}$, where $\mathbf{F}_i$ denotes the i^{th} row of the indicator matrix $\mathbf{F}$, $i = 1, 2, \cdots, d$. Based on this, the objective function in 4.15 may be written as:

$$\text{RatioCut}(\mathcal{C}_1, \cdots, \mathcal{C}_n) = \sum_{i=1}^{d} \frac{\mathbf{F}_i \mathbf{L} \mathbf{F}_i^{\mathrm{T}}}{|\mathbf{F}_i \mathbf{F}_i^{\mathrm{T}}|} = \sum_{i=1}^{d} (\mathbf{F}_i / \|\mathbf{F}_i\|) \mathbf{L} (\mathbf{F}_i / \|\mathbf{F}_i\|)^{\mathrm{T}}.$$

This objective function may be summarized in matrix form as follows:

$$\min_{\mathbf{G}} \text{Tr}[\mathbf{G} \mathbf{L} \mathbf{G}^{\mathrm{T}}]$$

where $\mathbf{G} = (\mathbf{F}\mathbf{F}^{\mathrm{T}})^{-\frac{1}{2}} \mathbf{F}$. Note that again we have $\mathbf{G}\mathbf{G}^{\mathrm{T}} = \mathbf{I}$, by construction. We may use the same technique as that used in the previous section to change the above minimization problem to a maximization problem:

$$\max_{\mathbf{G}} \mathrm{Tr}[\mathbf{G}(\lambda_{max}\mathbf{I} - \mathbf{L})\mathbf{G}^{\mathrm{T}}]$$

This formulation is presented in the forth row of Table 4.2. Now we turn our attention to Normalized cut spectral clustering.

Normalized cut spectral clustering is similar to ratio cut spectral clustering, but it uses a modified objective function as follows:

$$\mathrm{RatioCut}(\mathcal{C}_1, \cdots, \mathcal{C}_n) = \sum_{i=1}^{d} \frac{\mathrm{cut}(\mathcal{C}_i, \bar{\mathcal{C}}_i)}{\mathrm{vol}(\mathcal{C}_i)} \tag{4.16}$$

where $\mathrm{vol}(\mathcal{C}_i) = \sum_{r\in\mathcal{C}_i} \mathbf{D}_{r,r}$, by definition. We may express the above objective function presented in 4.16 in terms of the indicator matrix $\mathbf{F}$ as follows:

$$\begin{aligned} \mathrm{RatioCut}(\mathcal{C}_1, \cdots, \mathcal{C}_n) &= \sum_{i=1}^{d} \frac{\mathbf{F}_i\mathbf{L}\mathbf{F}_i^{\mathrm{T}}}{\mathbf{F}_i\mathbf{D}\mathbf{F}_i^{\mathrm{T}}} \qquad (4.17) \\ &= \sum_{i=1}^{d} \frac{\mathbf{F}_i\mathbf{D}^{\frac{1}{2}}\,\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\,\mathbf{D}^{\frac{1}{2}}\mathbf{F}_i^{\mathrm{T}}}{\mathbf{F}_i\mathbf{D}\mathbf{F}_i^{\mathrm{T}}} \\ &= \sum_{i=1}^{d} \left(\frac{\mathbf{F}_i\mathbf{D}^{\frac{1}{2}}}{\|\mathbf{F}_i\mathbf{D}^{\frac{1}{2}}\|}\right) \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} \left(\frac{\mathbf{F}_i\mathbf{D}^{\frac{1}{2}}}{\|\mathbf{F}_i\mathbf{D}^{\frac{1}{2}}\|}\right)^{\mathrm{T}} \end{aligned}$$

The above summation may be summarized in the matrix form as follows:

$$\min_{\mathbf{G}} \mathrm{Tr}[\mathbf{G}\,\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\,\mathbf{G}^{\mathrm{T}}] \tag{4.18}$$

where $\mathbf{G} = (\mathbf{F}\mathbf{D}\mathbf{F}^{\mathrm{T}})^{-\frac{1}{2}}\mathbf{F}\mathbf{D}^{\frac{1}{2}}$. Note that once again we have $\mathbf{G}\mathbf{G}^{\mathrm{T}} = \mathbf{I}$. An example of $\mathbf{G}$ in the case where we have 5 data points and 2 clusters, is presented below. Here, it is again assumed that the first 2 data points belong to the first cluster and the last 3 data points belong to the second cluster. Also, the degree corresponding to the data i^{th} data point, $i = 1, \cdots, 5$, is denoted by D_i.

$$\mathbf{G} = \begin{pmatrix} \frac{\sqrt{D_1}}{\sqrt{D_1+D_2}} & \frac{\sqrt{D_2}}{\sqrt{D_1+D_2}} & 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{D_3}}{\sqrt{D_3+D_4+D_5}} & \frac{\sqrt{D_4}}{\sqrt{D_3+D_4+D_5}} & \frac{\sqrt{D_5}}{\sqrt{D_3+D_4+D_5}} \end{pmatrix}$$

The objective function presented in 4.18 may be further simplified to:

$$\begin{aligned}\mathrm{Tr}[\mathbf{G}\,\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\,\mathbf{G}^{\mathrm{T}}] &= \mathrm{Tr}[\mathbf{G}\,\mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-\frac{1}{2}}\,\mathbf{G}^{\mathrm{T}}] \\ &= \mathrm{Tr}[\mathbf{G}\,(\mathbf{I}-\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})\,\mathbf{G}^{\mathrm{T}}] \\ &= d-\mathrm{Tr}[\mathbf{G}\,\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}\,\mathbf{G}^{\mathrm{T}}]\end{aligned}$$

Based on this, the minimization problem presented in 4.18 may be expressed as a maximization problem as follows:

$$\max_{\mathbf{G}}\mathrm{Tr}[\mathbf{G}\,\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}\,\mathbf{G}^{\mathrm{T}}] \tag{4.19}$$

Since the matrix $\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$ is not necessarily positive semidefinite, we may use the same technique as that introduced in the previous section to define the new kernel $\lambda_{max}\mathbf{I}+\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$. The result is presented in Table 4.2.

Remark. One may see that the kernels presented for weighted k-means, ratio cut spectral clustering and normalized cut spectral clustering are not double centered. Based on this, application of the centering matrix $\mathbf{H}$ might lead to slightly different versions of these algorithms.

