SUPERVISED MACHINE LEARNING MODELS FOR FEATURE SELECTION AND CLASSIFICATION ON HIGH DIMENSIONAL DATASETS

By

VIJAY SUNDER NAGA PAPPU

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY UNIVERSITY OF FLORIDA 2013
Mom, Dad and Manasa – This is for you.
ACKNOWLEDGMENTS

I would like to acknowledge the support of several people that have made my Ph.D. possible at University of Florida. Firstly, the emotional support and encouragement of my wife is invaluable as it helped me to realize my dream of Ph.D. The patience and support of my parents has always been one of my assets. I would like to thank my academic advisor Dr. Panos M. Pardalos for providing me opportunities to work on interesting research problems. I would also like to thank Dr. Pando G. Georgiev for the technical discussions.

I have greatly enjoyed working with Mike, Vladimir, Paul, Angelina, Orestis and Jason on several challenging research projects. I would also like to acknowledge the valuable advice that I received from Syed, Dmytro and Petros during my doctoral program. I would also like to thank other friends from the department: Ayse, Chrysafis, Jose, Kun, Oleg, Deon and Jing. Further, I would like to extend my thanks to Anuj, Kasi, Sam, Prateek and Imran for their confidence and support towards me.

Finally, I would like to thank Dr. Cole Smith and ISE department for admitting me into the doctoral program. Last, but certainly not the least, I would like to thank Dr. Anand Rangarajan, Dr. Petar Momcilovic and Dr. Vladimir Boginski for serving in my Ph.D. committee and providing valuable suggestions on improving my work.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>4</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>7</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>8</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>9</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>10</td>
</tr>
<tr>
<td>1.1 Challenges of High Dimensional Data Spaces</td>
<td>11</td>
</tr>
<tr>
<td>1.2 Dimensionality reduction</td>
<td>12</td>
</tr>
<tr>
<td>1.2.1 Feature extraction</td>
<td>13</td>
</tr>
<tr>
<td>1.2.2 Feature selection</td>
<td>13</td>
</tr>
<tr>
<td>1.3 Motivation and Significance</td>
<td>14</td>
</tr>
<tr>
<td>1.4 Thesis goal and Structure</td>
<td>15</td>
</tr>
<tr>
<td>2 OVERVIEW OF FEATURE SELECTION TECHNIQUES</td>
<td>16</td>
</tr>
<tr>
<td>2.1 Filter Methods</td>
<td>18</td>
</tr>
<tr>
<td>2.2 Wrapper Methods</td>
<td>20</td>
</tr>
<tr>
<td>2.3 Embedded Methods</td>
<td>21</td>
</tr>
<tr>
<td>3 OVERVIEW OF CLASSIFICATION TECHNIQUES</td>
<td>23</td>
</tr>
<tr>
<td>3.1 Support Vector Machines</td>
<td>23</td>
</tr>
<tr>
<td>3.2 Discriminant Functions</td>
<td>30</td>
</tr>
<tr>
<td>3.3 Hybrid Classifiers</td>
<td>40</td>
</tr>
<tr>
<td>3.4 Ensemble Classifiers</td>
<td>41</td>
</tr>
<tr>
<td>4 SPARSE PROXIMAL SUPPORT VECTOR MACHINES</td>
<td>48</td>
</tr>
<tr>
<td>4.1 Proximal Support Vector Machines (PSVMs)</td>
<td>48</td>
</tr>
<tr>
<td>4.2 Sparse Proximal Support Vector Machines (sPSVMs)</td>
<td>53</td>
</tr>
<tr>
<td>4.3 Numerical Experiments</td>
<td>54</td>
</tr>
<tr>
<td>4.3.1 Category I Datasets</td>
<td>56</td>
</tr>
<tr>
<td>4.3.2 Category II Datasets</td>
<td>61</td>
</tr>
<tr>
<td>5 COMBINING FEATURE SELECTION AND SUBSPACE LEARNING</td>
<td>64</td>
</tr>
<tr>
<td>5.1 $l_{2,1}$-norm</td>
<td>64</td>
</tr>
<tr>
<td>5.2 Joint Feature Selection and Subspace Learning</td>
<td>65</td>
</tr>
<tr>
<td>5.3 Current Approach</td>
<td>66</td>
</tr>
<tr>
<td>5.4 Alternate Approach</td>
<td>69</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4-1</td>
<td>Datasets used in numerical experiments of sPSVMs</td>
</tr>
<tr>
<td>4-2</td>
<td>Test accuracy comparison of sPSVMs for Category I datasets</td>
</tr>
<tr>
<td>4-3</td>
<td>Top ten selected features for each class in colon dataset</td>
</tr>
<tr>
<td>4-4</td>
<td>Average Jaccard Index comparison of sPSVMs for Category I datasets</td>
</tr>
<tr>
<td>4-5</td>
<td>CV accuracy comparison of sPSVMs for Category II datasets</td>
</tr>
<tr>
<td>5-1</td>
<td>Datasets used in numerical experiments of JSPCA</td>
</tr>
<tr>
<td>5-2</td>
<td>Accuracy comparison between PCA and JSPCA for spambase dataset</td>
</tr>
<tr>
<td>5-3</td>
<td>Accuracy comparison between PCA and JSPCA for WPBC dataset</td>
</tr>
<tr>
<td>5-4</td>
<td>Accuracy comparison between PCA and JSPCA for ionosphere dataset</td>
</tr>
<tr>
<td>5-5</td>
<td>Accuracy comparison between PCA and JSPCA for mushroom dataset</td>
</tr>
<tr>
<td>6-1</td>
<td>Comparison between LSC and CSC on two example datasets</td>
</tr>
<tr>
<td>7-1</td>
<td>Characteristics of the five ATCC breast cell lines studied</td>
</tr>
<tr>
<td>7-2</td>
<td>Sensitivity, specificity and average classification accuracy of FFS-SVM</td>
</tr>
<tr>
<td>7-3</td>
<td>Performance comparison among SVM, PCA-SVM, PCA-LDA and FFS-SVM</td>
</tr>
<tr>
<td>7-4</td>
<td>The top 10 features selected by FFS for the four binary classification tasks.</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-1</td>
<td>Feature frequency plot obtained from sPSVMs for the Colon dataset</td>
<td>58</td>
</tr>
<tr>
<td>4-2</td>
<td>Test accuracies obtained from sPSVMs as a function of $\tau$ for Category I datasets</td>
<td>59</td>
</tr>
<tr>
<td>4-3</td>
<td>Features selected from colon dataset as a function of $B$ for different values of $\tau$</td>
<td>61</td>
</tr>
<tr>
<td>6-1</td>
<td>Data and corresponding subspaces from LSC and CSC in example 1</td>
<td>84</td>
</tr>
<tr>
<td>6-2</td>
<td>Data and corresponding subspaces from LSC and CSC in example 2</td>
<td>84</td>
</tr>
<tr>
<td>6-3</td>
<td>LOOCV accuracies as a function of $\delta$ for different $k$ obtained from CSC</td>
<td>89</td>
</tr>
<tr>
<td>7-1</td>
<td>Spectral overlay of MCF10A cell line</td>
<td>96</td>
</tr>
<tr>
<td>7-2</td>
<td>Spectral preprocessing of Raman spectra</td>
<td>97</td>
</tr>
<tr>
<td>7-3</td>
<td>Peak finding in FFS</td>
<td>100</td>
</tr>
<tr>
<td>7-4</td>
<td>Peak coalescing in FFS</td>
<td>101</td>
</tr>
<tr>
<td>7-5</td>
<td>Hierarchical clustering of mean spectra of five cell lines</td>
<td>103</td>
</tr>
<tr>
<td>7-6</td>
<td>Mean spectra of the five cell lines</td>
<td>104</td>
</tr>
<tr>
<td>7-7</td>
<td>Classification accuracy comparison among RF-SVM, FFS-SVM, and PCA-SVM</td>
<td>106</td>
</tr>
</tbody>
</table>
Abstract of Dissertation Presented to the Graduate School of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

SUPERVISED MACHINE LEARNING MODELS FOR FEATURE SELECTION AND CLASSIFICATION ON HIGH DIMENSIONAL DATASETS

By

Vijay Sunder Naga Pappu

December 2013

Chair: Panos M. Pardalos
Major: Industrial and Systems Engineering

High dimensional datasets are currently prevalent in many applications due to significant advances in technology over the past decade. High dimensional datasets are generally characterized by large number of features and relatively lesser number of samples. Classification tasks on such datasets pose significant challenges to the standard statistical methods and render many existing classification techniques impractical due to curse of dimensionality. Classification performance is improved by reducing the dimensionality using feature selection and/or feature extraction.

In this work, we focus on building scalable efficient classification models for high dimensional datasets that would also be able to extract important features in addition to accurately predicting the class of unknown samples. In this regard, we begin with an overview of feature selection techniques and classification models and then proceed to introduce variants of existing methods that improve their generalization ability on high dimensional datasets. We also study a high-dimensional data application involving Raman Spectroscopy and propose a novel hierarchical classification framework to classify Breast cancer cells from non-cancer cells.
CHAPTER 1
INTRODUCTION

In the past decade, technological advances have had a profound impact on society and the research community [74]. High volume throughput data can be collected simultaneously and at relatively low cost in many applications. Often is the case, each observation is characterized with thousands of variables or features. For example, in the medical domain, huge numbers of magnetic resonance images (MRI) and functional MRI data are collected for each subject, where the intensity at each pixel may constitute a feature. Furthermore, every sample in gene-expression microarray datasets consist of measurements from thousands of genes [28]. Various types of spectral measurements including Mass Spectroscopy and Raman Spectroscopy are very common in chemometrics, where the recorded spectra number well into the thousands [48]. In many such biomedical applications, the measurements tend to be very expensive and the number of samples for such datasets are on the order of tens, or maybe low hundreds. These datasets, often termed as High Dimension Low Sample Size (HDLSS) datasets, are characterized with large number of features $p$ and a relatively small number of samples $n$; with $p >> n$ [174]. Such high dimensional datasets arising in many fields create golden opportunities as well as significant challenges for the advancement of mathematical sciences.

Classification is a supervised machine learning technique that maps some combination of input variables, into predefined categorical outputs, also sometimes called class labels. Classification models infer the mapping function by learning from a training set of examples. Each training example is a pair consisting of an input vector and a desired class label. The inferred mapping function is then used to predict the class label information of new examples. Classification problems occur in several fields of Science and technology like discriminating cancerous cells from non-cancerous cells, web document classification, categorizing images in remote sensing applications among
many others. Several algorithms starting with Neural Networks [73], Logistic Regression [90], Linear Discriminant Analysis [111], Support Vector Machines [165] and more recent ensemble methods like Boosting [54] and Random Forests [17], have been proposed to solve the classification problem in different contexts. However, classification tasks on HDLSS datasets pose significant challenges to the standard statistical methods and render many existing classification techniques impractical [85]. Below, some of the inherent difficulties of high dimensional spaces are discussed that pose significant challenges to existing classification methods.

1.1 Challenges of High Dimensional Data Spaces

**Poor generalization ability:** One challenge for modeling in high dimensional data spaces is to avoid overfitting the training data [28]. Classification models, in addition to performing well on the training set, are required to perform equally well on an independent testing set and hence it is imperative for these models to have good generalization ability. However, often the small number of samples in high dimensional data settings cause the classification model to overfit to the training data, thereby having poor generalization ability for the model. The accuracy of classification algorithms tends to deteriorate in high dimensions due a phenomenon called the curse of dimensionality [94]. This phenomenon is illustrated for classification by an example in [161]. Consider two equally probable, normally distributed classes with common variance. Let the mean of class 1 be \(1/k^{1/2}\) and the mean of class 2 be \(-1/k^{1/2}\), where \(k = 1, 2, 3, \ldots\) indicates feature index. Thus the discriminative power of each feature decreases as \(k\) increases. The author evaluated error rates for the Bayes' decision rule as a function of \(k\). The variance is assumed to be known while the class means are estimated based on the finite data set. The following observations were made:

- as the number of features tend to infinity, the test error reduces to random guessing;
- a finite number of features are needed to achieve the best test error; and
• the optimal dimensionality increases with increasing sample size. Thus, the ability of an algorithm to converge to a true model deteriorates rapidly with increasing feature dimensionality thus leading to poor generalization ability.

**Geometrical distortion of high dimensional data spaces:** Several earlier studies have revealed the geometrical distortion of high dimensional data spaces. Authors in [8] attempted to quantify this distortion. They argued that under reasonable assumptions, the ratio of distances between the nearest and the farthest to a given target is almost 1. They also show that this ratio remains constant for a wide variety of data distributions and distance functions. In such a case, the nearest neighbor problem becomes ill-defined, as the concept of proximity may not be meaningful from a qualitative perspective. A similar study in [94] show that the distance of a randomly selected point in a hypercube with diagonal length 1 from one corner approaches a constant value of 0.58 as the dimension $p$ increases, thus showing again that distance-based classifiers like kNN might perform poorly.

**Unreliable parameter estimation:** Several classification models like discriminant analysis and its variants, require knowing the class covariances $\Sigma$ *apriori*, in order to establish a discriminant rule for classification problems. However, in many problems, since the covariance is not known *apriori*, researchers often attempt to estimate the covariance from the sample data. However such estimates for high dimensional datasets are unreliable since an unknown covariance matrix for observations on $p$ variables in theory has $p(p + 1)/2$ unknown parameters and the sample covariance matrix is a poor estimator when $p$ is large. Additionally, since the sample covariance matrix is rank-deficient, estimating quantities like inverse would be unreliable thus leading to unpredictable performance on independent test datasets.

### 1.2 Dimensionality reduction

One common approach to address the aforementioned challenges involves reducing the dimensionality of the dataset either using feature extraction [104] and/or
feature selection [142] prior to classification. These dimensionality reduction techniques decrease the complexity of the classification model and thus improve the classification performance [142].

1.2.1 Feature extraction

Feature extraction, also known as subspace learning, transform the input data into a set of meta-features which extract relevant information from the input data for classification. Several feature extraction techniques such as Principal Component Analysis has been successfully applied in many high dimensional data applications [107]. However, in many biomedical applications, these techniques offer limited model interpretability to extract biomarker-type information as they transform the input space during the process [48].

1.2.2 Feature selection

Feature selection techniques, in contrast to feature extraction techniques, do not alter the input data space, but merely select a subset of features based on some optimality criteria [142]. Though feature extraction techniques generalize better when combined with classification models, feature selection techniques offer the advantage of interpretability by a domain expert as they preserve the original feature space. In the context of classification, feature selection methods can be broadly organized into three categories: filter methods, wrapper methods and embedded methods [142].

Filter methods assess feature relevance from the intrinsic properties of the data. In most cases the features are ranked using a feature relevance score where a high score indicates a greater discriminative power. The low-scoring features are generally removed. The reduced data obtained from considering only the selected features are then presented as an input to the classification algorithm. Additionally, feature ranking helps in limiting the number of features for classification depending on the sample sizes, thereby avoiding the curse of dimensionality and poor generalization ability. Contrary to filter methods, the wrapper methods integrate the classifier hypothesis search and
the feature subset search. In this framework, each method is tailored to a specific classification algorithm, where a search procedure in the feature space is first defined, and various subsets of features are generated and evaluated by training and testing the classification model [92, 142]. Similar to wrapper methods, embedded methods are also specific to a given classification model and integrate the search for an optimal subset of features with classifier building.

Filter methods offer several advantages including scalability to high dimensional datasets, computational efficiency, and independence from the classification algorithm. This independency offers the advantage of performing feature selection only once and then evaluating different classifiers. The wrapper and embedded methods consider feature dependencies and include interactions between feature subset search and learning model selection. However, a common drawback of both these methods is that they have an increased risk of overfitting and can also be computationally intensive depending on the computational complexity of the classification model.