4.4 Metric learning

In metric learning applications, we assume the data labels are given and the aim is to learn a linear transformation $\mathbf{U}$ such that the location of the points in the new space, $\mathbf{U}^{\mathrm{T}}\mathbf{X}$, well matches the underlying given labels.

In this section we mostly deal with the formulation of LEM introduced in section 4.2. The LEM method can be formulated as follows:

$$\min_{\mathbf{G}}\mathrm{Tr}\left[\mathbf{G}\mathbf{L}\mathbf{G}^{\mathrm{T}}\right], \qquad \text{s.t.}\ \ \mathbf{G}\mathbf{A}\mathbf{G}^{\mathrm{T}}{=}\mathbf{I} \tag{4.20}$$

where $\mathbf{L}{=}\mathbf{D}{-}\mathbf{W}$ is the Laplacian matrix, $\mathbf{W}$ is an affinity matrix and $\mathbf{D}$ is the corresponding degree matrix. In the above formulation, it is common to set matrix $\mathbf{A}$ either to the identity

	Kernel K	**Kernel B**	**Comments**
K-means	$\mathbf{X}^{\mathrm{T}}\mathbf{X}$	$\mathbf{G}^{\mathrm{T}}\mathbf{G}$	$\mathbf{G}=\left(\mathbf{F}\mathbf{F}^{\mathrm{T}}\right)^{-\frac{1}{2}}\mathbf{F}$ is the normalized indicator matrix
Kernel k-means	$\mathbf{K}$	$\mathbf{G}^{\mathrm{T}}\mathbf{G}$	$\mathbf{G}=\left(\mathbf{F}\mathbf{F}^{\mathrm{T}}\right)^{-\frac{1}{2}}\mathbf{F}$ is the normalized indicator matrix
Weighted k-means	$\mathbf{M}^{\frac{1}{2}}\ \mathbf{X}^{\mathrm{T}}\mathbf{X}\ \mathbf{M}^{\frac{1}{2}}$	$\mathbf{G}^{\mathrm{T}}\mathbf{G}$	$\mathbf{M}$ diagonal weight matrix $\mathbf{G}=\left(\mathbf{F}\mathbf{M}\mathbf{F}^{\mathrm{T}}\right)^{-\frac{1}{2}}\mathbf{F}\mathbf{M}^{\frac{1}{2}}$ is the normalized indicator matrix
Ratio Cut spectral clustering	$\lambda_{max}\mathbf{I}-\mathbf{L}$	$\mathbf{G}^{\mathrm{T}}\mathbf{G}$	$\mathbf{L}=\mathbf{D}-\mathbf{W}$ $\mathbf{W}$ is a positive symmetric affinity matrix, $\mathbf{D}$ is the corresponding degree matrix. $\mathbf{G}=\left(\mathbf{F}\mathbf{F}^{\mathrm{T}}\right)^{-\frac{1}{2}}\mathbf{F}$ is the normalized indicator matrix
Normalized Cut spectral clustering	$\lambda_{max}\mathbf{I}+\mathbf{D}^{-\frac{1}{2}}\ \mathbf{W}\mathbf{D}^{-\frac{1}{2}}$	$\mathbf{G}^{\mathrm{T}}\mathbf{G}$	$\mathbf{L}=\mathbf{D}-\mathbf{W}$ $\mathbf{W}$ is a positive symmetric affinity matrix, $\mathbf{D}$ is the corresponding degree matrix. $\mathbf{G}=\left(\mathbf{F}\mathbf{D}\mathbf{F}^{\mathrm{T}}\right)^{-\frac{1}{2}}\mathbf{F}\mathbf{D}^{\frac{1}{2}}$ is the normalized indicator matrix

Table 4.2: Different clustering techniques formulated in terms of the proposed general framework. In all techniques, the constraints $\mathbf{G} \geq \mathbf{0}$ and $\mathbf{G}\mathbf{G}^{\mathbf{T}}=\mathbf{I}$ are assumed.

matrix $\mathbf{I}$ or the degree matrix $\mathbf{D}$. Using the Rayleigh-Ritz theorem, it can be shown that the solution subject to the former constraint is given by the eigenvectors corresponding to the smallest eigenvalues of $\mathbf{L}$. Also, using the change of variable $\mathbf{G} = \mathbf{G}\mathbf{D}^{\frac{1}{2}}$, one can show that the solution subject to the later constraint is given by the eigenvectors corresponding to the smallest eigenvalues of $\mathbf{D}^{-1}\mathbf{W}$ [3].

In this section, we propose a new constraint for solving the optimization problem presented in 4.20. Suppose all eigenvalues of $\mathbf{L}$ are in interval $[0, 1)$. If not, one can multiply the underlying affinity matrix $\mathbf{W}$ by an appropriate constant such that the above condition become satisfied [4]. We propose to solve the optimization problem in 4.20 by setting the value of $\mathbf{A}$ to $\mathbf{H} - \mathbf{L}$, where $\mathbf{H}$ is again the centering matrix. The resulting optimization problem would be:

$$\min_{\mathbf{G}} \operatorname{Tr}\left[\mathbf{G}\mathbf{L}\mathbf{G}^{\mathrm{T}}\right], \qquad \text{s.t.} \;\; \mathbf{G}(\mathbf{H} - \mathbf{L})\mathbf{G}^{\mathrm{T}}=\mathbf{I} \tag{4.21}$$

We will see some applications and interpretations of this formulation later in this section. A discussion on the solution of the LEM problem subject to this constraint is presented in Appendix A.

At this step, let us show that the optimization problem presented in 4.21 can be expressed in terms of the formulation given in 4.2. To do so, we choose a kernel of the form $\mathbf{K} = \lambda(\mathbf{H}-\mathbf{L})-\mathbf{L}$ in 4.2, where λ is a constant factor. We show that: (i) the optimization problem presented in 4.2 with the above kernel and constraint $\mathbf{G}(\mathbf{H} - \mathbf{L})\mathbf{G}^{\mathrm{T}} = \mathbf{I}$ is equivalent to the optimization problem presented in 4.21. (ii) There exists some λ for which the above kernel

[3]Note that this type of normalization of the Laplacian matrix $\mathbf{L}$ is slightly different from that presented in section 4.2, where $\mathbf{L}$ was symmetrically normalized as $\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$.