1.3 Motivation and Significance

Due to the aforementioned challenges in high dimensional datasets, several standard classification methods like logistic regression, linear discriminant analysis, kNN classifiers etc., are known to perform poorly on different classification tasks. However, with the continuous advancements in technology, the measurements collected from each sample would only increase, thereby making it imperative to build scalable and efficient classification models and algorithms that would perform well on such datasets. Additionally, in many biomedical applications, in addition to building an accurate classification model with good generalization ability, it is also important to understand the features that contribute to the differences among the classes [48]. For example, while building a classification model for classifying normal patients from cancer patients using gene expression data, it is important to extract and analyze the most
important genes that contribute to the differences among the classes. Such an analysis could lead to the discovery of biomarkers in many biomedical applications.

1.4 Thesis goal and Structure

In this work our goal is to build scalable efficient classification models for high dimensional datasets that would also be able to extract important features in addition to accurately predicting the class of unknown samples. The rest of the work is organized as follows: Chapters 2 & 3 discuss standard feature selection and classification techniques and their applications to high dimensional datasets. In Chapter 4, we extend the classification technique proximal support vector machines to perform feature selection on high dimensional datasets. Chapter 5 explores the possibility of combining feature selection and feature extraction to take advantage of either techniques. Constrained subspace classifiers for high dimensional datasets are discussed in Chapter 6. A high dimensional data application of characterizing Breast cancer cells using Raman Spectroscopy is discussed in Chapter 7. We discuss further research directions and provide some conclusions in Chapter 8.
Recently, feature selection has been an active area of research among many researchers due to tremendous advances in technology enabling collecting samples with thousands of measurements in a single experiment. Feature selection techniques select a subset of features based on some optimality criteria [142]. Irrespective of the measure of optimality, the selected subset of features should ideally possess the following characteristics [66]:

- an optimally minimal number of features should be selected to accurately predict the class of unknown samples,
- the prediction accuracy of the classifier run on data with only selected subset of features should be better than running the classifier on data containing all features,
- The resulting class distribution from the selected features should be as close as possible to the original class distribution given all feature values.

Based on the above feature characteristics, it is obvious that irrelevant features would not be part of the optimal set of features, where an irrelevant feature with respect to the target class is defined as follows [173]:

Let $F$ be the full set of features and $C$ be the target class. Define $F_i \in F$ and $S_i = F - F_i$.

**Definition 1: (Irrelevance)** A feature $F_i$ is irrelevant if and only if:

$$P(C|F_i, S_i') = P(C|S_i'), \quad \forall S_i' \subseteq S_i \quad (2-1)$$

Irrelevance simply means that it is not necessary for classification since the class distribution resulting from any subset of other features does not change after eliminating the irrelevant feature.

The definition of relevance is not as straightforward as irrelevance. There have been several definitions for relevance in the past, however Kohavi et al. [92] argued that the earlier definitions weren’t adequate to accurately classify the features. Hence,
they defined relevance in terms of an optimal Bayes’ classifier. A feature $F_i$ is strongly relevant if removal of $F_i$ alone will result in decrease of classification performance of an optimal Bayes’ classifier. A feature $F_i$ is weakly relevant if it is not strongly relevant and there exists a subset of features, $S'_i$, such that the performance of a Bayes’ classifier on $S'_i$ is worse than the performance on $S'_i \cup \{F_i\}$.

**Definition 2: (Strong relevance)** A feature $F_i$ is strongly relevant if only and if:

$$P(C|F_i, S'_i) \neq P(C|S'_i), \quad S'_i \subseteq S_i$$  \hspace{1cm} (2–2)

**Definition 3: (Weak relevance)** A feature $F_i$ is weakly relevant if only and if:

$$P(C|F_i, S_i) = P(C|S_i) \quad \text{and,} \quad \exists S'_i \subseteq S_i, \quad P(C|F_i, S'_i) \neq P(C|S'_i)$$  \hspace{1cm} (2–3)

Strong relevance implies that the feature is indispensable and is required for an optimal set, while weak relevance implies that the feature may be required sometimes to improve the prediction accuracy. From this, one may conclude that the optimal set should consist of all the strongly relevant features, some of the weakly irrelevant features and none of the irrelevant features. However, the definitions do not explicitly mention which of the weakly relevant features should be included and which of them excluded. Hence, Yu *et al.* [173] claim that the weakly relevant features should be further classified to discriminate among the redundant features and the non-redundant features, since earlier research efforts showed that along with irrelevant features, redundant features also adversely effect the classifier performance. Before we provide definitions, we introduce another concept called feature’s Markov Blanket as defined by Koller *et al.* [93].

**Definition 4: (Markov Blanket)** Given a feature $F_i$, let $M_i \subset F (F_i \notin M_i)$, $M_i$ is said to be a Markov blanket for $F_i$ if only and if:

$$P(F - M_i - \{F_i\}, C|F_i, M_i) = P(F - M_i - \{F_i\}, C|M_i)$$  \hspace{1cm} (2–4)
The Markov blanket $M_i$ could be imagined as a blanket for the feature $F_i$ that subsumes not only the information that $F_i$ possesses about target class $C$, but also about other features. It is also important to note that the strongly relevant features cannot have a Markov Blanket. Since, the irrelevant features do not contribute to classification, Yu et al. [173] further classified the weakly relevant features into either redundant or non-redundant using the concept of Markov blanket.

**Definition 5: (Redundant feature)** Given a set of current features $G$, a feature is redundant and hence should be removed from $G$ if and only if it has a Markov Blanket within $G$.

From the above definitions, it is clear that the optimal set of features should consist of all of the strongly relevant features and the weakly relevant non-redundant features. However an exhaustive search over the feature space is intractable since there are $2^p$ possibilities with $p$ being the number of features. Hence, over the past decade, several heuristic and approximate methods have been developed to perform feature selection. Feature selection techniques, in the context of classification, can be organized into three categories: filter methods, wrapper methods and embedded methods [142].

### 2.1 Filter Methods

Filter methods assess feature relevance from the intrinsic properties of the data. In most cases the features are ranked using a feature relevance score where a high score indicates a greater discriminative power. The low-scoring features are generally removed. The reduced data obtained from considering only the selected features are then presented as an input to the classification algorithm. Filter techniques offer several advantages including scalability to high dimensional datasets, being computationally efficient, and are independent of classification algorithm. This independency offers the advantage of performing feature selection only once and then evaluating different classifiers.
Some univariate filter techniques perform simple hypothesis testing like Chi-Square ($\chi^2$) test or t-test to eliminate the irrelevant features, while other techniques estimate information theoretic measures like information gain and gain-ratio to perform the filtering process [7]. Although these techniques are simple, fast and highly scalable, they ignore feature dependencies which may lead to worse classification performance as compared with other feature selection techniques. In order to account for feature dependencies, a number of multivariate filter techniques were introduced. The multivariate filter methods range from accounting for simple mutual interactions [12] to more sophisticated solutions. One such technique called Correlation based Feature Selection (CFS) introduced by Hall (1999) [69], evaluates a subset of features by considering the discriminative power of each feature in addition to the degree of redundancy between them:

$$CFS_S = \frac{k\Phi_{cf}}{\sqrt{k + k(k - 1)\Phi_{ff}}}$$  \hspace{1cm} (2–5)$$

where $CFS_S$ is the score of a feature subset $S$ containing $k$ features, $\Phi_{cf}$ is the average feature to class correlation ($f \in S$), and $\Phi_{ff}$ is the average feature to feature correlation. Unlike the univariate filter methods, CFS presents a score for a subset of features. Since, exhaustive search is intractable, several heuristic techniques like greedy hill-climbing or best-first search have been proposed to find the feature subset with highest CFS score.

Another important multivariate filter method called the Markov blanket filtering was introduced by Koller et. al [93]. The idea here being that once we find a Markov blanket of feature $F_i$ in a feature set $G$, we can safely remove $F_i$ from $G$ without compromising on the class distribution. Since estimating the Markov blanket for a feature is hard, Koller et. al [93] propose a simple iterative algorithm that starts with the full feature set $F = G$ and then repeatedly eliminates one feature at a time based on cross-entropy of each feature until a pre-selected number of features are removed.
Koller et al [93] further prove that in such a sequential elimination process in which unnecessary features are removed one by one, a feature tagged as unnecessary based on the existence of a Markov blanket $M_i$ remains unnecessary in later stages when more features have been removed. Also, the authors claim that the process removes all the irrelevant as well as redundant features. Several variations to the Markov blanket filtering method like Grow-Shrink (GS) algorithms, Incremental Association Markov Blanket (IAMB), Fast-IAMB and recently $\lambda$-IAMB have been proposed by other authors [57]. There are other interesting multivariate filter methods like Fast-Correlation based Feature Selection (FCBF) [172], Minimum Redundancy-Maximum Relevance (MRMR) [41], and Uncorrelated Shrunken Centroid (USC) [171] algorithms proposed in literature.

### 2.2 Wrapper Methods

As seen in the earlier section, filter methods treat the problem of finding a good feature subset independently of the classifier building step. Wrapper methods, on the other hand, integrate the classifier hypothesis search within the feature subset search. In this framework, each method is tailored to a specific classification algorithm, where a search procedure in the feature space is first defined, and various subsets of features are generated and evaluated by training and testing the classification model [92, 142]. Advantages of wrapper methods include consideration of feature dependencies and the ability to include interactions between the feature subset search and model selection. A common drawback includes the risk of overfitting than the filter methods and could be computationally intensive if the classification model especially has a high computational cost.

The wrapper methods generally employ a search algorithm in order to search through the space of all feature subsets. The search algorithm is wrapped around the classification model which provides a feature subset that can be evaluated by the classification algorithm. As mentioned earlier, since an exhaustive search is not practical, heuristic search methods are used to guide the search. These search
methods can be broadly classified as deterministic and randomized search algorithms. Deterministic search methods include a set of sequential search techniques like the Sequential Forward Selection [89], Sequential Backward Selection [89], Plus-l Minus-r Selection [50], Bidirectional Search, Sequential Floating Selection [134] etc., where the features are either sequentially added or removed based on some criterion measure. Randomized Search algorithms include popular techniques like Genetic Algorithms [33], Simulated Annealing [88], Randomized Hill Climbing [149], etc.

2.3 Embedded Methods

Embedded methods integrate the search for an optimal subset of features into the classifier construction and can be seen as a search in the combined space of feature subsets and hypotheses. Similar to wrapper methods, embedded approaches are also specific to a given learning algorithm and include the interaction with the classification model, but unlike the wrapper methods, also has the advantage to be less computationally intensive [142].

Recently embedded methods have gained importance among the research community due to its advantages. The embedded characteristic of several classifiers to eliminate features irrelevant to classification and thus select a subset of features, has been exploited by several authors. Examples include the use of random forests (discussed later) in an embedded way to calculate the importance of each feature [39, 82]. Another line of embedded feature selection techniques uses the weights of each feature in linear classifiers, such as SVM [67] and logistic regression [108]. These weights are used as a measure of relevance of each feature, and thus allow for the removal of features with very small weights. Also, recently regularized classifiers like Lasso and elastic-net have also been successfully employed in performing feature selection in microarray gene analysis [175]. Another interesting technique called feature selection via sparse SVM has been recently proposed by Tan et al. [153]. This technique called the Feature generating Machine (FGM) adds a binary variable for every
feature in the sparse formulation of SVM via $l_0$-norm and and the authors propose a cutting plane algorithm combined with multiple kernel learning to efficiently solve the convex relaxation of the optimization problem.
CHAPTER 3
OVERVIEW OF CLASSIFICATION TECHNIQUES

In this chapter, we present standard classification techniques and discuss their extensions in applying to high dimensional datasets. We begin the discussion with Support Vector Machines and its variants and proceed to discriminant functions and their modifications for high dimensional datasets. We then introduce hybrid classifiers followed by ensemble methods and their applications.

3.1 Support Vector Machines

Hard-Margin Support Vector Machines: In the last decade, Support Vector Machines (SVM) have attracted the attention of many researchers with successful application to several classification problems in bioinformatics, finance and remote sensing among many others. Standard SVM construct a hyperplane, also known as decision boundary, that best divides the input space $\chi$ into two disjoint regions. The hyperplane $f : \chi \rightarrow \mathbb{R}$, is estimated from the training set $S$. The class membership for an unknown sample $x \in \chi$ can be based on the classification function $g(x)$ defined as:

$$g(x) = \begin{cases} -1, & f(x) < 0 \\ 1, & f(x) > 0 \end{cases} \quad (3-1)$$

Consider a binary classification problem with the training set $S$ defined as:

$$S = \{(x_i, y_i) | x_i \in \mathbb{R}^p, y_i \in \{-1, 1\}, \quad i = 1, 2, ..., n \} \quad (3-2)$$

where $y_i$ is either -1 or 1 depending on the class that each $x_i$ belongs to. Assume that the two classes are linearly separable and hence there exists at least one hyperplane that separates the training data correctly. A hyperplane parametrized by the normal vector $w \in \mathbb{R}^p$ and bias $b \in \mathbb{R}$ is defined as:

$$\langle w, x \rangle - b = 0 \quad (3-3)$$
where the inner product $\langle \cdot, \cdot \rangle$ is defined on $\mathbb{R}^p \times \mathbb{R}^p \rightarrow \mathbb{R}$. The training set $S$ satisfies the following linear inequality with respect to the hyperplane:

$$y_i(\langle w, x_i \rangle - b) \geq 1 \quad \forall i = 1, 2, \ldots, n$$

(3–4)

where the parameters $w$ and $b$ are chosen such that the distance between the hyperplane and the closest point is maximized. This geometrical margin can be expressed by the quantity $\frac{1}{||w||}$. Hence, for linearly separable set of training points, SVM can be formulated as linearly constrained quadratic convex optimization problem given as:

$$\begin{align*}
\text{minimize} & \quad ||w||^2 \\
\text{subject to} & \quad y_i(\langle w, x_i \rangle - b) \geq 1 \quad \forall i = 1, 2, \ldots, n
\end{align*}$$

(3–5)

This classical convex optimization problem can be rewritten (using the Lagrangian formulation [14]) into the following dual problem:

$$\begin{align*}
\text{maximize} & \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\langle x_i, x_j \rangle) \\
\text{subject to} & \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad \text{and}, \quad \alpha_i \geq 0, \quad i = 1, 2, \ldots, n
\end{align*}$$

(3–6)

where the Lagrange multipliers $\alpha_i$ ($i = 1, 2, \ldots, n$) expressed in (3–6) can be estimated using quadratic programming (QP) methods [37]. The optimal hyperplane $f$ can then be estimated using the Lagrange multipliers obtained from solving (3–6) and the training samples, i.e.,

$$f(x) = \sum_{i \in S} \alpha_i y_i (\langle x \cdot x_i \rangle) - b$$

(3–7)

where $S$ is the subset of training samples called support vectors that correspond to non-zero Lagrange multipliers $\alpha_i$. Support vectors include the training points that exactly satisfy the inequality in (3–5) and lie at a distance equal to $\frac{1}{||w||}$ from the optimal separating hyperplane. Since the Lagrange multipliers are non-zero only for the support vectors and zero for other training samples, the optimal hyperplane in (3–7) effectively
consists of contributions from the support vectors. It is also important to note that the Lagrange multipliers $\alpha_i$ qualitatively provide relative weight of each support vector in determining the optimal hyperplane.

The convex optimization problem in (3–5) and the corresponding dual in (3–6) converge to a global solution only if the training set is linearly separable. This SVM is called the hard-margin support vector machines.