[4]This constant should converge to $1/\lambda_m$ from left, where λ_m is the largest eigenvalue of $\mathbf{L}$. Note that $\mathbf{L}$ is positive semidefinite, and all its eigenvalues are nonnegative.

is positive semidefinite. Verification of the first part is straightforward. We note that:

$$\begin{aligned}\mathrm{Tr}[\mathbf{G}\mathbf{K}\mathbf{G}^{\mathrm{T}}] &= \mathrm{Tr}[\mathbf{G}\left(\lambda(\mathbf{H}-\mathbf{L})-\mathbf{L}\right)\mathbf{G}^{\mathrm{T}}] \\ &= \lambda\mathrm{Tr}[\mathbf{G}(\mathbf{H}-\mathbf{L})\mathbf{G}^{\mathrm{T}}] - \mathrm{Tr}[\mathbf{G}\mathbf{L}\mathbf{G}^{\mathrm{T}}] \\ &= \lambda\mathrm{Tr}[\mathbf{I}] - \mathrm{Tr}[\mathbf{G}\mathbf{L}\mathbf{G}^{\mathrm{T}}] \\ &= \lambda d - \mathrm{Tr}[\mathbf{G}\mathbf{L}\mathbf{G}^{\mathrm{T}}]\end{aligned} \tag{4.22}$$

Since λd is a constant, maximizing the objective function in 4.22 is equivalent to minimizing the objective function in 4.21. From this, it follows that the optimization problems in 4.2 and 4.21 are equivalent when they are solved subject to constraint $\mathbf{G}(\mathbf{H}-\mathbf{L})\mathbf{G}^{\mathrm{T}} = \mathbf{I}$.

Now we show that there exists a parameter λ for which the introduced kernel $\mathbf{K} = \lambda(\mathbf{H}-\mathbf{L})-\mathbf{L}$ is positive semidefinite. Let $0 \leq \alpha_1 \leq \cdots \leq \alpha_n$ be eigenvalues of the Laplacian matrix, $\mathbf{L}$. We show that the value of $\lambda_0 = \max\{\frac{\alpha_i}{1-\alpha_i}\}_{i=1}^{n}$ satisfies the above property, that is the corresponding kernel $\mathbf{K}_0 = \lambda_0(\mathbf{H}-\mathbf{L})-\mathbf{L}$ will be positive semidefinite. Considering the fact that $1-\alpha_i > 0$ for all $1 \leq i \leq n$, one can check that with the above choice of λ_0 we have:

$$\lambda_0 - (\lambda_0+1)\alpha_i \geq 0 \qquad \text{for all } \ 1 \leq i \leq n \tag{4.23}$$

We will use these inequalities later in our proof. To show that $\mathbf{K}_0$ is positive semidefinite, we first note that $\mathbf{K}_0 = \lambda_0(\mathbf{H}-\mathbf{L})-\mathbf{L} = \lambda_0\mathbf{H} - (\lambda_0+1)\mathbf{L}$ is a real symmetric matrix. So its eigenvalues are real, and the eigenvectors corresponding to distinct eigenvalues are orthogonal. It is not hard to see that $\mathbf{1}$ is an eigenvector of $\mathbf{K}_0$, and the corresponding eigenvalue is zero. To show that $\mathbf{K}_0$ is positive semidefinite, we need to show that the nonzero eigenvalues of $\mathbf{K}_0$ are positive. Let β be a nonzero eigenvalue of $\mathbf{K}_0$, and $\mathbf{u}$ be the corresponding eigenvector. Due to the orthogonality property mentioned above, we have $\mathbf{1}^{\mathrm{T}}\mathbf{u} = 0$. From this expression, one can see that $\mathbf{H}\mathbf{u} = (\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^{\mathrm{T}})\mathbf{u} = \mathbf{u}$. Now recall that $\mathbf{K}_0\mathbf{u} = \beta\mathbf{u}$, so we have:

$$\beta\mathbf{u} = \mathbf{K}_0\mathbf{u} = (\lambda_0\mathbf{H} - (\lambda_0+1)\mathbf{L})\mathbf{u} = \lambda_0\mathbf{H}\mathbf{u} - (\lambda_0+1)\mathbf{L}\mathbf{u} = \lambda_0\mathbf{u} - (\lambda_0+1)\mathbf{L}\mathbf{u}$$

Rearranging the terms in the above equation, we obtain:

$$\mathbf{L}\mathbf{u} = \frac{\lambda_0 - \beta}{\lambda_0 + 1}\mathbf{u}$$

From this expression it follows that if β is a nonzero eigenvalue of $\mathbf{L}$, then $\alpha^* = \frac{\lambda_0-\beta}{\lambda_0+1}$ is an eigenvalue of $\mathbf{L}$. Using this result, one can express the value of β as follows:

$$\beta = \lambda_0 - (\lambda_0 + 1)\alpha^*$$

Recall from 4.23 that we have $\lambda_0 - (\lambda_0 + 1)\alpha_i \geq 0$ for all $1 \leq i \leq n$. So $\beta \geq 0$, and the proof is complete.

In this chapter, we use the formulation presented in 4.21 to reproduce two examples of successful metric learning techniques, namely, Fisher's discriminant analysis (FDA) and Closed-form metric learning (CFML). In addition, we propose a new metric learning method based on ideas from normalized cut spectral clustering. The proposed method is associated with a convex optimization problem.

Let us start by reviewing the methods of Fisher's discriminant analysis (FDA) and Closed-form metric learning (CFML).

Fisher Discriminant Analysis (FDA): Let $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$ be n data points in $\mathbb{R}^m$. Assume the labels of data are given, meaning that the data is already partitioned into, say d, classes. Let $\mathcal{C}_i$ be the set of indices of data points that belong to the i^{th} class, $i = 1, 2, \cdots, d$. In metric learning methods the goal is to find a linear transformation $\mathbf{U}$ such that the points $\mathbf{z}_i$=$\mathbf{U}^{\mathrm{T}}\mathbf{x}_i$ $(i = 1, \ldots, n)$ in the low-dimensional space satisfy the following property: Points in the same class lie as close as possible to each other, and points from different classes lie as far as possible from each other. FDA accomplishes this goal by maximizing the following objective function:

$$\max_{\mathbf{U}} \operatorname{Tr}[(\mathbf{U}^{\mathrm{T}}\mathbf{S}_W\mathbf{U})^{-1}(\mathbf{U}^{\mathrm{T}}\mathbf{S}_B\mathbf{U})] \tag{4.24}$$

where $\mathbf{S}_W$ is the within covariance of the data points, and $\mathbf{S}_B$ is the between covariance of the data points. These covariance matrices are mathematically defined as:

$$\begin{aligned}\mathbf{S}_W &= \sum_{i=1}^{d}\sum_{j\in\mathcal{C}_i}(\mathbf{x}_j-\mathbf{c}_i)^{\mathrm{T}}(\mathbf{x}_j-\mathbf{c}_i) \\ \mathbf{S}_B &= \sum_{i=1}^{d}(\mathbf{c}_i-\mathbf{c})^{\mathrm{T}}(\mathbf{c}_i-\mathbf{c})\end{aligned} \tag{4.25}$$

where $\mathbf{c}_i$ denotes the mean point of the i^{th} class of data, and $\mathbf{c}$ denotes the mean of the whole mean points $\mathbf{c}_i$'s, $i = 1, 2, \cdots, d$. It is shown (see, e.g., [18]) that the solutions to the above optimization problem are determined by those eigenvectors corresponding to the largest eigenvalues of matrix $\mathbf{S}_W^{-1}\mathbf{S}_B$.

To extract the FDA method based on the formulation presented in (4.21), we set the value of $\mathbf{G}$ to $\mathbf{U}^{\mathrm{T}}\mathbf{X}$, that is, a low-dimensional representation of the data. Also, we set the value of $\mathbf{L}$ to a Laplacian of the labels. The appropriate Laplacian for the labels that reproduces the FDA method is obtained based on the affinity matrix $\mathbf{W}{=}\mathbf{G}^{\mathrm{T}}\mathbf{G}$, where $\mathbf{G}{=}\left(\mathbf{F}\mathbf{F}^{\mathrm{T}}\right)^{-\frac{1}{2}}\mathbf{F}$ is a normalized indicator matrix. A simple example of $\mathbf{W}$ for the case where we have only two clusters with 2 and 3 members is given below:

$$\mathbf{W} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

As mentioned earlier, the Laplacian matrix would be obtained from this affinity matrix via equation $\mathbf{L}{=}\mathbf{D}{-}\mathbf{W}$. To solve the underlying LEM problem we first need to choose a constraint, as presented in (4.20). Using the newly introduced constraint $\mathbf{G}\left(\mathbf{H}{-}\mathbf{L}\right)\mathbf{G}^{\mathrm{T}}{=}\mathbf{I},$ which in this application becomes $\left(\mathbf{U}^{\mathrm{T}}\mathbf{X}\right)\left(\mathbf{H}{-}\mathbf{L}\right)\left(\mathbf{X}^{\mathrm{T}}\mathbf{U}\right){=}\mathbf{I}$, we arrive at the following optimization problem:

$$\min_{\mathbf{U}} \mathrm{Tr}\left[\mathbf{U}^{\mathrm{T}}\mathbf{S}_W\mathbf{U}\right], \qquad \text{s.t.} \;\; \mathbf{U}^{\mathrm{T}}\mathbf{S}_B\mathbf{U}{=}\mathbf{I} \tag{4.26}$$

where $\mathbf{S}_W = \mathbf{X}\mathbf{L}\mathbf{X}^{\mathrm{T}}$ and $\mathbf{S}_B = \mathbf{X}\left(\mathbf{H}{-}\mathbf{L}\right)\mathbf{X}^{\mathrm{T}}$. One may see that the values of $\mathbf{S}_W$ and $\mathbf{S}_B$ presented above, in fact, denote the within- and between covariance of the data points $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$, presented in 4.25. From the Rayleigh-Ritz theorem, it follows that the solutions to the above optimization problem are determined by the eigenvectors corresponding to the smallest eigenvalues of matrix $\mathbf{S}_B^{-1}\mathbf{S}_W$. Recall that the solutions of FDA are given by the top eigenvectors of $\mathbf{S}_W^{-1}\mathbf{S}_B$. The equivalence between the two approaches can be seen by considering the fact that the matrix $\mathbf{S}_B^{-1}\mathbf{S}_W$ is the inverse of $\mathbf{S}_W^{-1}\mathbf{S}_B$, and thus its eigenvalues are the inverse of the eigenvalues of matrix $\mathbf{S}_W^{-1}\mathbf{S}_B$.

Closed Form Metric Learning (CFML): closed Form Metric Learning (CFML) [1] is a metric learning technique which, similar to FDA, provides *closed-form* solutions for the optimum linear transformation $\mathbf{U}$. In this section, we discuss type (II) of this method. Let $\mathcal{S}$ be the set of all similar pairs in the data, e.i., the set of all pairs of data that belong to the same class. Also let $\mathcal{D}$ be the set of all dissimilar pairs, e.i., the set of all pairs that belong to different classes. Type II of CFML solves the following optimization problem:

$$\min_{\mathbf{U}} \operatorname{Tr}[\mathbf{U}^{\mathrm{T}}(\mathbf{S}_W - \mathbf{S}_B)\mathbf{U}] \qquad \text{s.t.} \;\; \mathbf{U}^{\mathrm{T}}\mathbf{S}_W\mathbf{U} = \mathbf{I}$$

where $\mathbf{S}_W$ and $\mathbf{S}_B$ are defined as:

$$\mathbf{S}_W = \frac{1}{|\mathcal{S}|}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}} \tag{4.27}$$

$$\mathbf{S}_B = \frac{1}{|\mathcal{D}|}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}} \tag{4.28}$$

It may be shown [1] that the solution to the above optimization problem is given by the top eigenvectors of matrix $\mathbf{S}_W^{-1}\mathbf{S}_B$.

To extract this technique based on the formulation given in (4.20), we may use the same steps as those described for the FDA. The only difference is that we need to use the affinity matrix $\mathbf{W}{=}\frac{1}{n}\mathbf{F}^{\mathrm{T}}\mathbf{F}$ to include the side information given by the labels. A simple example of $\mathbf{W}$ for the case where we have only two clusters with 2 and 3 members is given below:

$$\mathbf{W} = \frac{1}{5}\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

It is not hard to show that, in this case, $\mathbf{S}_W = \mathbf{X}\mathbf{L}\mathbf{X}^{\mathrm{T}}$ and $\mathbf{S}_B = \mathbf{X}\left(\mathbf{H}{-}\mathbf{L}\right)\mathbf{X}^{\mathrm{T}}$ would be equal to $\mathbf{S}_W$ and $\mathbf{S}_B$ presented in equations (4.27) and (4.28), up to scaling factors $\frac{1}{|\mathcal{S}|}$ and $\frac{1}{|\mathcal{D}|}$, respectively. Recall that the solution to the optimization problem presented in 4.26 is given by the eigenvectors corresponding to the smallest eigenvalues of matrix $\mathbf{S}_B^{-1}\mathbf{S}_W$. So the solution is identical to the solution of the CFML, which is given by the eigenvectors corresponding to the top eigenvalues of matrix $\mathbf{S}_W^{-1}\mathbf{S}_B$.