**Soft-Margin Support Vector Machines:** The maximum-margin objective introduced in the previous subsection to obtain the optimal hyperplane is susceptible to the presence of outliers. Also, it is often difficult to adhere to the assumption of linear separability in real world datasets. Hence, in order to handle non-linearly separable datasets as well as be less sensitive to outliers, soft-margin support vector machines are proposed. The objective cost function in (3–5) is modified to represent two competing measures namely, margin maximization (as in the case of linearly separable data) and error minimization (to penalize the wrongly classified samples). The new cost function is defined as:

$$\Psi(w, \xi) = \frac{1}{2}||w||^2 + C \sum_{i=1}^{n} \xi_i$$  \hspace{1cm} (3–8)

where $\xi$ is the slack variable introduced to account for the non-separability of data, and the constant $C$ represents a regularization parameter that controls the penalty assigned to errors. The larger the $C$ value, the higher the penalty associated to misclassified samples. The minimization of the cost function expressed in (3–8) is subject to the following constraints:

$$y_i(\langle w, x_i \rangle - b) \geq 1 - \xi_i, \quad \forall i = 1, 2, ..., n$$

$$\xi_i \geq 0, \quad \forall i = 1, 2, ..., n$$  \hspace{1cm} (3–9)
The convex optimization problem can then be formulated using (3–8) and (3–9) for the non-linearly separable data as:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} ||w||^2_2 + C \sum_{i=1}^{n} \xi_i \\
\text{subject to} & \quad y_i (\langle w, x_i \rangle - b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i = 1, 2, \ldots, n
\end{align*}
\](3–10)

The optimization problem in (3–10) accounts for the outliers by adding a penalty term \(C \xi_i\) for each outlier to the objective function. The corresponding dual to (3–10) can be written using the Lagrange formulation as:

\[
\begin{align*}
\text{maximize} & \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\
\text{subject to} & \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad \text{and}, \quad 0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, n
\end{align*}
\](3–11)

The quadratic optimization problem in (3–11) can be solved using standard QP techniques [37] to obtain the Lagrange multipliers \(\alpha_i\).

**Kernel Support Vector Machines:** The idea of linear separation between two classes mentioned in the subsections above can be naturally extended to handle nonlinear separation as well. This is achieved by mapping the data through a particular nonlinear transformation into a higher dimensional feature space. Assuming that the data is linearly separable in this high dimensional space, a linear separation, similar to earlier subsections, can be found. Such a hyperplane can be achieved by solving a similar dual problem defined in (3–11) by replacing the inner products in the original space with inner products in the transformed space. However, an explicit transformation from the original space to feature space could be expensive and at times infeasible as well. The kernel method [22] provides an elegant way of dealing with such transformations.
Consider a kernel function $K(\cdot, \cdot)$, satisfying Mercer’s theorem, that equals an inner product in the transformed higher dimensional feature space \cite{112}, i.e.,

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle \quad (3–12)$$

where $\Phi(x_i)$ and $\Phi(x_j)$ correspond to the mapping of data points $x_i$ and $x_j$ from the original space to the feature space. There are several kernel functions defined in literature that satisfy the Mercer’s conditions. One such kernel, called the Gaussian kernel is given by:

$$K(x_i, x) = \exp(-\sigma \|x_i - x\|^2) \quad (3–13)$$

where $\sigma$ is a parameter inversely proportional to the width of the Gaussian radial basis function. Another extensively studied kernel is the polynomial function of order $p$ expressed as:

$$K(x_i, x) = (\langle x_i, x \rangle + 1)^p \quad (3–14)$$

Such kernel functions defined above allow for efficient estimation of inner products in feature spaces without the explicit functional form of the mapping $\Phi$. This elegant calculation of inner products in higher dimensional feature spaces, also called the kernel trick, considerably simplifies the solution to the dual problem. The inner products between the training samples in the dual formulation (3–11) can be replaced with a kernel function $K$ and rewritten as:

$$\maximize_{\alpha \in \mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

subject to $\sum_{i=1}^{n} \alpha_i y_i = 0$, and, $0 \leq \alpha_i \leq C$, $i = 1, 2, \ldots, n$ \quad (3–15)

The optimal hyperplane $f$ obtained in the higher dimensional feature space can be conveniently expressed as a function of data in the original input space as:

$$f(x) = \sum_{i \in S} \alpha_i y_i K(x_i, x) - b \quad (3–16)$$
where $S$ is a subset of training samples with non-zero Lagrange multipliers $\alpha_j$. The shape of $f(x)$ depends on the type of kernel functions adopted.

It is important to note that the performance of kernel-based SVM is dependent on the optimal selection of multiple parameters, including the kernel parameters (e.g., $\sigma$ and $p$ parameters for the gaussian and polynomial kernels, respectively) and the regularization parameter $C$. A simple and successful technique that has been employed involves a grid search over a wide range of the parameters. The classification accuracy of SVM for every pair of parameters is estimated using a leave-one-out cross validation technique and the pair corresponding to the highest accuracy is chosen. Also, some interesting automatic techniques have been developed to estimate these parameters \cite{25, 27}. They involve constructing an optimization problem that would maximize the margin as well as minimize the estimate of the expected generalization error. Optimization of the parameters is then carried out using a gradient descent search over the space of the parameters. Recently, more heuristic based approaches have been proposed to deal with this issue. A continuous version of Simulated Annealing (SA) called Hide and Seek SA was employed in \cite{101} to estimate multiple parameters as well as select a subset of features to improve the classification accuracy. Similar approaches combining Particle Swarm Optimization (PSO) with SVM are proposed in \cite{65} and \cite{102}. Furthermore, a modified Genetic Algorithm (GA) was also implemented along with SVM to estimate the optimal parameters \cite{76}.

**SVM applied to High Dimensional Classification Problems:** SVM have been successfully applied to high dimensional classification problems arising in fields like remote sensing, web document classification, microarray analysis etc. As mentioned earlier, conventional classifiers like logistic regression, maximum likelihood classification etc., on high dimensional data tend to overfit the model using training data and run the risk of achieving lower accuracies on testing data. Hence, a pre-processing step like either feature selection and/or dimensionality reduction techniques are proposed
to alleviate the problem of curse of dimensionality while working with these traditional classifiers. Surprisingly, SVM have been successfully applied to hyperspectral remote sensing images without any pre-processing steps [130]. Researchers show that SVM are more effective that the traditional pattern recognition approach that involves a feature selection procedure followed by a conventional classifier and are also insensitive to Hughes phenomena [80]. This is particularly helpful as it avoids the unnecessary additional computation of an intermediary step like feature selection/dimensionality reduction to achieve high classification accuracy.

Similar observations were reported in the field of document classification in [83], where SVM were trained directly on the original high dimensional input space. Kernel SVM (gaussian and polynomial kernels) were employed and compared with other conventional classifiers like k-NN classifiers, Naive-Bayes Classifier, Rocchio Classifier and C4.5 Decision Tree Classifier. The results show that Kernel SVM outperform the traditional classifiers. Also, in the field of microarray gene expression analysis, SVM have been successfully applied to perform classification of several cancer diagnosis tasks [19, 139].

The insensitivity of SVM to overfitting and the ability to overcome the curse of dimensionality can be explained via the generalization error bounds developed by Vapnik et al. [166]. Vapnik showed the following generalization error bounds for Large Margin Classifiers:

$$

\epsilon = \tilde{O} \left( \frac{1}{m} \left( \frac{R^2}{\gamma^2} + \log \frac{1}{\delta} \right) \right)

$$

(3–17)

where $m$ is the number of training samples, $\gamma$ is the margin between the parallel planes, and $(R, \delta) \in \mathbb{R}^+$ with $0 < \delta \leq 1$. This error bound is inversely dependent on the sample size $m$ and the margin $\gamma$. For a finite sample size, maximizing the margin $\gamma$ (or minimizing the weight vector) would reduce the generalization error $\epsilon$. Interestingly, this error bound does not depend on the dimensionality of the input space. Since, it is highly
likely to linearly separate the data in higher dimensions, SVM tend to perform well with classification tasks in high dimensions.

### 3.2 Discriminant Functions

A discriminant function $g : \mathbb{R}^p \rightarrow \{-1, 1\}$ assigns either class 1 or class 2 to an input vector $x \in \mathbb{R}^p$. We consider here a class of discriminant functions $G$ that are well studied in literature and traditionally applied to binary classification problems.

**Quadratic and Linear Discriminant Analysis:** Consider a binary classification problem with classes $C_1$ and $C_2$ and prior probabilities given as $\pi_1$ and $\pi_2$. Assume the class conditional probability densities $f_1(x)$ and $f_2(x)$ to be normally distributed with mean vectors $\mu_1$ and $\mu_2$ and covariance matrices $\Sigma_1$ and $\Sigma_2$ respectively:

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}}\exp\left(-\frac{1}{2}(x - \mu_k)^T\Sigma_k^{-1}(x - \mu_k)\right) \quad k = 1, 2. \quad (3–18)$$

where, $|\Sigma_k|$ is the determinant of the covariance matrix $\Sigma_k$. Following Bayes optimal rule [11], Quadratic Discriminant Analysis (QDA) [111] assigns class 1 to an input vector $x$ if the following condition holds:

$$\pi_1 f_1(x) \geq \pi_2 f_2(x) \quad (3–19)$$

Linear Discriminant Analysis (LDA) [111] further assumes the covariances $\Sigma_1$ and $\Sigma_2$ are equal to $\Sigma$ and classifies an input vector again in accordance to Bayes optimal rule. The condition in (3–19) can then be rewritten as:

$$\log \frac{\pi_1}{\pi_2} + (x - \mu)^T\Sigma^{-1}(\mu_1 - \mu_2) \geq 0, \quad \mu = \frac{1}{2}(\mu_1 + \mu_2). \quad (3–20)$$

Assuming the prior probabilities to be equal, (3–20) is equivalent to:

$$(x - \mu_1)^T\Sigma^{-1}(x - \mu_1) \leq (x - \mu_2)^T\Sigma^{-1}(x - \mu_2) \quad (3–21)$$

It is interesting to note that LDA compares the squared Mahalanobis distance [35] of $x$ from the class means $\mu_1$ and $\mu_2$ and assigns the class that is closest. The squared Mahalanobis distance of a point $x$ from a distribution $\mathcal{P}$ characterized by mean vector $\mu$
and covariance matrix $\Sigma$ is defined as:

$$d_M(x, P) = (x - \mu)^T \Sigma^{-1} (x - \mu)$$  \hspace{1cm} (3–22)$$

This distance measure, unlike Euclidean distance measure, accounts for correlations among different dimensions of $x$. Equation (3–21) shows how LDA differs from other distance-based classifiers like $k$-NN classifier [11] which measures Euclidean distance to assign the class.

**Fisher Linear Discriminant Analysis:** Fisher Linear discriminant Analysis (FLDA) [11], unlike LDA, does not make assumptions on the class conditional densities. Instead, it estimates the class means from the training set. In practice, the most commonly used estimators are their maximum-likelihood estimates, given by:

$$\hat{\mu}_1 = \frac{1}{N_1} \sum_{k \in C_1} x_k, \quad \hat{\mu}_2 = \frac{1}{N_2} \sum_{k \in C_2} x_k.$$  \hspace{1cm} (3–23)$$

FLDA attempts to find a projection vector $w$ that maximizes the class separation. In particular, it maximizes the following Fisher criterion given as:

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$  \hspace{1cm} (3–24)$$

where $S_B$ is the between-class covariance matrix and is given by:

$$S_B = (\hat{\mu}_2 - \hat{\mu}_1)(\hat{\mu}_2 - \hat{\mu}_1)^T$$  \hspace{1cm} (3–25)$$

and $S_W$ is the within-class covariance matrix and is given by:

$$S_W = \sum_{k \in C_1} (x_k - \hat{\mu}_1)(x_k - \hat{\mu}_1)^T + \sum_{k \in C_2} (x_k - \hat{\mu}_2)(x_k - \hat{\mu}_2)^T$$  \hspace{1cm} (3–26)$$

The optimal Fisher discriminant $w^*$ can be obtained by maximizing the Fisher criterion:

$$\max_w J(w)$$  \hspace{1cm} (3–27)$$
An important property to notice about the objective function \( J(w) \) is that it is invariant to the rescalings of the vector \( w \rightarrow \alpha w, \forall \alpha \in \mathbb{R} \). Hence, \( w \) can be chosen in a way that the denominator is simply \( w^T S_W w = 1 \), since it is a scalar itself. For this reason, we can transform the problem of maximizing Fisher criterion \( J \) into the following constrained optimization problem,

\[
\begin{align*}
\text{maximize} & \quad w^T S_B w \\
\text{subject to} & \quad w^T S_W w = 1
\end{align*}
\]  

(3–28)

The KKT conditions for (3–28) can be solved to obtain the following generalized eigenvalue problem, given as:

\[
S_B w = \lambda S_W w
\]  

(3–29)

where \( \lambda \) represents the eigenvalue and the optimal vector \( w^* \) corresponds to the eigenvector with the largest eigenvalue \( \lambda_{\text{max}} \) and is proportional to:

\[
w^* \propto S_W^{-1}(\hat{\mu}_2 - \hat{\mu}_1)
\]  

(3–30)

The class of an input vector \( x \) is determined using the following condition:

\[
\langle w^*, x \rangle < c
\]  

(3–31)

where \( c \in \mathbb{R} \) is a threshold constant.

**Diagonal Discriminant Analysis**: Diagonal Linear Discriminant Analysis (DLDA) extends on LDA and assumes independence among the features [55]. In particular, the discriminant rule in (3–20) is replaced with:

\[
\log \frac{\pi_1}{\pi_2} + (x - \mu)^T D^{-1}(\mu_2 - \mu_1) \geq 0
\]

(3–32)

where \( D = \text{diag}(\Sigma) \). The off-diagonal elements of the covariance matrix \( \Sigma \) are replaced with zeros by independence assumption.
Similarly, Diagonal Quadratic Discriminant Analysis (DQDA) \cite{44} assumes the independence rule for QDA. The discriminant rule in this case is given by:

\[
log \frac{\pi_1}{\pi_2} + (x - \mu_2)^T D_2^{-1} (x - \mu_2) - (x - \mu_1)^T D_1^{-1} (x - \mu_1) \geq 0
\]  

(3–33)

where \(D_1 = \text{diag}(\Sigma_1)\), and \(D_2 = \text{diag}(\Sigma_2)\).

DQDA and DLDA classifiers are sometimes called Naive Bayes classifiers because they can arise in a Bayesian setting \cite{9}. Additionally, it is important to note that FLDA and Diagonal Discriminant analysis (DLDA and DQDA) are commonly generalized to handle multi-class problems as well.

**Sparse Discriminant Analysis:** The optimal discriminant vector in FLDA (3–30) involves estimating the inverse of covariance matrix obtained from sample data. However the high dimensionality in some classification problems poses the threat of singularity and thus lead to poor classification performance. One approach to overcome singularity involves a variable selection procedure that selects a subset of variables most appropriate for classification. Such a sparse solution has several advantages including better classification accuracy as well as interpretability of the model. One of the ways to induce sparsity is via the path of regularization. Regularization techniques have been traditionally used to prevent overfitting in classification models, but recently, they have been extended to induce sparsity as well in high dimensional classification problems. Here, we briefly discuss some standard regularization techniques that facilitate variable selection and prevent overfitting.

Given a set of instance-label pairs \((x_i, y_i); i = 1,2,\ldots, n;\) a regularized classifier optimizes the following unconstrained optimization problem:

\[
\min_{\beta} \Phi(x, y, \beta) + \lambda ||\beta||_p
\]  

(3–34)
where $\Phi$ represents a non-negative loss function, $(p, \lambda) \in \mathbb{R}$ and $\beta$ is the coefficient vector. Classifiers with $p = 1$ (Lasso-penalty) and $p = 2$ (ridge-penalty) have been successfully applied to several classification problems [175].

In the context of regression, Tibshirani [156] introduced variable selection via the framework of regularized classifiers using the $l_1$-norm. This method, also called Least Absolute Shrinkage and Selection Operator (LASSO), considers the least-squares error as the loss function. The user-defined parameter $\lambda$ trades regularization with the loss terms. The $l_1$-norm in LASSO produces some coefficients that are exactly 0 thus facilitating the selection of only a subset of variables useful for regression. The LASSO regression, in addition to providing a sparse model, also shares the stability of ridge regression. Several algorithms have been successfully employed to solve the Lasso regression in the past decade. Efron et al. [46] showed that, starting from zero, the LASSO solution paths grow piecewise linearly in a predictable way and hence exploit this predictability to propose a new algorithm called Least Angle Regression that solves the entire LASSO path efficiently. The LASSO framework has been further extended to several classification problems by considering different loss functions, and has been highly successful in producing sparse models with high classification accuracy.

A Lasso-type framework, however, is not without its limitations. Zou et al. [175] mention that a Lasso framework, in high dimensional problems, suffers from two drawbacks namely, the number of variables selected is limited by the number of samples $n$, and in the case of highly correlated features, the method selects one of them, neglecting the rest and also does not care about the one selected. The second limitation, also called the grouping effect, is very common in high dimensional classification problems like microarray gene analysis where a group of variables are highly correlated to each other. The authors propose a new technique that overcomes the limitations of Lasso. The technique, called elastic-net, considers a convex combination of $l_1$ and $l_2$-norms to induce sparsity. In particular, in an elastic-net
framework, the following optimization problem is minimized:

$$\min_{\beta} \Phi(x, y, \beta) + \lambda \|\beta\|_1 + (1 - \lambda) \|\beta\|_2$$  \hspace{1cm} (3–35)

where $\Phi$ is the loss function, and $0 \leq \lambda \leq 1$. When $\lambda = 0$ (or $= 1$), the elastic-net framework simplifies to Lasso (or ridge) frameworks. The method could simultaneously perform variable selection along with continuous shrinkage and also select groups of correlated variables. An efficient algorithm, called LARS-EN, along the lines of LARS, was proposed to solve the elastic-net problem. It is important to note that these regularized frameworks are very general and can be added to models that suffer from overfitting. They provide better generalization performance by inherently performing variable selection and thus also producing better interpretable models.

Sparsity can be induced to the solution of FLDA using regularization techniques described above. One such method called Sparse Linear Discriminant Analysis (SLDA), is inspired from penalized least squares where regularization is applied to the solution of least squares problem via Lasso-penalty. The penalized least squares problem is formulated as:

$$\min_{\beta} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1$$ \hspace{1cm} (3–36)

where $X$ represents the data matrix and $y$ is the outcome vector. The second term in (3–36) is assumed to induce sparsity to the optimal $\beta$.

In order to induce sparsity in FLDA via the $l_1$ penalty, the generalized eigenvalue problem in (3–29) is first reformulated as an equivalent least squares regression problem and is shown that the optimal discriminant vector of FLDA is equivalent to the optimal regression coefficient vector. This is achieved by applying the following theorem:

**Theorem 3.1.** Assume the between-class covariance matrix $S_B \in \mathbb{R}^{p \times p}$ and the within-class covariance matrix $S_W \in \mathbb{R}^{p \times p}$ be given by (3–25) and (3–26). Also, assume $S_W$ is positive definite and denote its Cholesky decomposition as $S_W = R_W^T R_W$ where $R_W \in \mathbb{R}^{p \times p}$ is an upper triangular matrix. Let $H_B \in \mathbb{R}^{n \times p}$ satisfy $S_B = H_B^T H_B$. Let
\( \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_q (q \leq \min(p, n-1)) \) denote the eigenvectors of problem (3–29) corresponding to the \( q \) largest eigenvalues \( \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_q \). Let \( \mathbf{A} \in \mathbb{R}^{p \times q} = [\alpha_1, \alpha_2, \ldots, \alpha_q] \) and \( \mathbf{B} \in \mathbb{R}^{p \times q} = [\beta_1, \beta_2, \ldots, \beta_q] \). For \( \lambda > 0 \), let \( \hat{\mathbf{A}} \) and \( \hat{\mathbf{B}} \) be the solution to the following least squares regression problem:

\[
\minimize_{\mathbf{A}, \mathbf{B}} \sum_{i=1}^{n} \| \mathbf{R}_{\mathbf{W}^{-T} \mathbf{H}_{B,i}} - \mathbf{A} \mathbf{B}^T \mathbf{H}_{B,i} \|^2 + \sum_{j=1}^{q} \beta_j^T \mathbf{S}_{\mathbf{W}} \beta_j,
\]

subject to \( \mathbf{A}^T \mathbf{A} = \mathbf{I} \)

(3–37)

where, \( \mathbf{H}_{B,i} \) is the \( i^{th} \) row of \( \mathbf{H}_B \). Then \( \hat{\beta}_j, j = 1, 2, \ldots, q \), span the same subspace as \( \mathbf{v}_j, j = 1, 2, \ldots, q \).

Please refer to [138] for the proof of Theorem 3.1.

After establishing the equivalence, the regularization is applied on the least squares formulation in (3–37) via the Lasso-penalty as shown below:

\[
\minimize_{\mathbf{A}, \mathbf{B}} \sum_{i=1}^{n} \| \mathbf{R}_{\mathbf{W}}^{-T} \mathbf{H}_{B,i} - \mathbf{A} \mathbf{B}^T \mathbf{H}_{B,i} \|^2 + \sum_{j=1}^{q} \beta_j^T \mathbf{S}_{\mathbf{W}} \beta_j + \sum_{j=1}^{q} \lambda_j \| \beta_j \|_1,
\]

subject to \( \mathbf{A}^T \mathbf{A} = \mathbf{I} \)

(3–38)

Since (3–38) is non-convex, finding the global optimum is often difficult. Qiao et al. (2009) [138], suggest a technique to obtain a local optimum by alternating optimization over \( \mathbf{A} \) and \( \mathbf{B} \).

Clemmensen et al. [29] also propose a similar sparse model using FLDA for classification problems. They also follow the approach of re-casting the optimization problem of FLDA into an equivalent least squares problem and then inducing sparsity by introducing a regularization term. However, the reformulation is achieved via an optimal scoring function that maps categorical variables to continuous variables via a sequence of scorings. Given a data matrix \( \mathbf{X} \in \mathbb{R}^{n \times p} \) and the samples belonging to one of the \( K \):
classes, the equivalent regression problem can be formulated as:

$$\begin{align*}
\text{minimize} & \quad \| \mathbf{Y}_k - \mathbf{X}_k \beta_k \|^2_2 \\
\text{subject to} & \quad \frac{1}{n} \theta_k^T \mathbf{Y}^T \mathbf{Y} \theta_k = 1 \\
& \quad \theta_k^T \mathbf{Y}^T \mathbf{Y} \theta_l = 0, \forall l < k,
\end{align*}$$

(3–39)

where $\theta_k$ is the score vector and $\beta_k$ is the coefficient vector. It can be shown that the optimal vector $\beta_k$ from (3–39) is also optimal to FLDA formulation in (3–28). Sparse discriminant vectors are then obtained by adding an $l_1$-penalty to the objective function in (3–39) as:

$$\begin{align*}
\text{minimize} & \quad \| \mathbf{Y}_k - \mathbf{X}_k \beta_k \|^2_2 + \gamma \beta_k^T \Omega \beta_k + \lambda \| \beta_k \|_1 \\
\text{subject to} & \quad \frac{1}{n} \theta_k^T \mathbf{Y}^T \mathbf{Y} \theta_k = 1 \\
& \quad \theta_k^T \mathbf{Y}^T \mathbf{Y} \theta_l = 0, \forall l < k,
\end{align*}$$

(3–40)

where $\Omega$ is a positive-definite matrix. The authors propose a simple iterative algorithm to obtain a local minima for the optimization problem in (3–40). The algorithm involves holding $\theta_k$ fixed and optimizing with respect to $\beta_k$, and holding $\beta_k$ fixed and optimizing with respect to $\theta_k$ until a pre-defined convergence criteria is met.

**Discriminant Functions for High Dimensional Data Classification:** LDA and QDA require the covariance within classes to be known apriori in order to establish a discriminant rule in classification problems. In many problems, since the covariance is not known apriori, researchers often attempt to estimate the covariance from the sample data. However, in high dimensional problems, the sample covariance matrix is ill-conditioned and hence induces singularity in the estimation of the inverse covariance matrix. FLDA also faces similar challenges since within-scatter and in-between scatter are estimated from the sample data. In fact, even if the true covariance matrix is not ill conditioned, the singularity of the sample covariance matrix will make these methods inapplicable when the dimensionality is larger than the sample size. Several authors performed a theoretical study on the performance of FLDA in high dimensional
classification settings. Bickel et al. [9] showed that under some regularity conditions, as the ratio of features $p$ and the number of samples $n$ tend to infinity, the worst case misclassification rate tends to 0.5. This proves that as the dimensionality increases, FLDA is only as good as random guessing.

Several alternatives have been proposed to overcome the problem of singularity in LDA and QDA. Thomaz et al. [155] propose a new LDA algorithm (NLDA), which replaces the less reliable smaller eigenvalues of the sample covariance matrix with the grand mean of all eigenvalues and keeps larger eigenvalues unchanged. NLDA has been used successfully in face recognition problems. Xu et al. [169] state the lack of theoretical basis for NLDA and introduced a modified version of LDA called MLDA, which is based on a well-conditioned estimator for high dimensional covariance matrices. This estimator has been shown to be more accurate than the sample covariance matrix asymptotically.

The assumption of independence in DLDA greatly reduces the number of parameters in the model and often results in an effective and interpretable classifier. Despite the fact that features will rarely be independent within a class, in the case of high dimensional classification problems, the dependencies cannot be estimated due to lack of data. DLDA is shown to perform well for high dimensional classification setting in spite of this naive assumption. Bickel et al. [9] theoretically showed that it will outperform classical discriminant analysis in high dimensional problems. However, one shortcoming of DLDA is uses all features and hence is not convenient for interpretation. Tibshirani et al. [157] introduced further regularization in DLDA using a procedure called nearest shrunken centroids (NSC) in order to improve misclassification error as well as interpretability. The regularization is introduced in a way that automatically assigns a weight zero to features that do not contribute to the class predictions. This is achieved by shrinking the classwise mean toward the overall mean, for each feature separately. DLDA integrated with nearest shrunken centroids was applied to gene expression array
analysis and is shown to be more accurate than other competing methods. The authors prove that the method is highly efficient in finding genes representative of small round blue cell tumors and leukemias. Several variations of NSC also exist in literature, for example [32, 158]. Interestingly, NSC is also shown to be highly successful in open-set classification problems [144, 145] where the number of classes is not necessarily closed.

Another framework applied to high dimensional classification problems include combining DLDA with shrinkage [132, 159]. Pang et al. [132] combined the shrinkage estimates of variances with diagonal discriminant scores to define two shrinkage-based discriminant rules called Shrinkage-based DQDA (SDQDA) and Shrinkage-based DLDA (SDLDA). Furthermore, the authors also applied regularization to further improve the performance of SDQDA and SDLDA. The discriminant rule combining shrinkage-based variances and regularization in diagonal discriminant analysis showed improvement over the original DQDA and DLDA, SVM, and $k$-Nearest Neighbors in many classification problems. Recently, Huang et al. [78] observed that the diagonal discriminant analysis suffers from serious drawback of having biased discriminant scores. Hence, they proposed bias-corrected diagonal discriminant rules by considering unbiased estimates for the discriminant scores. Especially in the case of highly unbalanced classification problems, the bias corrected rule is shown to outperform the standard rules.

Recently, SLDA has shown promise in high dimensional classification problems. In [138], SLDA was applied to synthetic and real world datasets including wine datasets and gene expression datasets is shown to perform very well on training and testing data with lesser number of significant variables. The authors in [29] compared SLDA obtained via optimal scoring to other methods like shrunken centroid regularized discriminant analysis, sparse partial least squares regression and the elastic-net regression on a number of high dimensional datasets and is shown to have comparable performance to other methods but with lesser number of significant variables.
3.3 Hybrid Classifiers

We now discuss an important set of classifiers that are frequently used for classification in the context of high dimensional data problems. High dimensional datasets usually consist of irrelevant and redundant features that adversely effect the performance of traditional classifiers. Also, the high dimensionality of the data makes the estimation of statistical measures difficult. Hence, several techniques have been proposed in the literature to perform feature selection that selects relevant features suitable for classification [75]. Generally, feature selection is performed as a dimensionality reduction step prior to building the classification model using the traditional classifiers. Unlike, other dimensionality reduction techniques like those based on transformation (e.g. principal component analysis) or compression (e.g. based on information theory), feature selection techniques do not alter the original dimensional space of the features, but merely select a subset of them [142]. Thus, they offer the advantage of interpretability by a domain expert as they preserve the original feature space. Also, feature selection helps to gain a deeper insight into the underlying processes that generated the data and thus play a vital role in the discovery of biomarkers especially in biomedical applications [48]. Thus the classification framework can be viewed as a two-stage process with dimensionality reduction via feature selection being the first step followed by a classification model. We call these set of classifiers as hybrid classifiers as different techniques pertaining to two stages have been combined to produce classification frameworks that have been successful in several high dimensional problems.

Statnikov et al. [150] recently performed a comprehensive comparative study between Random Forests [17] and Support Vector Machines for microarray-based cancer classification. They adopt several filter methods like the sequential filtering techniques as a pre-processing step to select a subset of features which are then used as input to the classifiers. It is shown that on an average, Support Vector Machines
outperform Random Forests on most microarray datasets. Recently, Pal et al. [129] studied the effect of dimensionality on performance of Support Vector Machines using four feature selection techniques namely CFS, MRMR, Random Forests and SVM-RFE [67] on hyperspectral data. Unlike earlier findings, they show that dimensionality might effect the performance of SVM and hence a pre-processing step like feature selection might still be useful to improve the performance.

3.4 Ensemble Classifiers

Ensemble classifiers have gained increasing attention from the research community over the past years, ranging from simple averaging of individually trained neural networks to the combination of thousands of decision trees to build Random Forests [17], to the boosting of weak classifiers to build a strong classifier where the training of each subsequent classifier depends on the results of all previously trained classifiers [141]. The main idea of an ensemble methodology is to combine a set of models, each of which solves the same original task, in order to obtain a better composite global model, with more accurate and reliable estimates or decisions. They combine multiple hypotheses of different models with the hope to form a better classifier. Alternatively, an ensemble classifier can also be viewed as a technique for combining many weak learners in an attempt to produce a strong learner. Hence an ensemble classifier is itself a supervised learning algorithm capable of making prediction on unknown sample data. The trained ensemble classifier, therefore, represents a single hypothesis that is not necessarily contained within the hypothesis space of the constituent models. This flexibility of ensemble classifiers can theoretically overfit to the training data more than a single model would, but however surprisingly, in practice, some ensemble techniques (especially bagging and Random Forests) combining the outputs of multiple classifiers reduces the generalization error [40].

Ensemble methods are particularly effective due to the phenomenon that various types of classifiers have different inductive biases. Additionally, ensemble methods can
effectively make use of such diversity to reduce the variance-error while keeping the bias-error in check. In certain situations, an ensemble can also reduce bias-error, as shown by the theory of large margin classifiers. So, diversified classifiers help in building lesser number of classifiers especially in the case of Random Forests. The increase in prediction accuracy comes at a cost of performing more flops in comparison to a single model. So, the ensemble methods can be thought of as trading the poor performance of a weak classifier with performing a lot of computations. So, a fast poor learner like decision trees have certainly performed better with ensemble methods; although slow algorithms can also benefit from ensemble techniques.

Recently, ensemble methods have shown promise in high dimensional data classification problems. In particular, bagging methods, random forests and boosting have been particularly impressive due to their flexibility to create stronger classifiers from weak classifiers. Here, we describe two methods: AdaBoost and Random Forests, and show their importance in high dimensional problems.

**AdaBoost:** Boosting [54, 146] is a general method which attempts to boost the accuracy of any given learning algorithm. The inception of Boosting can be traced back to a theoretical framework for studying machine learning called the PAC learning model [163]. Kearns et.al. [86] were among the first authors to pose the question of whether a weak learner which is only slightly correlated with true classification in the PAC model can be boosted into an accurate strong learning algorithm that is arbitrarily well-correlated with true classification. Schapire [146] proposed the first provable polynomial-time boosting algorithm in 1989. In 1990, Freund [53] developed a much more efficient boosting algorithm which being optimal in one sense suffered from certain practical drawbacks.

Boosting generally consists of a family of methods that produce a set of classifiers. The training set presented to each classifier is chosen based on the performance of the earlier classifier in the series. Unlike other ensemble methods like bagging [15],
in boosting, the base classifiers are trained in sequence, and each base classifier is trained using a weighted variant of the dataset in which the individual weighting coefficient of each data point depends on the performance of previous classifiers. In particular, points that are misclassified by one of the earlier classifiers are given greater weight when used to train the next classifier in the sequence. Once all the classifiers have been trained, their predictions are then combined through a weighted majority voting scheme.