A new Algorithm: Metric learning based on Normalized cut spectral clustering (ML-SpCl):

In the previous subsections, we used a *linear kernel* for the low-dimensional representation $\mathbf{U}^{\mathrm{T}}\mathbf{X}$ in association with a *Laplacian* of the labels. In this subsection, we use a *Laplacian* of low-dimensional representation $\mathbf{U}^{\mathrm{T}}\mathbf{X}$ in association with a *linear kernel* of $\mathbf{G}$. As we will see shortly, this provides us with a new algorithm for distance metric learning which has an associated underlying convex optimization problem. The main idea in this technique is as follows: we learn the transformation matrix $\mathbf{U}$ such that the clusters obtained by *Normalized cut spectral clustering* in the low-dimensional representation of data (given by $\mathbf{U}^{\mathrm{T}}\mathbf{X}$) matches the clusters already given by the data labels.

To do so, we use a Gaussian similarity matrix $\mathbf{W}$ to construct the Laplacian for $\mathbf{U}^{\mathrm{T}}\mathbf{X}$, as follows:

$$\mathrm{W_{ij}}{=}\exp\left(-\left\|\mathbf{U}^{\mathrm{T}}\mathbf{x}_{\mathrm{i}}{-}\mathbf{U}^{\mathrm{T}}\mathbf{x}_{\mathrm{j}}\right\|^2\right){=}\exp\left(-\left(\mathbf{x}_{\mathrm{i}}{-}\mathbf{x}_{\mathrm{j}}\right)^{\mathrm{T}}\mathbf{P}\left(\mathbf{x}_{\mathrm{i}}{-}\mathbf{x}_{\mathrm{j}}\right)\right) \tag{4.29}$$

where $\mathbf{P} = \mathbf{U}\mathbf{U}^{\mathrm{T}}$ is a symmetric semi-positive matrix.

	Kernel K	Affinity matrix	Comments
CFML	$\left(\mathbf{U}^{\mathrm{T}}\mathbf{X}\right)^{\mathrm{T}}\left(\mathbf{U}^{\mathrm{T}}\mathbf{X}\right)$	$\mathbf{W}=\frac{1}{\mathrm{n}}\mathbf{F}^{\mathrm{T}}\mathbf{F}$	$\mathbf{F}\mathbf{F}^{\mathrm{T}}$=$\mathbf{D}$, $\mathbf{F}\geq\mathbf{0}$ **F** is the indicator matrix consisting of 0's and 1's.
FDA	$\left(\mathbf{U}^{\mathrm{T}}\mathbf{X}\right)^{\mathrm{T}}\left(\mathbf{U}^{\mathrm{T}}\mathbf{X}\right)$	$\mathbf{W}=\mathbf{G}^{\mathrm{T}}\mathbf{G}$	$\mathbf{G}\mathbf{G}^{\mathrm{T}}$=$\mathbf{I}$, $\mathbf{G}\geq\mathbf{0}$ $\mathbf{G}=\left(\mathbf{F}\mathbf{F}^{\mathrm{T}}\right)^{-\frac{1}{2}}\mathbf{F}$ is normalized indicator matrix
ML-SpCl	$\mathbf{G}^{\mathrm{T}}\mathbf{G}$	$\mathrm{W}_{\mathrm{ij}}=\exp\left(-\left\|\mathbf{U}^{\mathrm{T}}\mathbf{x}_{\mathrm{i}}-\mathbf{U}^{\mathrm{T}}\mathbf{x}_{\mathrm{j}}\right\|^{2}\right)$	$\mathbf{G}\mathbf{G}^{\mathrm{T}}$=$\mathbf{I}$, $\mathbf{G}\geq\mathbf{0}$ $\mathbf{G}=\left(\mathbf{F}\mathbf{F}^{\mathrm{T}}\right)^{-\frac{1}{2}}\mathbf{F}$ is normalized indicator matrix

Table 4.3: Different Metric learning techniques formulated in terms of the proposed general framework.

As mentioned in previous sections, in the Normalized cut spectral clustering, we are looking for a normalized indicator function $\mathbf{G}=\left(\mathbf{F}\mathbf{F}^{\mathbf{T}}\right)^{-\frac{1}{2}}\mathbf{F}$ ($\mathbf{G}\mathbf{G}^{\mathrm{T}}=\mathbf{I}$) which maximizes the following trace:

$$\max \quad \mathrm{Tr}\left[\mathbf{G}\ \ \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}\ \ \mathbf{G}^{\mathrm{T}}\right] \tag{4.30}$$

where **W** is the Gaussian similarity matrix and **D** is the corresponding degree matrix. In our application, however, the situation is the reverse. That is, we already have an indicator matrix **G** and need to learn the similarity matrix **W**, or equivalently the transformation matrix **U**. This can be accomplished by setting the value of **G** based on the given data labels and solving the maximization problem presented in (4.30) for **U**.

Relation to other techniques: The technique introduced above is closely related to Neighborhood Component Analysis (NCA) [21] and Maximally Collapsing Metric Learning

(MCML) techniques [20]. In NCA, a conditional probability $p(i|j)$ is defined as follows:

$$p(i|j) = \frac{\exp(-\|\mathbf{U}^{\mathrm{T}}\mathbf{x}_i - \mathbf{U}^{\mathrm{T}}\mathbf{x}_j\|^2)}{\sum_{k\neq i}\exp(-\|\mathbf{U}^{\mathrm{T}}\mathbf{x}_i - \mathbf{U}^{\mathrm{T}}\mathbf{x}_k\|^2)} \tag{4.31}$$

where $\mathbf{U}$ is the transformation which is supposed to be learnt. This conditional probability is interpreted as the probability that point i selects another point j as one of its neighbors (classmates). The probability that point i selects itself as a neighbor, $p(i|i)$, is set to zero by definition in NCA. The NCA method maximizes the following objective function:

$$\max_{\mathbf{U}} \sum_{t=1}^{d} \sum_{(i,j)\in\mathcal{C}_t} p(i|j)$$

where, as before, set $\mathcal{C}_t$, $t \in \{1, \cdots, d\}$, is the set of indices of points in the i^{th} class. It may be easily seen that the objective function of NCA can be written as follows:

$$\max_{\mathbf{U}} \quad \mathrm{Tr}\left[\mathbf{D}^{-1}\mathbf{W}\ (\mathbf{F}^{\mathrm{T}}\mathbf{F})\right] = \mathrm{Tr}\left[\mathbf{F}\ \mathbf{D}^{-1}\mathbf{W}\ \mathbf{F}^{\mathrm{T}}\right]$$

where $\mathbf{W}$ is the Gaussian similarity matrix presented in 4.29.