AdaBoost, short for Adaptive Boosting, formulated by Yoav Freund and Robert Schapire [54], solved many of the earlier practical difficulties of earlier boosting algorithms. It can be considered as classification framework that can be used in conjunction with many other learners to improve their performance. AdaBoost is adaptive in the sense that subsequent classifiers built are tailored to favor those instances misclassified by previous classifiers. The framework provides a new weak classifier with a form of training set that is representative of the performance of previous classifiers. The weights of those training samples that are misclassified by earlier weak classifiers are assigned higher values than those that are correctly classified. This allows the new classifier to adapt to the misclassified training samples and focus on predicting them correctly. After the training phase is complete, each classifier is assigned a weight and their outputs are linearly combined to make predictions on the unknown sample. Generally, it provides a significant performance boost to weak learners that are only slightly better than random guessing. The classifiers with higher error rate would have negative coefficients in the final combination of classifiers and thereby behave like their inverses. The steps involved in AdaBoost algorithm are described below [11].

Consider a binary classification problem, in which the training data comprises input vectors \(x_1, x_2, \ldots, x_N\) along with corresponding binary target variables given by \(t\) where \(t_n \in \{-1, 1\}\). Each data point is given an associated weighting parameter \(w_n\), which is
initially set to 1/N for all data points. We assume that we have an algorithm for training a base classifier using weighted data to give a discriminative function \( y(x) \in \{-1, 1\} \).

- Initialize the data weighting coefficients \( \{w_n\} \) by setting \( w_n^{(1)} = 1/N \) for \( n = 1, 2, \ldots, N \).
- For \( m = 1, \ldots, M \):
  
  (a) Fit a classifier \( y_m(x) \) to the training data by minimizing the weighted error function

  \[
  J_m = \sum_{n=1}^{N} w_n^{(m)} l(y_m(x_n) \neq t_n)
  \]

  where \( l(y_m(x_n) \neq t_n) \) is the indicator function and equals 1 when \( (y_m(x_n) \neq t_n) \) and 0 otherwise.

  (b) Evaluate the quantities:

  \[
  \epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} l(y_m(x_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}
  \]

  and then use \( \epsilon_m \) to evaluate

  \[
  \alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}
  \]

  (c) Update the data weighting coefficients

  \[
  w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m l(y_m(x_n) \neq t_n)\}
  \]

- Make predictions using the final model, which is given by:

  \[
  Y_M(x) = \text{sign} \left( \sum_{m=1}^{M} \alpha_m y_m(x) \right)
  \]

We see that the first weak learner \( y_1(x) \) is trained using weighting coefficients \( w_n^{(1)} \) that are all equal and hence is similar to training a single classifier. From (3–44), we see that in subsequent iterations the weighting coefficients \( w_n^{(m)} \) are increased for data points that are misclassified and decreased for data points that are correctly classified. Successive classifiers are therefore forced to focus on points that have been misclassified by previous classifiers, and data points that continue to be misclassified by successive classifiers receive even greater weight. The quantities \( \epsilon_m \) represent weighted measures of the error rates of each of the base classifiers on the dataset. We therefore
see that the weighting coefficients $\alpha_m$ defined by (3–43) give greater weight to more accurate classifiers when computing the overall output for unknown samples given by (3–45). AdaBoost is sensitive to noisy data and outliers. In some problems, however, it can be less susceptible to the overfitting problem than most learning algorithms.

Boosting framework in conjunction with several classifiers have been successfully applied to high dimensional data problems. As discussed in [16] boosting framework can be viewed as a functional gradient descent technique. This analysis of boosting connects the method to more common optimization view of statistical inference. Bühlmann and Yu [21] investigate one such computationally simple variant of boosting called $L_2$Boost, which is constructed from a functional gradient descent algorithm with the $L_2$-loss function. In particular, they study the algorithm with cubic smoothing spline as the base learner and show emperically on real and simulation datasets the effectiveness of the algorithm in high dimensional predictors. Bühlmann (2003) [20] presented an interesting review on how the boosting methods can be useful for high dimensional problems. He proposes that inherent variable selection and assigning variable amount of degrees of freedom to selected variables by boosting algorithms could be a reason for high performance in high dimensional problems. Additionally, he suggests that boosting yields consistent function approximations even when the number of predictors grow fast to infinity, where the underlying true function is sparse. Dettling and Bühlmann (2003) [38] applied boosting to perform classification tasks with gene expression data. A modified boosting framework in conjunction with decision trees that does pre-selection was proposed and shown to yield slight to drastic improvement in performance on several publicly available datasets.

**Random Forests:** Random forests is an ensemble classifier that consists of many tree-like classifiers with each classifier being trained on a bootstrapped sample of the original training data, and determines a split for each node by searching only across a randomly selected subset of the input variables. For classification, each tree in the
Random Forest independently predicts the class for an input $x$. The output of the Random Forest for an unknown sample is then determined by a majority voting scheme among the trees. The algorithm for inducing Random Forests was developed by Leo Breiman [17] and can be summarized as below:

Assume the number of training samples be $N$, and the number of features be given by $M$. Also, assume that random $m$ number of features ($m < M$) used for decision at each split. Each tree in the Random Forest is constructed as follows:

- Choose a training set for this tree by bootstrapping the original training set $n$ times. The rest of the samples are used as a testing set to estimate the error of the tree.
- For each node of the tree, the best split is based on randomly choosing $m$ features for each training sample and the tree is fully grown without pruning.

For prediction, a new sample is pushed down the tree. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the class obtained from majority vote of all the trees is reported as Random Forest prediction.

Random Forests is considered one of the most accurate classifiers and are reported to have several advantages. Random Forests are shown to handle many features and also assign a weight relative to their importance in classification tasks which can further be explored for feature selection. The computational complexity of the algorithm is reduced as the number of features used for each split is bounded by $m$. Also, non-pruning of the trees also help in reducing the computational complexity further. Such random selection of features to build the trees also limits the correlation among the trees thus resulting in error rates similar to those of AdaBoost. The analysis of Random Forest show that its computational time is $cT\sqrt{M}N \log(N)$ where $c$ is a constant, $T$ is the number of trees in the ensemble, $M$ is the number of features and $N$ is the number of training samples in the dataset. It should be noted that although Random Forests are not computationally intensive, they require a fair amount of memory as they store an $N$
by T matrix in memory. Also, Random Forests have sometimes been shown to overfit to the data in some classification problems.

Random Forests, due to the aforementioned advantages, can handle high dimensional data by building large number of trees using only a subset of features. This combined with the fact that the random selection of features for a split seeks to minimize the correlation between the trees in the ensemble, certainly helps in building an ensemble classifier with high generalization accuracy for high dimensional data problems. Gislason et al. (2005) [59] performed a comparative study among Random Forests and other well-known ensemble methods for multisource remote sensing and geographic data. They show that Random Forests outperforms a single CART classifier and performs on par with other ensemble methods like bagging and boosting. On a related remote sensing application, Pal (2006) [128] investigated the use of Random Forests for classification tasks and compared its performance with SVM. Pal showed that Random Forests perform equally well to SVM in terms of classification accuracy and training time. Additionally, Pal concludes that the user defined parameters in Random Forests are less than those required for SVM. Pang et al. [131] proposed a pathway-based classification and regression method using Random Forests to analyze gene expression data. The proposed method allows to rank important pathways, discover important genes and find pathway-based outlying cases. Random Forests, in comparison with other machine learning algorithms, was shown to have either lower or second-lowest classification error rates. Recently, Genuer (2010) [58] used Random Forests to perform feature selection as well. The authors propose a strategy involving ranking of the explanatory variables using the Random Forests score of importance.
CHAPTER 4
SPARSE PROXIMAL SUPPORT VECTOR MACHINES

Proximal Support Vector Machines (PSVMs) [64, 109], unlike SVMs, find two hyperplanes, one for each class, that satisfy the condition of being closest to one class and farthest from the other. PSVMs are formulated as a Rayleigh Quotient problem and the hyperplanes are obtained by solving two generalized eigenvalue problems. The performance of PSVMs on several real world datasets shows that they perform well in comparison to SVMs [64, 109]. The simple problem formulation, computational efficiency and comparative performance of PSVMs makes them an attractive SVM-like alternative for classification tasks. Though, several researchers have extended SVMs to feature selection and shown their superior performance, extension of PSVMs to feature selection is seldom considered in literature. In this chapter, we introduce a new embedded feature selection method called Sparse Proximal Support Vector Machines (sPSVMs) that extend PSVMs to feature selection by inducing sparsity in the hyperplanes. The generalized eigenvalue problem in PSVMs is first cast as an equivalent least squares problem and sparsity is introduced via $l_1$-norm on the coefficient vector. This equivalent formulation helps in solving sPSVMs efficiently using alternate optimization techniques. The classification performance of sPSVMs on several publicly available datasets is evaluated and compared with other embedded feature selection methods. Additionally, feature selection stability of sPSVMs is also studied and contrasted with other feature selection methods. In addition to performing feature selection, we show that sPSVMs also offer the advantage of interpreting the selected features in the context of the classes.

4.1 Proximal Support Vector Machines (PSVMs)

Generalized Eigenvalue Approach: Consider a binary classification problem. Let the matrices $A_1 \in \mathbb{R}^{m \times p}$ and $A_2 \in \mathbb{R}^{n \times p}$ be given, whose rows represent the training examples of two classes $C_1$ and $C_2$ respectively. The number of samples in $C_1$ and $C_2$ are...
given by $m$ and $n$ respectively. The number of features is given by $p$. PSVMs attempts to find two hyperplanes with each hyperplane being farthest from one class and closest to the other class. Let the proximal hyperplane farthest from $C_2$ and closest to $C_1$ be given by:

$$P_1 = \{ x \in \mathbb{R}^p \mid \langle w_1, x \rangle - b_1 = 0 \}$$

(4–1)

where the weight vector $w_1 \in \mathbb{R}^p$ and the offset parameter $b_1 \in \mathbb{R}$.

PSVMs is formulated as the following Rayleigh Quotient problem:

$$\max_{z \in \mathbb{R}^{p+1}} \ r(z) = \frac{z^t H_2 z}{z^t G_1 z}$$

(4–2)

where,

$$G_1 = [A_1 - e] [A_1 - e] + \nu I, \quad H_2 = [A_2 - e] [A_2 - e], \quad z^t = [w_1^t b_1]$$

(4–3)

where $\nu$ is the Tikhonov regularization constant and $e$ is a vector of 1.

The stationary points of (4–2) are given by eigenvectors of the following generalized eigenvalue problem $GEV(H_2, G_1)$:

$$H_2 z = \lambda G_1 z$$

(4–4)

where $\lambda$ represent the eigenvalues of $GEV(H_2, G_1)$.

The hyperplane $P_1$ is given by the eigenvector corresponding to the maximum eigenvalue satisfying (4–4).

Similarly, the proximal hyperplane $P_2$ (closest to $C_2$ and farthest from $C_1$) is given by:

$$P_2 = \{ x \in \mathbb{R}^p \mid \langle w_2, x \rangle - b_2 = 0 \}$$

(4–5)

where the weight vector $w_2 \in \mathbb{R}^p$ and the offset parameter $b_2 \in \mathbb{R}$.

$P_2$ can be obtained by solving for the eigenvector corresponding to maximum eigenvalue of the generalized eigenvalue problem $GEV(H_1, G_2)$ where,

$$G_2 = [A_2 - e] [A_2 - e] + \nu I, \quad H_1 = [A_1 - e] [A_1 - e], \quad z^t = [w_2^t b_2]$$

(4–6)
A new point \( x \) is classified by computing the distance from either hyperplanes:

\[
\text{dist}(x, P_i) = \frac{|w_i^T x - b_i|}{|w_i|}, \quad i \in \{1, 2\}
\]

and the class is determined as:

\[
\text{class}(x) = \arg\min_{i \in \{1, 2\}} \{\text{dist}(x, P_i)\}
\]

**Least Squares Approach:** An alternate formulation for PSVMs via the least squares approach is proposed in this section. We first establish the equivalence between eigenvalue problems and the least-squares problems.

**Theorem 4.1.** Consider a real matrix \( X \in \mathbb{R}^{n \times p} \) with rank \( r \leq \min(n, p) \). Let matrices \( V \in \mathbb{R}^{p \times p} \) and \( D \in \mathbb{R}^{p \times p} \) satisfy the following relation:

\[
V^T (X^T X) V = D \tag{4–9}
\]

where, \( D = \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_r^2, 0, 0, \ldots, 0)_{p \times p} \). Assume \( \sigma_1^2 \geq \sigma_2^2 \geq \cdots \geq \sigma_r^2 \). For the following optimization problem,

\[
\begin{align*}
\text{minimize} & \quad \alpha^T X \beta + \mu \beta^T \beta \\
\text{subject to} & \quad \alpha^T \alpha = 1
\end{align*} \tag{4–10}
\]

\( \hat{\beta}_{opt} \) is proportional to \( v_1 \), where \( v_1 \) is the eigenvector corresponding to the largest eigenvalue \( \sigma_1^2 \).

Please refer to [176] for a proof of theorem 1.

Using theorem 1, we now establish that the proximal hyperplanes \( P_1 \) and \( P_2 \) can be obtained via the least-squares approach. Let the cholesky decomposition of the matrices \( H_2 \) and \( G_1 \) be given by:

\[
H_2 = U_2^T U_2, \quad G_1 = U_1^T U_1 \tag{4–11}
\]

where \( U_1 \) and \( U_2 \) are upper triangular matrices.
Using (4–11) in $GEV(H_2, G_1)$,

$$U_2^TU_2z = \lambda U_1^TU_1z \quad (4–12)$$

$$(U_2U_1^{-1})^T(U_2U_1^{-1})U_1z = \lambda U_1z \quad (4–13)$$

$$(U_2U_1^{-1})^T(U_2U_1^{-1})y = \lambda y \quad (4–14)$$

where $U_1z = y$.

The optimal eigenvector corresponding to proximal hyperplane $P_1$ (4–1) can be found by the following relation:

$$z_{opt} = U_1^{-1}\hat{y} \quad (4–15)$$

where $\hat{y}$ is the eigenvector corresponding to the maximum eigenvalue of the symmetric eigenvalue problem given in (4–14).

By substituting $X = U_2U_1^{-1}$, $\hat{\beta} = U_1\beta$ in (4–10), and re-arranging the terms, the following least-squares optimization problem is obtained:

$$\min_{\alpha, \beta} ||U_2U_1^{-1} - U_2\beta\alpha^T||_F^2 + \mu\beta^TG_1\beta \quad (4–16)$$

subject to $\alpha^T\alpha = 1$

By theorem 1, the optimal solution for (4–16) $\beta_{opt}$ is proportional to $z_1$, the eigenvector corresponding to the largest eigenvalue of $GEV(H_2, G_1)$.