Before discussing the relation between the NCA and our proposed method, let us review the MCML technique, as well. The method of MCML maximizes the KL divergence between the above conditional probabilities and the conditional probabilities in the ideal situation. The conditional probabilities in the ideal situation can be obtained from expression 4.31; but in this case, it is assumed that all points in the same class collapse to a single point, and at the same time, points from different classes get infinitely far from each other. It can be seen that the maximization of KL divergence in this problem results in the following maximization problem:

$$\max_{\mathbf{U}} \sum_{t=1}^{d} \sum_{(i,j)\in\mathcal{C}_t} \frac{1}{|\mathcal{C}_t - 1|} \log p(i|j) \tag{4.32}$$

One may see that this maximization problem is equivalent to maximizing the following objective function:

$$\max_{\mathbf{U}} \mathrm{Tr}\left[\log(\mathbf{D}^{-1}\mathbf{W})(\mathbf{G}^{\mathrm{T}}\mathbf{G})\right] = \mathrm{Tr}\left[\mathbf{G}\ \log(\mathbf{D}^{-1}\mathbf{W})\ \mathbf{G}^{\mathrm{T}}\right] \tag{4.33}$$

where $\mathbf{G} = (\mathbf{F}\mathbf{F}^{\mathrm{T}} - \mathbf{I})^{-\frac{1}{2}}\mathbf{F}$ is the normalized indicator matrix. (As before, $\mathbf{F}$ is the indicator matrix consisting of zeros and ones). In 4.33, $\log(\cdot)$ denotes the componentwise logarithm operation.

One may note that in both the NCA and MCML techniques we are aligning a non-symmetric matrix $\mathbf{D}^{-1}\mathbf{W}$ with the symmetric matrix $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ or $\mathbf{G}^{\mathrm{T}}\mathbf{G}$, which may not make good sense. Maaten and Hinton in [50] suggest using $\left(\mathbf{D}^{-1}\mathbf{W}+(\mathbf{D}^{-1}\mathbf{W})^{\mathrm{T}}\right)/2$ or $(\mathrm{Tr}\,[\mathbf{D}])^{-1}\mathbf{W}$ instead of $\mathbf{D}^{-1}\mathbf{W}$, to obtain a symmetric probability matrix. However, none of these alternatives has an explicit clustering interpretation. So, in one sense, our method may be interpreted as a systematic way for obtaining such a symmetric similarity matrix to be aligned with $\mathbf{G}^{\mathrm{T}}\mathbf{G}$.

The optimization problem presented in (4.30), as it stands, is non-convex, and may be solved in a similar fashion to the optimization problem presented in the NCA technique. However, one might change the formulation presented in (4.30) by applying a logarithm operator to the entries of $\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$. In this case, it may be seen that, similar to the situation that we have in MCML, the resulting optimization problem would be convex. To see this, we note that the logarithm of (i, j) entry of the symmetrically normalized Laplacian, $\left[\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}\right]_{\mathrm{ij}}$, equals:

$$
\begin{aligned}
-\log\left(\left[\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}\right]_{\mathrm{ij}}\right) &= -\log\left(\left[\frac{\mathrm{W}_{ij}}{\sqrt{\sum_j \mathrm{W}_{\mathrm{ij}}}\sqrt{\sum_i \mathrm{W}_{\mathrm{ij}}}}\right]_{ij}\right) \\
&= (\mathbf{x}_{\mathrm{i}}-\mathbf{x}_{\mathrm{j}})^{\mathrm{T}}\mathbf{P}\,(\mathbf{x}_{\mathrm{i}}-\mathbf{x}_{\mathrm{j}}) + \\
&\quad \frac{1}{2}\log\left(\sum_j \mathrm{W}_{\mathrm{ij}}\right) + \frac{1}{2}\log\left(\sum_i \mathrm{W}_{\mathrm{ij}}\right)
\end{aligned} \tag{4.34}
$$

As in MCML, the first term is linear in $\mathbf{P}$ and thus convex, and the next terms are $\log(\sum\exp)$ functions of affine functions of $\mathbf{P}$ and therefore they are convex, as well. As a result, the underlying convex optimization problem may be solved uniquely and does not suffer from the local minima problem. Interesting future work would be to apply this method to a number of real-world data sets, and compare its performance with the NCA and MCML techniques.

Chapter 5

Conclusion and Future Work

Herein, we have presented a feature selection algorithm based on Hilbert-Schmidt independence criterion. To do so, we defined an optimization problem whose solutions are known to be eigenvectors of a given matrix, and then we obtained sparse solutions to this optimization problem. Optimization problems of this type frequently appear in the area of machine learning (and other areas), and the same methodology can be used to achieve sparse solutions to such problems. This general methodology can thus have applications in a variety of areas including sparse SVD, variable selection, etc.

To perform the feature selection task, as mentioned in previous chapters we identified a *sparse* projection vector such that the dependence between the low-dimensional representation of data obtained from this projection and the response variables is maximized. We used a linear kernel for data and an arbitrary kernel for labels. Since there is no restriction on the kernel for the labels, several variations can be derived from our method. One may note that each of these techniques corresponds to a linear metric learning algorithm derived from the original optimization problem when no sparsity constraint is imposed. Fisher discriminant analysis (FDA) and closed-form metric learning (CFML) are examples of such metric learning techniques. Based on this flexibility, an immediate future direction of research is to test the applicability of our algorithm to the feature selection task in contexts rather than gene expression data.

As for the biological part of this work, the research is ongoing to learn more about the biological significance of the extracted genes. An ultimate goal of the fields of computational and molecular biology is to extract the gene regulatory networks which control and regulate the expression levels of genes in the cell. Since our proposed algorithm is a multi-gene feature selection algorithm it has the capability to take into account the interactions between genes. This feature is particularly relevant in the study of gene regulatory networks. So one of the directions of future research is to combine the results obtained from gene expression data analysis with other sources of biological information to pave the way for an understanding and analysis of the genetic networks in cancerous and normal cells.