This optimization problem is solved by alternating over $\alpha$ and $\beta$. For a fixed $\beta$, the following optimization problem is solved to obtain $\alpha$.

$$\min_{\alpha} ||U_2U_1^{-1} - U_2\beta\alpha^T||_F^2 \quad (4–17)$$

subject to $\alpha^T\alpha = 1$

Expanding the objective function,

$$\text{tr}(U_2U_1^{-1} - U_2\beta\alpha^T)(U_2U_1^{-1} - U_2\beta\alpha^T))$$

$$= \text{tr}(-2\alpha\beta^TH_2U_1^{-1} + \alpha^T\alpha\beta H_2\beta + U_1^{-1}H_2U_1^{-1})$$
Substituting $\alpha^T \alpha = 1$, the optimization problem in (4–17) is equivalent to:

$$\begin{align*}
\text{maximize} & \quad \alpha^T U_1^{-T} H_2 \beta \\
\text{subject to} & \quad \alpha^T \alpha = 1
\end{align*}$$

(4–18)

An analytical solution for this problem exists and the $\alpha_{opt}$ is given by,

$$\alpha_{opt} = \frac{U_1^{-T} H_2 \beta}{\|U_1^{-T} H_2 \beta\|}$$

(4–19)

For a given $\alpha$, the optimization problem (4–16) can be reduced to ridge regression-type problem. To see this, let $\hat{A}$ be an orthogonal set of vectors such that $A_{ort} = [\alpha; \hat{A}]$ is $p \times p$ orthogonal. Considering the first term in (4–16),

$$\begin{align*}
||U_2 U_1^{-1} - U_2 \beta \alpha^T||_F^2 \\
= \text{tr}((U_2 U_1^{-1} - U_2 \beta \alpha^T)^T (U_2 U_1^{-1} - U_2 \beta \alpha^T)) \\
= \text{tr}(A_{ort}^T (U_2 U_1^{-1} - U_2 \beta \alpha^T)^T (U_2 U_1^{-1} - U_2 \beta \alpha^T) A_{ort}) \\
= \text{tr}((U_2 U_1^{-1} A_{ort} - U_2 \beta [1, 0, 0, \ldots, 0]_{p \times p})^T (U_2 U_1^{-1} A_{ort} - U_2 \beta [1, 0, 0, \ldots, 0]_{p \times p})) \\
= ||U_2 U_1^{-1} A_{ort} - U_2 \beta [1, 0, 0, \ldots, 0]_{p \times p}||_F^2 \\
= ||U_2 U_1^{-1} \alpha - U_2 \beta||^2_F + ||U_2 U_1^{-1} \hat{A}||_F^2
\end{align*}$$

(4–21)

So, if $\alpha$ is fixed, $\beta$ optimizes the following regression problem:

$$\begin{align*}
\text{minimize} & \quad ||U_2 U_1^{-1} \alpha - U_2 \beta||^2_F + \mu \beta^T G_1 \beta \\
\end{align*}$$

(4–22)

In this case as well, an analytical solution can be found by the following expression:

$$\beta_{opt} = (H_2 + \mu G_1)^{-1} H_2 U_1^{-1} \alpha$$

(4–23)

Algorithm 1 summarizes the steps needed to solve for the optimal hyperplane $P_1$ in PSVMs using the least squares (LS) approach. Similarly, the hyperplane $P_2$ can be obtained from Algorithm 1 with input parameters $(H_1, G_2)$. 

52
**Algorithm 1** PSVMs-via-LS ($H_2, G_1$)

1. Initialize $\beta$.
2. Find the upper triangular matrix $U_1$ from the cholesky decomposition of $G_1$.
3. Find $\alpha$ from the following relation:
   \[
   \alpha = \frac{U_1^{-T}H_2\beta}{\|U_1^{-T}H_2\beta\|}
   \]
4. Find $\beta$ as follows:
   \[
   \beta = (H_2 + \mu G_1)^{-1}H_2^{-1}U_1^{-1}\alpha
   \]
5. Alternate between 3 and 4 until convergence.

### 4.2 Sparse Proximal Support Vector Machines (sPSVMs)

PSVMs are extended to perform feature selection with respect to input features by learning a sparse representation of the proximal hyperplanes. Sparsity has been well studied in the context of regression problems and is generally achieved via regularization techniques. Hence, sparsity is induced in PSVMs by solving the PSVMs-via-LS problem with an additional $l_1$-norm term. Unlike the $l_2$-norm, the $l_1$-norm term is known to drive some components of the coefficient vector to exactly zero [156].

The features corresponding to the non-zero coefficients in sparse hyperplanes obtained from sPSVMs can be interpreted as the important features discriminating the classes.

**Inducing sparsity**: Sparsity is induced in PSVMs by adding an $l_1$-norm term to the objective function given in (4–16). The resulting optimization problem is given by:

\[
\begin{aligned}
\text{minimize} & \quad \|U_2U_1^{-1} - U_2\beta \alpha^T\|_F^2 + \mu \beta^T G_1 \beta + \delta \|\beta\|_1 \\
\text{subject to} & \quad \alpha^T \alpha = 1
\end{aligned}
\]

(4–24)

where the parameter $\delta$ controls the level of sparsity in the coefficient vector $\beta$.

Again, the optimization problem is solved by alternating over $\alpha$ and $\beta$. For a fixed $\beta$, $\alpha$ can be found as before:

\[
\alpha_{\text{opt}} = \frac{U_1^{-T}H_2\beta}{\|U_1^{-T}H_2\beta\|}
\]

(4–25)
Given \( \alpha, \beta \) can be found by solving the following optimization problem:

\[
\min_{\beta} ||U_2U_1^{-1}\alpha - U_2\beta||^2 + \mu \beta^T G_1 \beta + \delta ||\beta||_1
\]  

(4–26)

Expanding (4–26),

\[
\min_{\beta} (U_2U_1^{-1}\alpha - U_2\beta)^T (U_2U_1^{-1}\alpha - U_2\beta) + \mu \beta^T G_1 \beta + \delta ||\beta||_1
\]

\[
\min_{\beta} \beta^T (H_2 + \mu G_1) \beta - 2\alpha^T U_1^{-T} H_2 \beta + \delta ||\beta||_1
\]

Assuming, \( W = [U_2 \sqrt{\mu} U_1]^T, y = [(U_2U_1^{-1}\alpha)^T 0]^T, \)

\[
\min_{\beta} \beta^T W^T W \beta - 2y^T W \beta + \delta ||\beta||_1
\]  

(4–27)

The optimization problem (4–27) is commonly referred as LASSO regression in the statistics literature [156]. Several efficient algorithms like Least Angle Regression (LARS) [46] and coordinate descent methods [56] exist in literature to solve this problem and obtain partial solution paths. So, instead of tuning \( \delta \), we solve sPSVMs for a pre-specified number of non-zero coefficients or features \( B \). This helps in comparing the performance of sPSVMs with other embedded feature selection methods for different values of \( B \). The steps of the algorithm to solve sPSVMs are summarized in Algorithm 2.

### 4.3 Numerical Experiments

**Datasets and Experimental Setup.** The performance of sPSVMs is evaluated on two categories of publicly available datasets. The first category includes three high dimensional datasets: Colon [3], Leukemia [61] and Breast [164], where the number of features are much larger than the number of samples. The second category consists of 5 publicly available datasets from the UCI machine learning repository [51]: WPBC, Ionosphere, WDBC, Spambase and Mushroom; where the number of samples outweigh the number of features. Detailed information on the dimensions of the datasets are given in Table 4-1.
Algorithm 2 sPSVMs ($H_2, G_1$)

1. Initialize $\beta$.
2. Find $U_1$ and $U_2$ that satisfy,
   \[ G_1 = U_1^T U_1, \quad H_2 = U_2^T U_2 \]
3. Find $\alpha$ from the following equation:
   \[ \alpha = \frac{U_1^{-T} H_2 \beta}{\|U_1^{-T} H_2 \beta\|} \]
4. Solve the following LASSO regression problem to obtain $\beta$:
   \[
   \min_{\beta} \|y - W\beta\|^2 + \delta \|\beta\|_1
   \]
   where $W$ and $y$ are given by:
   \[
   W = \begin{bmatrix} U_2 \\ \sqrt{\mu} U_1 \end{bmatrix},
   y = \begin{bmatrix} U_2 U_1^{-1} \alpha \\ 0 \end{bmatrix}
   \]
5. Alternate between 3 and 4 until convergence.

The performance of sPSVMs on Category I datasets is evaluated for different values of selected features $B = \{10, 20, 30\}$, and compared with C-SVMs, PSVMs, Regularized Logistic Regression (RLR) [175], sparse SVMs (s-SVMs) [153] and SVM-RFE [67]. As two independent optimization problems are solved in sPSVMs, number of selected features for each problem are chosen to be $B/2$. Due to small sample size, the parameters for different methods are chosen apriori. The value of C for C-SVMs and SVM-RFE is set to 1, while $\nu = 0.1$ in PSVMs. The value of $\alpha$ in RLR is chosen as 1 and $\mu = 100$ in sPSVMs. For the category II datasets, the performance of sPSVMs is compared against PSVMs and C-SVMs. The model parameters are obtained via 10-fold cross validation. LibSVM package (http://www.csie.ntu.edu.tw/~cjlin/libsvm/) is used for C-SVMs and SVM-RFE, FGM package (http://c2inet.sce.ntu.edu.sg/Mingkui/FGM.htm) for s-SVMs and GLMNET package (http://www-stat.stanford.edu/~tibs/glmnet-matlab/) for RLR and sPSVMs.
Table 4-1. Datasets used in numerical experiments.

<table>
<thead>
<tr>
<th>DATA SET</th>
<th># SAMPLES</th>
<th># FEATURES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATEGORY I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLON</td>
<td>62</td>
<td>2000</td>
</tr>
<tr>
<td>LEUKEMIA</td>
<td>38</td>
<td>3051</td>
</tr>
<tr>
<td>BREAST</td>
<td>77</td>
<td>4869</td>
</tr>
<tr>
<td>CATEGORY II</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WPBC</td>
<td>198</td>
<td>33</td>
</tr>
<tr>
<td>IONOSPHERE</td>
<td>351</td>
<td>34</td>
</tr>
<tr>
<td>WDBC</td>
<td>569</td>
<td>30</td>
</tr>
<tr>
<td>SPAMBASE</td>
<td>4601</td>
<td>57</td>
</tr>
<tr>
<td>MUSHROOM</td>
<td>8124</td>
<td>126</td>
</tr>
</tbody>
</table>

4.3.1 Category I Datasets

Classification Performance: The performance of sPSVMs on category I datasets is evaluated and compared with other methods. All experiments are conducted with 80/20 split on data, where 80% of the data is randomly selected for training the classification model and the remaining 20% is used to test its performance. This process is repeated for \( N = 50 \) repetitions. Since, the test sets in 50 repetitions are expected to overlap and hence no longer independent, comparing test accuracies among different classification methods for statistical significance, by performing conventional hypothesis testing may not be accurate [117]. Hence, we follow the procedure outlined in [117] and perform the corrected resampled t-test to calculate confidence intervals for the average test accuracies and also assess the statistical significance of the observed test performance differences among classification methods and sPSVMs. Table 4-2 shows the test accuracies along with their confidence intervals of all classification methods for different values of \( B \). Two approaches of PSVMs: PSVMs-GEV and PSVMs-LS are also evaluated. The results indicate that PSVMs-LS is an equivalent formulation of PSVMs-GEV. C-SVMs and PSVMs perform equally well in comparison to sPSVMs on all datasets and interestingly are not affected by the high dimensionality of the data. However, in comparison to C-SVMs and PSVMs, sPSVMs also achieve similar test accuracies, but with relatively small number of features by inherently performing feature
selection. In the case of colon dataset, all methods perform equally well for all values of \( B \). For the Leukemia dataset, all methods perform well for \( B = 10 \) case, while sPSVMs, RLR and s-SVMs perform better than SVM-RFE for \( B = \{20, 30\} \). In the case of Breast dataset, RLR performs well for all values of \( B \), while sPSVMs and s-SVMs perform better than SVM-RFE for higher values of \( B \).

Table 4-2. Test accuracies for Category I datasets from C-SVMs, PSVMs, sPSVMs, RLR, s-SVMs and SVM-RFE.

<table>
<thead>
<tr>
<th>DATASETS</th>
<th>C-SVMs (ACC(%))</th>
<th>PSVMs (GEV)</th>
<th>sPSVMs (LS)</th>
<th>RLR (B) ACC(%)</th>
<th>s-SVMs ACC(%)</th>
<th>SVM-RFE ACC(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLON</td>
<td>82.17±8.64</td>
<td>87.83±6.95</td>
<td>87.83±6.95</td>
<td>10 80.67±17.82</td>
<td>84.17±8.50</td>
<td>78.17±10.63</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20 83.80±12.50</td>
<td>83.67±10.06</td>
<td>81.17±11.30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30 84.67±11.76</td>
<td>81.33±10.15</td>
<td>78.33±12.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40 94.29±11.47</td>
<td>90.86±10.68</td>
<td>94.00±10.01</td>
</tr>
<tr>
<td>LEUKEMIA</td>
<td>98.86±3.90</td>
<td>99.14±3.42</td>
<td>99.14±3.42</td>
<td>10 98.29±4.81</td>
<td>92.57±9.65</td>
<td>96.29±7.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20 100.0</td>
<td>92.57±9.65</td>
<td>96.29±6.94</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30 64.12±11.85</td>
<td>70.25±11.01</td>
<td>62.63±11.65</td>
</tr>
<tr>
<td>BREAST</td>
<td>63.75±10.28</td>
<td>63.50±9.95</td>
<td>63.50±9.95</td>
<td>10 65.25±12.22</td>
<td>69.25±10.70</td>
<td>60.25±12.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20 66.13±11.94</td>
<td>67.87±11.53</td>
<td>61.75±10.93</td>
</tr>
</tbody>
</table>

**Feature Elimination:** Since different subsets of data are used for training sPSVMs during the 50 repetitions, there is a possibility of different sets of features being selected during each repetition. Figure 4-1 shows the feature selection frequency plot for each class in colon dataset for the case \( B = 20 \). The y-axis represents the number of times each feature is selected during the training phase. It is interesting to note that different features are repeatedly selected for each class. The list of top-ten selected features for each class in terms of frequency is shown in Table 4-3. It is worth noting that only one feature (765) appears in both the classes. Thus sPSVMs could be viewed as inducing class-specific local sparsity instead of global sparsity like other embedded methods. Such class-specific feature selection could potentially assist a domain expert to analyze the selected features in the context of different classes.

Assuming that selection frequency reflects the importance of a feature, this could be further utilized to perform feature elimination. A feature filtering parameter \( \tau \) is defined that reflects minimum no. of times a feature is selected. For e.g., setting \( \tau = 20 \) would select those features that have been picked atleast 20 times during the training phase.
Figure 4-1. Frequency plot of 2000 features in Colon dataset during 50 iterations of training phase - A) Tumor class, B) Normal class.

Table 4-3. The list of top-ten selected features in terms of frequency for each class in colon dataset for \( B = 20 \).

<table>
<thead>
<tr>
<th>Tumor Class</th>
<th>Normal Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>223, 405, 423, <strong>765</strong>, 769, 1073, 1313, 1318, 1520, 1614</td>
<td>249, 493, 513, <strong>765</strong>, 897, 1042, 1325, 1318, 1504, 1668</td>
</tr>
</tbody>
</table>

Thus, a reduced dataset can be formed from selected features, on which sPSVMs can be re-run to perform feature selection/classification. The no. of selected features and their corresponding test accuracies as a function of \( \tau \) for \( B = \{10, 20, 30\} \) are shown in Figure 4-2. As \( \tau \) increases from 0 to 25, the number of selected features decreases, while the test accuracies increase for all datasets, showing that features irrelevant to classification are removed in all the cases. The maximum number of selected features corresponding to \( \tau = 25 \) for the 3 datasets are 9, 16 and 12 respectively; which account to less than 2% of the original number of features. Thus, sPSVMs offer an effective
way to eliminate more than 98% of the features without compromising on classification accuracies.

Figure 4-2. The number of selected features (dashed line) and the corresponding test accuracies (solid line) obtained from sPSVMs as a function of $\tau$ for Category I datasets - A) Colon, B) Leukemia, C) Breast.
**Feature Selection Stability:** It is important for the proposed method to show stability in feature selection process by repeatedly selecting important features albeit different model parameters and small perturbations of the training sets. Hence, we first study the identity of selected features over different folds in the training phase. As mentioned earlier, different sets of features could be selected during the training phase as different subsets of data are presented to sPSVMs. For any two sets of features \(S_i\) and \(S_j\), we estimate the Jaccard index \(J(i, j)\) [70] as follows:

\[
J(i, j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}, \quad \forall i, j = 1, 2, \ldots, N
\]  

(4–28)

Intuitively, the Jaccard index indicates the degree of overlap in the selected features across any two folds. Generally, univariate feature selection techniques are known to promote feature selection stability [72]. For this reason, t-test, Fisher criterion and Wilcoxon Rank-sum tests [142] are considered here for comparison with sPSVMs. Table 4-4 shows the average Jaccard index for these methods along with sPSVMs on all the datasets. The performance of sPSVMs is on par with other univariate feature selection techniques and generally seems to promote greater stability for smaller values of \(B\).

<table>
<thead>
<tr>
<th>Data sets</th>
<th>B</th>
<th>sPSVMs</th>
<th>t-test</th>
<th>Fisher Criterion</th>
<th>Wilcoxon rank-sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colon</td>
<td>10</td>
<td>0.21</td>
<td>0.16</td>
<td>0.16</td>
<td>0.21</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.21</td>
<td>0.21</td>
<td>0.23</td>
<td>0.24</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.22</td>
<td>0.23</td>
<td>0.23</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.25</td>
<td>0.12</td>
<td>0.14</td>
<td>0.16</td>
</tr>
<tr>
<td>Leukemia</td>
<td>20</td>
<td>0.22</td>
<td>0.14</td>
<td>0.18</td>
<td>0.20</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.21</td>
<td>0.16</td>
<td>0.21</td>
<td>0.21</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.10</td>
<td>0.11</td>
<td>0.11</td>
<td>0.12</td>
</tr>
<tr>
<td>Breast</td>
<td>20</td>
<td>0.09</td>
<td>0.12</td>
<td>0.12</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.09</td>
<td>0.13</td>
<td>0.13</td>
<td>0.13</td>
</tr>
</tbody>
</table>

The effect of \(B\) on feature selection process is studied next. It is expected that the proposed method selects important features irrespective of the value of \(B\). The
features selected for all values of $B$ at different thresholds $\tau = \{15, 20, 25\}$ for the colon dataset is shown in Figure 4-3. For $\tau = 15$, features $\{493, 765, 1042, 1325\}$ selected for $B = 10$ consistently appear for other values of $B$ as well. Similarly, features $\{765, 1042, 1325\}$ selected for $\tau = \{20, 25\}$ also appear for other values of $B$. This behavior has been observed for other datasets (not shown here), where the features selected for lower values of $B$ are repeatedly picked for higher values of $B$ as well. Thus sPSVMs are consistent in selecting features important for classification; which could be further analyzed by a domain expert.

![Figure 4-3. Features selected from colon dataset as a function of $B$ for different values of $\tau$ - A) $\tau = 15$, B) $\tau = 20$, C) $\tau = 25$.](image)

### 4.3.2 Category II Datasets

The effectiveness of sPSVMs in eliminating irrelevant/noisy features on normal datasets is evaluated by performing numerical experiments on publicly available smaller dimensional datasets listed as Category II in Table 4-1. The number of features $B$ is varied between $0.1p$ to $0.5p$ where $p$ represents the total number of features. The
parameters $\nu$ and $\mu$ are varied in the range $\{10^{-5}, 10^{-4}, \ldots, 0.1\}$ and $\{0.1, 1, 10, 100\}$ respectively. The parameter $C$ in C-SVMs is chosen from the set $\{10^{-5}, 10^{-4}, \ldots, 100\}$. A 10-fold cross validation is performed to select the best parameters and the corresponding average classification accuracies are reported in Table 4-5. Generally, the accuracies of sPSVMs increase with the value of $B$. Overall, standard SVMs are more efficient on category II datasets, but still sPSVMs perform better than PSVMs for the datasets WPBC, Ionosphere and WDBC with less than half the number of features. Additionally, sPSVMs perform comparably well on Mushroom datasets. This indicates that sPSVMs are applicable even on smaller dimensional datasets and are effective in removing irrelevant features that would affect the classification accuracy.
Table 4-5. CV accuracies for Category II datasets from C-SVMs, PSVMs and sPSVMs.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>C-SVMs</th>
<th>C</th>
<th>Acc(%)</th>
<th>ν</th>
<th>PSVMs</th>
<th>ν</th>
<th>Acc(%)</th>
<th>sPSVMs</th>
<th>B</th>
<th>ν</th>
<th>μ</th>
<th>Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WPBC</td>
<td>1</td>
<td>83.88±6.04</td>
<td>1</td>
<td>68.7±8.21</td>
<td>0.3ρ</td>
<td>0.1</td>
<td>100</td>
<td>78.1±2.6</td>
<td>0.4ρ</td>
<td>0.1</td>
<td>100</td>
<td>78.1±2.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5ρ</td>
<td>0.1</td>
<td>100</td>
<td>77.2±3.6</td>
<td>0.1ρ</td>
<td>0.001</td>
<td>10</td>
<td>54.5±20.9</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>10</td>
<td>87.78±5.41</td>
<td>0.1</td>
<td>77.5±4.81</td>
<td>0.3ρ</td>
<td>0.1</td>
<td>10</td>
<td>78.5±3.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.4ρ</td>
<td>0.1</td>
<td>10</td>
<td>78.9±3.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5ρ</td>
<td>0.1</td>
<td>10</td>
<td>79.1±4.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WDBC</td>
<td>1</td>
<td>95.9±2.2</td>
<td>10⁻³</td>
<td>92.6±2.9</td>
<td>0.3ρ</td>
<td>0.01</td>
<td>100</td>
<td>91.7±2.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.4ρ</td>
<td>0.01</td>
<td>100</td>
<td>93.2±2.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5ρ</td>
<td>0.01</td>
<td>10</td>
<td>94.7±2.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spambase</td>
<td>1</td>
<td>72.1±1.2</td>
<td>0.1</td>
<td>68.4±1.3</td>
<td>0.3ρ</td>
<td>0.1</td>
<td>100</td>
<td>61.2±0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.4ρ</td>
<td>0.1</td>
<td>100</td>
<td>61.3±0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5ρ</td>
<td>0.1</td>
<td>100</td>
<td>61.3±0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mushroom</td>
<td>0.01</td>
<td>99.45±0.12</td>
<td>0.1</td>
<td>99.8±0.1</td>
<td>0.3ρ</td>
<td>0.01</td>
<td>100</td>
<td>99.7±0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.4ρ</td>
<td>0.01</td>
<td>100</td>
<td>99.8±0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5ρ</td>
<td>0.01</td>
<td>100</td>
<td>99.8±0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Recently, norm-based sparsity in dimensionality reduction has been widely investigated and also applied for feature selection studies. Popular subspace learning methods like Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) have been reformulated using norm constraints to induce feature sparsity. For example, Zou et al. [176] proposed a sparse PCA algorithm based on $l_1$-norm and $l_2$-norm regularization. Moghaddam et al. [113] proposed both exact and greedy algorithms for binary class sparse LDA. Cai et al. [24] proposed a unified sparse subspace learning framework based on $l_1$-norm regularization.

Subspace learning methods learn a coefficient matrix $W$ from the data,

$$W = [w_1, w_2, \ldots, w_k]$$  \hspace{1cm} (5–1)

where $w_i \in \mathbb{R}^p$ represents a coefficient vector. Here, each row in $W$ represents a feature, while the column represents dimension of the subspace. For example, in the case of Principal Component Analysis (PCA), $w$ represents the principal components. However, the features selected by sparse subspace methods are generally independent and are different for each dimension of the subspace. Though dimensionality reduction has been achieved, it is still unclear which are the most important features. In order to achieve feature selection in such methods, ideally, we would like to induce row sparsity on $W$, where the all the elements in a row are simultaneously zero. Then, the rows corresponding to non-zero coefficients can be interpreted as the most important features across all dimensions. Row sparsity has been achieved in literature using a $l_{2,1}$-norm of a matrix as described below:

### 5.1 $l_{2,1}$-norm

The $l_{2,1}$-norm of a matrix was first introduced in [42] as the rotational invariant $l_1$-norm and also used for multi-task learning [47, 125] and tensor factorization [77]. It is
defined as:
\[
\|W\|_{2,1} = \sum_{i=1}^{p} \sqrt{\left( \sum_{j=1}^{k} w_{ij}^2 \right)} = \sum_{i=1}^{p} \|w_i\|_2
\]  

(5–2)

where \(w_i\) represents the \(i\)-th row of the matrix.

The \(l_{2,1}\)-norm is rotational invariant for rows: \(\|WR\|_{2,1} = \|W\|_{2,1}\). Additionally, it can be shown that it is a valid norm as it satisfies the three norm conditions:

- positivity:
  \[
  \|A\|_{2,1} > 0, \quad \forall \|A\| \neq 0,
  \]  

(5–3)

- absolute homogeneity:
  \[
  \|\alpha A\|_{2,1} = |\alpha| \|A\|, \quad \forall \alpha \in \mathbb{R} \neq 0,
  \]  

(5–4)

- triangular inequality:
  \[
  \|A\|_{2,1} + \|B\|_{2,1} \geq \|A + B\|_{2,1},
  \]  

(5–5)

where \(A\) and \(B\) \(\in \mathbb{R}^{p \times k}\).

Similar to \(l_1\)-norm, minimization of \(l_{2,1}\)-norm encourages rows of the matrix to simultaneously go to zero. This property of \(l_{2,1}\)-norm has been utilized to induce row sparsity in the coefficient matrix.

### 5.2 Joint Feature Selection and Subspace Learning

Yan et al. [170] proposed a unified graph embedding framework that can be utilized to interpret popular subspace learning methods like PCA and LDA. In a graph embedding framework, given a data matrix \(X \in \mathbb{R}^{m \times n}\), a data graph \(G\) is constructed whose vertices correspond to \(\{x_1, x_2, \ldots, x_m\}\). Let \(A \in \mathbb{R}^{m \times m}\) be a symmetric adjacency matrix with \(A_{ij}\) representing some similarity relationship between data. The purpose of graph embedding is to find a lower dimensional subspace representation of graph \(G\) that best preserves the relationship between the data points. The optimal \(W\) can be obtained by:

\[
\text{minimize} \quad \text{tr}(W^T XLX^T W)
\]

subject to \(W^T XDX^T W = I_{k \times k}\)

(5–6)
where $D_{ii} = \sum_j A_{ij}$ is the diagonal matrix, $L = D - W$ is the graph laplacian [26].

The solution to (5–6) can be obtained by solving the following generalized eigenvalue problem given by:

$$XLX^TW = \Lambda XDX^TW$$  

(5–7)

where $\Lambda$ is a diagonal matrix with eigenvalues as the diagonal elements.

Different subspace learning methods can be obtained with different choices of the adjacency matrix $A$. For example, in the case of Linear Discriminant Analysis, for $c$ classes with $n_k$ samples in each class,

$$f(x) = \begin{cases} \frac{1}{n_k}, & \text{if } x_i \text{ and } x_j \text{ belong to the k-th class;} \\ 0, & \text{otherwise.} \end{cases}$$  

(5–8)

$l_{2,1}$-norm is used to induce row sparsity in subspace learning methods. The following general optimization problem is solved to perform joint feature selection and subspace learning:

$$\minimize_{W \in \mathbb{R}^{p \times k}} \text{tr}(W^T XLX^TW) + C\|W\|_{2,1}$$  

subject to $W^T XDX^TW = I_{k \times k}$  

(5–9)

The parameter $C$ controls row sparsity with increasing values of $C$ forcing more rows to go to zero. Though the objective function in (5–9) is convex, the optimization problem is difficult to solve as the constraint is non-convex.

5.3 Current Approach

Nie et al. [119] proposed an efficient reformulation to (5–9). The following theorem allows us to efficiently solve (5–9).

**Theorem 5.1.** Let $Y \in \mathbb{R}^{n \times m}$ be a matrix of which each column is an eigenvector of eigen-problem $Wy = \lambda Dy$. If there exists a matrix $A \in \mathbb{R}^{d \times m}$ such that $X^T A = Y$, then each column of $A$ is an eigenvector of eigen-problem $XWX^T a = \lambda XDX^T a$ with the same eigenvalue $\lambda$. 
Please refer to corollary of Theorem 1 in [24] for the proof of theorem 5.1.

Theorem 5.1 shows that instead of solving the eigen-problem $\mathbf{X} \mathbf{L} \mathbf{X}^T \mathbf{W} = \Lambda \mathbf{X} \mathbf{D} \mathbf{X}^T \mathbf{W}$, $\mathbf{W}$ can be obtained by the following two steps:

- Solve the eigenvalue problem: $\mathbf{A} \mathbf{Y} = \Lambda \mathbf{D} \mathbf{Y}$ \hspace{1cm} (5–10)
- Find $\mathbf{W}$ such that it satisfies the following linear system of equations:
  \[ \mathbf{X}^T \mathbf{W} = \mathbf{Y} \] \hspace{1cm} (5–11)

Since (5–11) is a linear system, it could have the following possibilities:

1. The system has infinitely many solutions,
2. The system has a single unique solution,
3. The system has no solution.

Gu et al. [62] propose algorithms to solve in case of three possibilities.

**Case 1:** In case when the linear system has infinitely many solutions, the following convex relaxation is proposed:

\[
\text{minimize}_{\mathbf{W} \in \mathbb{R}^{p \times k}} \| \mathbf{W} \|_{2,1}
\text{subject to } \mathbf{X}^T \mathbf{W} = \mathbf{Y} \] \hspace{1cm} (5–12)

A simple iterative algorithm is proposed in [62] to solve (5–12). The algorithm is summarized as follows:
Algorithm 3 Joint Feature Selection Case 1($X, W, D$)

1. Initialize $G_0 = I$, $k = 0$.

2. Compute $Y$ based on $WY = ADY$.

3. Compute:

$$W_{k+1} = G_k^{-1}X(X^TG_k^{-1}X)^{-1}Y$$ (5–13)

4. Compute the diagonal matrix $G_{k+1}$ as follows:

$$g_{ii} = \begin{cases} 0, & \text{if } w_i = 0 \\ \frac{1}{\|a_i\|_2}, & \text{otherwise} \end{cases}$$ (5–14)

5. Alternate between 3 and 4 until convergence.

The convergence of this algorithm is proved in [119].

**Case 2:** In the case when the linear system (5–11) has a unique solution or no solutions, the following constrained optimization problem is considered:

$$\minimize_{W \in \mathbb{R}^{p \times k}} \|W\|_{2,1}$$

subject to $\|X^TW - Y\|_F^2 \leq \delta$ (5–15)

Here, the parameter $\delta$ controls the row sparsity with smaller values of $\delta$ inducing larger sparsity. [62] again proposes a simple efficient algorithm for solving (5–15).
Algorithm 4 Joint Feature Selection Case 2($X, W, D, \delta$)

1. Initialize $G_0 = I$, $k = 0$.

2. Compute $Y$ based on $WY = \Lambda D Y$.

3. Compute:

$$W_{k+1} = G_{k}^{-1}X(X^T G_{k}^{-1}X + \delta I)^{-1}Y$$

(5–16)

4. Compute the diagonal matrix $G_{k+1}$ as follows:

$$g_{ii} = \begin{cases} 0, & \text{if } w_i = 0 \\ \frac{1}{\|a_i\|^2}, & \text{otherwise.} \end{cases}$$

(5–17)

and increment $k$ to $k + 1$.

5. Alternate between 3 and 4 until convergence.

Though the current approach is simple and efficient, it has its own limitations. For example, in the case when the linear system (5–11) has infinitely many solutions, the reformulation to (5–9) limits control on row sparsity as the parameter $C$ is not explicitly set. This is especially true in the case of high dimensional datasets as they generally conform to case 1. An explicit control on row sparsity is critical in the case of limited sample high dimensional datasets as increasing values of $C$ could lead to higher sparsity and thus improve the generalization performance. Hence, we propose to provide an exact solution to the original optimization problem (5–9) with an alternate approach described below.