APPENDICES

Appendix A

In the appendix, we elaborate to some extent on the constraint $\mathbf{G}\left(\mathbf{H}-\mathbf{L}\right)\mathbf{G}^{\mathrm{T}}=\mathbf{I}$ used for solving the LEM objective function presented in equation (4.20). First, we note that $\mathbf{H}-\mathbf{L}$, as it stands[1], does not have full rank[2]. The reason is that, columns of both matrices $\mathbf{H}$ and $\mathbf{L}$, and thus $\mathbf{H}-\mathbf{L}$**,** sum to zero. So the solutions are not simply given by the Rayleigh-Ritz theorem as the lowest eigenvectors of $\left(\mathbf{H}-\mathbf{L}\right)^{-1}\mathbf{L}$. We guess that the solution to this problem should involve the pseudo inverse of matrix $\mathbf{H}-\mathbf{L}$.

In this appendix, however, we address a weaker version of this constraint, that is the trace constraint $\mathrm{Tr}\left[\mathbf{G}\left(\mathbf{H}-\mathbf{L}\right)\mathbf{G}^{\mathrm{T}}\right]=1$. We will show that the solution to this new optimization problem would be a rank-one matrix $\mathbf{G}$ whose columns are given by eigenvector corresponding to the smallest (nonzero) eigenvalue of $\mathbf{L}$.

To see this, we use a Lagrange multiplier α to obtain:

$$\mathrm{L}\left(\mathbf{G},\alpha\right)=\mathrm{Tr}\left[\mathbf{G}\mathbf{L}\mathbf{G}^{\mathrm{T}}\right]-\alpha\ \left(\mathrm{Tr}\left[\mathbf{G}\left(\mathbf{H}-\mathbf{L}\right)\mathbf{G}^{\mathrm{T}}\right]-1\right)$$

[1] As seen in metric Learning applications, we may set the value of $\mathbf{G}$ to $\mathbf{U}^{\mathrm{T}}\mathbf{X}$. In this case, the introduced constraint becomes $\mathbf{U}^{\mathrm{T}}\left(\mathbf{X}\left(\mathbf{H}-\mathbf{L}\right)\mathbf{X}^{\mathrm{T}}\right)\mathbf{U}=\mathbf{I}$ which is expressed in terms of the new variable $\mathbf{U}$. In this form, the matrix $\mathbf{S}_B=\mathbf{X}\left(\mathbf{H}-\mathbf{L}\right)\mathbf{X}^{\mathrm{T}}$ can have full rank and be positive semidefinite and thus the problem may be solved easily using the Rayleigh-Ritz theorem.

[2] We note that as long as the dimension of the low-dimensional representation of data d (or the number of clusters in clustering applications) is less than the number of data points, the ranks of both sides of the equality $\mathbf{G}\left(\mathbf{H}-\mathbf{L}\right)\mathbf{G}^{\mathrm{T}}=\mathbf{I}_{d\times d}$ may be still equal.

Differentiating with respect to $\mathbf{G}$ gives:

$$\mathbf{GL} = \alpha\mathbf{G}\left(\mathbf{H}{-}\mathbf{L}\right)$$

By left multiplying both sides by $\mathbf{G}^{\mathrm{T}}$ and taking the trace, we note that we must look for a solution with the lowest corresponding Lagrange multiplier α. To proceed, we define a new variable $\beta = \alpha/(\alpha+1)$ and take the transpose of both sides of the above equation to obtain:

$$\mathbf{LG}^{\mathrm{T}}{=}\beta\mathbf{HG}^{\mathrm{T}} \tag{A.1}$$

This is a generalized eigen-decomposition problem based on a single eigenvalue β. To find the solution to this equation, we note that any solution to the ordinary eigen-decomposition problem $\mathbf{LG}^{\mathrm{T}}{=}\beta\mathbf{G}^{\mathrm{T}}$ (which would be a *rank one* solution) may be a solution to A.1, as well. The reason is that all of the eigenvectors of Laplacian matrix $\mathbf{L}$ are orthogonal to the eigenvector $\mathbf{1}$ and thus the sum of their entries equals zero. (It may be seen that eigenvector $\mathbf{1}$ also holds in equation A.1. However, we are not interested in that.) As a result, multiplication by the centering matrix $\mathbf{H}$ leaves them unchanged. Since we are looking for a solution with the corresponding lowest Lagrange multiplier $\alpha = \beta/(1-\beta)$, we have to choose the eigenvector of $\mathbf{L}$ corresponding to the eigenvalue for which the value of $\beta/(1-\beta)$ is a minimum. Since all eigenvalues of $\mathbf{L}$ are assumed to be in the interval $[0, 1)$, this is equivalent to choosing the eigenvector corresponding to the smallest eigenvalue of $\mathbf{L}$.

Bibliography

[1] Babak Alipanahi, Michael Biggs, and Ali Ghodsi. Distance metric learning vs. fisher discriminant analysis. In *AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence*, pages 598–603. AAAI Press, 2008. 40, 59

[2] A. A. Alizadeh, M. B. Eisen, R. E. Davis, C. Ma, I. S. Lossos, A. Rosenwald, J. C. Boldrick, H. Sabet, T. Tran, X. Yu, J.I. Powell, L. Yang, G. E. Marti, T. Moore, J. Hudson Jr, L. Lu, D. B. Lewis, R. Tibshirani, G. Sherlock, W. C. Chan, T. C. Greiner, D. D. Weisenburger, J. O. Armitage, R. Warnke, R. Levy, W. Wilson, M. R. Grever, J. C. Byrd, D. Botstein, P. O. Brown, and L. M. Staudt. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, 2003. 8, 36

[3] U Alon, N Barkai, DA Notterman, K Gish, S Ybarra, D Mack, and AJ Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Science of the United States of America*, 96(12):6745–6750, JUN 8 1999. 36

[4] Edoardo Amaldi and Viggo Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems, 1997. 13

[5] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003. 41

[6] Bhattacharjee. Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses. *Proceedings of the National Academy of Science of the United States of America*, 98(24):13790–13795, NOV 20 2001. 36

[7] M. Biggs, A. Ghodsi, and S. Vavasis. Nonnegative matrix factorization via rank-one downdate. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 64–71, New York, NY, USA, 2008. ACM. 26, 30, 32

[8] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. 36

[9] FS Collins, ES Lander, J Rogers, RH Waterston, and Int Human Genome Sequencing Conso. Finishing the euchromatic sequence of the human genome. *NATURE*, 431(7011):931–945, 2004. 3

[10] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995. 9

[11] T. Cox and M. Cox. *Multidimensional Scaling*. Chapman Hall, Boca Raton, 2nd edition, 2001. 41, 44

[12] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990. 30

[13] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means, spectral clustering and normalized cuts, 2004. 39

[14] R. Dìaz-Uriarte and S. Alvarez de Andrés. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7, 2006. 10

[15] S. Draghichi. *Data Analysis Tools for DNA Microarrays*. Chapman & Hall/CRC, 2003. 11

[16] Rine Dudoit, Jane Fridly, and Terence P. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 97:77–87, 2002. 37

[17] RA Fisher. The use of multiple measurements in taxonomic problems. *ANNALS OF EUGENICS*, 7:179–188, 1936. 40

[18] Keinosuke. Fukunaga. *Introduction to statistical pattern recognition.* Academic Press, New York,, 1972. 58

[19] Ali Ghodsi. Dimensionality reduction, a short tutorial. *Department of Statistics and Actuarial Science, University of Waterloo*, 2006. 41

[20] Amir Globerson and Sam Roweis. Metric learning by collapsing classes, 2006. 62

[21] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 17*, pages 513–520. MIT Press, 2004. 61

[22] TR Golub. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, OCT 15 1999. 10, 36

[23] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf. Measuring statistical dependence with Hilbert-Schmidt norms. In *Algorithmic Learning Theory, 16th International Conference, ALT 2005, Singapore, October 2005, Proceedings*, volume 3734 of *Lecture Notes in Artificial Intelligence*, pages 63–77. Springer, October 2005. 25, 26, 27

[24] A. Gretton, K. Fukumizu, C. Hui Teo, L. Song, B. Schölkopf, and Alex Smola. A kernel statistical test of independence. In *Advances in Neural Information Processing Systems 20*, pages 585–592. MIT Press, 2008. 25

[25] I. Guyon and Elisseeff A. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003. 10

[26] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46(1-3), 2002. 10, 12

[27] J Khan, JS Wei, M Ringner, LH Saal, M Ladanyi, F Westermann, F Berthold, M Schwab, CR Antonescu, C Peterson, and PS Meltzer. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7(6):673–679, JUN 2001. 36

[28] R. Kohavi and G.H. John. An introduction to variable and feature selection. *Artif Intel*, 1(2):273–324, 1997. 9

[29] T Kohonen and P Somervuo. Self-organizing maps of symbol strings. *NEUROCOMPUTING*, 21(1-3):19–30, 1998. 8

[30] D. Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999. 7

[31] H. Lütkepohl. *Handbook of Matrices*. Wiley, New York, 1997. 6, 29, 42

[32] S. Ma and J. Huang. Regularized roc method for disease classification and biomarker selection with microarray data. *Bioinformatics*, 21:43564362, 2005. 10

[33] Sebastian Mika, Jihun Ham, Jihun Ham, Daniel D. Lee, Daniel D. Lee, Mika Sebastian, Bernhard Schlkopf, and Bernhard Schlkopf. A kernel view of the dimensionality reduction of manifolds, 2003. 39, 41

[34] F Rosenblatt. The Perceptron: A Perceiving and Recognizing Automaton. *Technical Report*, 86-460-1, 1957. 9

[35] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000. 41, 42

[36] Yvan Saeys, Inaki Inza, and Pedro Larranaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 2007. 9, 10

[37] L. Saul and S. Roweis. Think globally, fit locally: Unsupervised learning of nonlinear manifolds. *JMLR*, 2003. 41

[38] D Singh. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1(2):203–209, MAR 2002. 36

[39] Sheila K. Singh, Cynthia Hawkins, Ian D. Clarke, Jeremy A. Squire, Jane Bayani, Takuichiro Hide, R. Mark Henkelman, Michael D. Cusimano, and Peter B. Dirks. Identification of human brain tumour initiating cells. *Nature*, 432(7015):396–401, November 2004. 15, 16

[40] Le Song, Justin Bedo, Karsten M. Borgwardt, Arthur Gretton, and Alex Smola. Gene selection via the BAHSIC family of algorithms. *Bioinformatics*, 23(13), 2007. 26

[41] Le Song, Alex Smola, Arthur Gretton, and Karsten M. Borgwardt. A dependence maximization view of clustering. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, 2007. 39, 41, 48

[42] Le Song, Alex Smola, Arthur Gretton, and Karsten M. Borgwardt. Supervised feature selection via dependence estimation, 2007. 26

[43] T. P. Speed. *Statistical analysis of gene expression microarray data.* Chapman & Hall/CRC, Boca Raton, Fla., 2003. 9

[44] G. W. Stewart. On the early history of the singular value decomposition. *SIAM Review*, 35:551–566, 1993. 30

[45] AI Su, JB Welsh, LM Sapinoso, SG Kern, P Dimitrov, H Lapp, PG Schultz, SM Powell, CA Moskaluk, HF Frierson, and GM Hampton. Molecular classification of human carcinomas by use of gene expression signatures. *Cancer Research*, 61(20):7388–7393, OCT 15 2001. 36

[46] E.K. Tang, P.N. Suganthan, and X. Yao. Gene selection algorithms for microarray data based on least squares support vector machine. *BMC Bioinformatics*, 7, 2006. 10

[47] J. Tenenbaum. Mapping a manifold of perceptual observations. In *Advances in Neural Information Processing Systems 10*, pages 682–687, 1998. 41

[48] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000. 41

[49] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994. 30

[50] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journl of Machine Learning Research*, 9:2579–2605, 2008. 63

[51] Ulrike von Luxburg. A tutorial on spectral clustering. *Technical Report, Max Plank Institute for Biologocal Cybernetics*, 2006. 51

[52] Jason Weston. *SPIDER: a machine learning toolbox (Matlab)*, 2003. 36

[53] Jason Weston, Andr Elisseeff, Bernd Schlkopf, and Mike Tipping. Use of the zero-norm with linear models and kernel methods, 2002. 12, 13, 14

[54] X. Zhou and K.Z. Mao. Ls bound based gene selection for dna microarray data. *Bioinformatics*, 21:1559–1564, 2005. 10