5.4 Alternate Approach

The optimization problem (5–9) can be re-written with a change of variables. Let,

$$\hat{W} = D^{1/2}X^T W$$

(5–18)
Then, the optimization problem (5–9) reduces to:

\[
\begin{align*}
\text{minimize} & \quad \text{tr}(\mathbf{W}^T \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{-T/2} \mathbf{W}) + C \| \mathbf{D}^{1/2} \mathbf{X}^T \mathbf{W} \|_{2,1} \\
\text{subject to} & \quad \mathbf{W}^T \mathbf{W} = \mathbf{I}_{k \times k}
\end{align*}
\]  

(5–19)

Minimization with orthogonality constraints (\(\mathbf{W}^T \mathbf{W} = \mathbf{I}_{k \times k}\)) has wide applications in combinatorial optimization, eigenvalue problems, sparse PCA etc. These optimization problems are difficult to solve as the constraints are non-convex and could lead to many local minimizers. Many of these problems in special forms are NP-hard. Additionally, these constraints are numerically expensive to preserve during iterations. The feasible set \(\mathcal{M}_p^k := \{ \mathbf{Y} \in \mathbb{R}^{p \times k} : \mathbf{Y}^T \mathbf{Y} = \mathbf{I} \}\) is often referred to as the Stiefel manifold. Most existing constraint-preserving algorithms either use matrix re-orthogonalization or generate points along geodesics of \(\mathcal{M}_p^k\). The former requires matrix factorizations such as the singular value decomposition, and the latter must compute matrix exponentials or solve partial differential equations.

Several optimization methods for Riemannian manifolds are applicable for solving the optimization problem (5–19). Such methods generally rely on either geodesics or projections. Simple methods are based on the steepest descent gradient approach [1, 162] while more sophisticated methods like conjugate gradient method and Quasi-Newton method have also been extended to Riemannian manifolds in [45] and [137] respectively. Second-order methods like Newton’s methods [1, 45] have been applied to achieve super-linear convergence. Since, the second-order methods may require additional computation depending on the cost functions, they can run slower than the simple algorithms that are only based on the gradient.

Here, we propose to solve (5–19) using an alternate constraint-preserving update scheme suggested in [168]. Wen et al. apply the Cayley transform, a Crank-Nicolson-like [60] update scheme, to preserve the constraints and based on it, develop curvilinear
search algorithms with lower computational cost compared to those based on projections and geodesics. We briefly describe the method below:

**Cayley transformation based minimization:** Consider the following optimization problem with orthogonality constraints:

\[
\begin{align*}
\text{minimize} & \quad \mathcal{F}(Y) \\
\text{subject to} & \quad Y^T Y = I
\end{align*}
\]

where \( I \) is the identity matrix and \( \mathcal{F}(Y) : \mathbb{R}^{p \times k} \to \mathbb{R} \) is the objective function. The idea of Cayley transformation method is to generate feasible points on the Stiefel manifold \( \mathcal{M}_p^k \) using Cayley transform constraint preserving update scheme and monotone curvilinear search algorithms at each iteration. The update scheme and the curvilinear search approach is described below.

**Constraint-preserving update scheme:** Given a feasible point \( Y \) and the gradient \( G := \nabla \mathcal{F}(Y) = \left( \frac{\partial \mathcal{F}(Y)}{\partial Y_{ij}} \right) \), define a skew-symmetric matrix \( M \) as:

\[
M := G Y^T - Y G^T
\]

Given the current iterate as \( Y_k \), the new trial point is determined along the curve given by:

\[
Y(\alpha) = Y_k - \alpha M \left( \frac{Y_k + Y(\alpha)}{2} \right)
\]

which satisfies \( Y(\alpha)^T Y(\alpha) = Y^T Y \) for any skew-symmetric matrix \( M \) and \( \alpha \in \mathbb{R} \). Lemma 1 below shows that the orthogonality constraints are preserved and that \( Y(\alpha) \) defines a descent direction at \( \alpha = 0 \).

**Lemma 1.** (1) Given any skew-symmetric matrix \( M \in \mathbb{R}^{p \times p} \), then the matrix \( Q := (I + M)^{-1}((I - M) \) is well-defined and orthonormal, i.e. \( Q^T Q = I \).
Given any skew-symmetric matrix $M \in \mathbb{R}^{p \times p}$, the matrix $Y(\alpha)$ defined by (5–22) satisfies $Y(\alpha)^T Y(\alpha) = Y^T Y$. Additionally, $Y(\alpha)$ can be expressed as:

$$Y(\alpha) = \left( I + \frac{\alpha}{2} M \right)^{-1} \left( I - \frac{\alpha}{2} M \right) Y$$  \hspace{1cm} (5–23)

and its derivative with respect to $\alpha$ is given as:

$$Y'(\alpha) = - \left( I + \frac{\alpha}{2} M \right)^{-1} M \left( \frac{Y + Y(\alpha)}{2} \right)$$  \hspace{1cm} (5–24)

and in particular, $Y'(0) = -MY$.

Assume $M := G Y^T - Y G^T$. Then $Y(\alpha)$ is a descent curve at $\alpha = 0$, that is,

$$F'(\alpha)(Y(0)) := \frac{\partial F(Y(\alpha))}{\partial \alpha} \bigg|_{\alpha=0} = -\frac{1}{2} \left\| M \right\|_F^2$$  \hspace{1cm} (5–25)

Please refer to [168] for proof of Lemma 1.

The matrix inverse $(I + \frac{\alpha}{2} M)^{-1}$ dominates the computation of $Y(\alpha)$. This is particularly problematic in the case of high dimensional datasets as the size of the inverse matrix would be on the order of the number of features. However, the inverse computation becomes relatively cheap when the dimensionality of the subspace is much smaller than the number of features ($k << p$). The following Lemma shows an efficient way to calculate the inverse using Sherman-Morrison-Woodbury (SMW) formula [60].

**Lemma 2.** Suppose $M = LR^T - RL^T$, where $L, R \in \mathbb{R}^{p \times k}$. Rewrite $W = UV^T$ for $U = [L, R]$ and $V = [R, -L]$. If $I + \frac{\alpha}{2} V^T U$ is invertible, then (5–23) is equivalent to:

$$Y(\alpha) = Y - \alpha U \left( I + \frac{\alpha}{2} V^T U \right)^{-1} V^T Y$$  \hspace{1cm} (5–26)

Please refer to [168] for the proof of Lemma 2.

Generally, in the case of high dimensional datasets, when $k << p$, inverting $I + U^T U \in \mathbb{R}^{k \times k}$ is much easier than inverting $(I + \frac{\alpha}{2} M) \in \mathbb{R}^{p \times p}$. 

72
**Monotone curvilinear search:** Firstly, we assume that the descent curve $Y(\alpha)$ is generated by a skew-symmetric matrix $M$ satisfying the following condition:

$$F'_\alpha(Y(0)) \leq -\sigma \|M\|_F^2$$  \hspace{1cm} (5–27)

where $\sigma > 0$ is a constant.

Several previous studies showed that the steepest descent method with a fixed step size may not converge. However, by choosing an appropriate step size, convergence can be guaranteed with accelerated speed without significantly increasing the cost of each iteration. Ideally, at each iteration $k$, one could choose the step size by minimizing $F(Y_k(\alpha))$ along the curve $Y_k(\alpha)$ to global optimality. Since finding a such a global optimizer is computationally expensive at each iteration, one generally finds an approximate step size that satisfies the Armijo-Wolfe conditions. In our case, since the objective function is non-differentiable, we find step-size that satisfies the weak Armijo-Wolfe line conditions given by:

$$F(Y_k(\alpha_k)) \leq F(Y_k(0)) + \rho_1 \alpha_k F'_\alpha(Y_k(0))$$  \hspace{1cm} (5–28)

$$F(Y_k(\alpha_k)) \geq \rho_2 F'_\alpha(Y_k(0))$$  \hspace{1cm} (5–29)

where $0 < \rho_1 < \rho_2 < 1$ are two parameters. Several algorithms have been proposed in literature to find $\alpha_k$ that satisfy (5–28) and (5–29) [114, 120]. The algorithm to find the step size is given in Algorithm 5:

**Termination Rules:** As the iterates approach a stationary point, it is important to detect the slowdown and stop appropriately. Additionally, it is tricky to correctly predict whether an algorithm is temporarily or permanently trapped in a region when its convergence speed has reduced. Hence, it is important to have flexible termination rules. We define the following termination rules:

- Norm of function gradient $\nabla F(Y)$ goes below a threshold $\epsilon$:

$$\nabla F(Y) := G - YG^T Y \leq \epsilon$$  \hspace{1cm} (5–30)

73
Algorithm 5 Weak Wolfe Line Search ($\mathcal{F}(\mathbf{Y}_k(0)), \mathcal{F}'_{\alpha}(\mathbf{Y}_k(0)))$

Set $\alpha = 1, \mu = 0, \nu = \infty$.

while true do
  if $\mathcal{F}(\mathbf{Y}_k(\alpha_k)) > \mathcal{F}(\mathbf{Y}_k(0)) + \rho_1 \alpha_k \mathcal{F}'_{\alpha}(\mathbf{Y}_k(0))$ then
    $\nu = \alpha$
  else if $\mathcal{F}(\mathbf{Y}_k(\alpha_k)) < \rho_2 \mathcal{F}'_{\alpha}(\mathbf{Y}_k(0))$ then
    $\mu = \alpha$
  else
    STOP.
  end if
  if $\nu < \infty$ then
    $\alpha = (\mu + \nu) / 2$
  else
    $\alpha = 2\alpha$
  end if
end while

• Number of iterations $k$ are greater than a pre-defined limit $K$.

• Relative change in $\mathbf{Y}$ at iteration $k$ and $k+1$ is less than a threshold $\epsilon_Y$,

$$\text{tol}^Y_k := \frac{\|\mathbf{Y}^{(k)} - \mathbf{Y}^{(k+1)}\|_F}{\sqrt{n}} \leq \epsilon_Y$$  \hspace{1cm} (5–31)

• Relative change in objective function value $\mathcal{F}$ at iteration $k$ and $k+1$ is less than the threshold $\epsilon_F$,

$$\text{tol}^F_k := \frac{\mathcal{F}^{(k+1)} - \mathcal{F}^{(k)}}{|\mathcal{F}^{(k)}| + 1} \leq \epsilon_F$$  \hspace{1cm} (5–32)

• Mean change in iterate $\mathbf{Y}^{(k)}$ and objective function $\mathcal{F}^{(k)}$ over the last $T$ iterations is below thresholds $10\epsilon_Y$ and $10\epsilon_F$ respectively:

$$\text{mean}([\text{tol}^Y_{k-min(k,T)+1}, \ldots, \text{tol}^Y_k]) \leq 10\epsilon_Y \quad \text{and} \quad \text{mean}([\text{tol}^F_{k-min(k,T)+1}, \ldots, \text{tol}^F_k]) \leq 10\epsilon_F$$  \hspace{1cm} (5–33)

The algorithm is run until one of the termination rules are satisfied. Our algorithm combining the constraint preserving update and the curvilinear search is described in Algorithm 6.
Algorithm 6  Cayley transform with curvilinear search ($\mathcal{F}$)

1. Initialize $Y^{(0)} \in M_k^p$, $k \leftarrow 0$, $\epsilon \geq 0$, and $0 < \rho_1 < \rho_2 < 1$.

2. Calculate $M$ according to (5–21).

3. Compute the step size $\alpha_k$ according to Algorithm [ref].

4. Set $Y^{(k+1)} \leftarrow Y(\alpha_k)$.

5. If one of the termination rules satisfy, then STOP; otherwise $k \leftarrow k + 1$ and go to step 2.

5.5 Extending Principal Component Analysis to feature selection (JSPCA)

We now apply the above mentioned alternate approach to two different applications in extending Principal Component Analysis (PCA) to feature selection. PCA finds a set of linearly uncorrelated variables called principal components from a set of observations of possibly correlated variables. PCA removes redundancy by transforming the data from a higher dimensional space into an orthogonal lower dimensional space. This transformation is performed in a way that the first principal component accounts for the maximum possible variance, and each succeeding component having decreasing values of variance. Generally, the principal components are obtained as linear combinations of all features and hence it is difficult to extract the most important features for further classification tasks. Here, we show how PCA could be extended to perform feature selection using the $l_{2,1}$-norm. We call this technique as Joint Sparse Principal Component Analysis (JSPCA). We solve the optimization problem using the alternate approach described above and show numerical results on publicly available datasets.

Let $X \in \mathbb{R}^{m \times p}$ represent be given with the rows representing samples and the columns representing features. Let $U = [u_1, u_2, \ldots, u_k]_{p \times k}$ represent the orthonormal basis of a $k$-dimensional linear subspace $S$. $S$ attempts to capture maximal variance in
the data by optimizing the following optimization problem:

\[
\begin{align*}
\text{maximize} & \quad \text{tr}(U^T X^T X U) \\
\text{subject to} & \quad U^T U = I_{k \times k}
\end{align*}
\] (5–34)

The solution to the optimization problem (5–35) is given by eigenvectors corresponding to \( k \) largest eigenvalues of matrix \( X^T X \) [60].

We extend PCA to perform feature selection by introducing the \( l_{2,1} \)-norm on the matrix \( U \) in (5–34). Hence, the optimization problem is modified as:

\[
\begin{align*}
\text{maximize} & \quad \text{tr}(U^T X^T X U) - C \|U\|_{2,1} \\
\text{subject to} & \quad U^T U = I_{k \times k}
\end{align*}
\] (5–35)

where \( C > 0 \) controls the row-sparsity in \( U \) with increasing values of \( C \) promoting higher levels of sparsity. We adopt the alternate approach using the Cayley transform to solve (5–35).

**Numerical Experiments:** We perform numerical experiments on publicly available datasets to evaluate the performance of JSPCA. We combine JSPCA with the 1-NN classifier to evaluate the quality of the features selected. We compare the results with PCA-1NN classifier, where the original dataset is first transformed to a lower dimensional subspace using PCA and then classified using 1-NN classifier. We perform 10-fold cross validation and compute the average classification accuracies. We vary the dimensionality of the subspaces from \( k = \{1, 2, 3, 4, 5\} \). The dimensions of the selected datasets are given in Table 5-1.

Table 5-1. Datasets used in numerical experiments.

<table>
<thead>
<tr>
<th>Data set</th>
<th># Samples</th>
<th># Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>WPBC</td>
<td>198</td>
<td>33</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>351</td>
<td>34</td>
</tr>
<tr>
<td>Spambase</td>
<td>4601</td>
<td>57</td>
</tr>
<tr>
<td>Mushroom</td>
<td>8124</td>
<td>126</td>
</tr>
</tbody>
</table>
Tables 5-2, 5-3, 5-4 and 5-5 compare the performance of JSPCA against PCA. The results are reported for varying values of $k$ and $C$. The number of features selected decreases with increasing values of $C$ for all datasets. The classification performance of JSPCA-1NN classifier outperforms PCA-1NN classifier for lower values of $k$ and higher values of $C$ for spambase, ionosphere and mushroom datasets. In all the cases, the performance of JSPCA-1NN classifier compares well with PCA-1NN classifier for higher values of $k$. However, in all such cases, the performance is achieved with less than half the number of features.

Table 5-2. Accuracy comparison between PCA-1NN classifier and JSPCA-1NN classifier for the spambase dataset.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\text{Acc}_{\text{PCA}}$</th>
<th>$C_{\text{JSPCA}}$</th>
<th># selected features</th>
<th>$\text{Acc}_{\text{JSPCA}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>69.42±2.10</td>
<td>1000</td>
<td>45</td>
<td>70.91±2.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5000</td>
<td>19</td>
<td>71.27±1.78</td>
</tr>
<tr>
<td>2</td>
<td>72.94±1.27</td>
<td>10000</td>
<td>13</td>
<td>71.87±2.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20000</td>
<td>12</td>
<td>74.27±1.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50000</td>
<td>8</td>
<td>72.62±1.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000</td>
<td>53</td>
<td>73.85±1.71</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5000</td>
<td>33</td>
<td>73.33±1.91</td>
</tr>
<tr>
<td>3</td>
<td>78.35±2.14</td>
<td>10000</td>
<td>25</td>
<td>73.44±2.06</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20000</td>
<td>16</td>
<td>74.03±2.44</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50000</td>
<td>6</td>
<td>79.11±1.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000</td>
<td>54</td>
<td>76.90±1.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5000</td>
<td>37</td>
<td>74.81±1.08</td>
</tr>
<tr>
<td>4</td>
<td>80.63±1.75</td>
<td>10000</td>
<td>24</td>
<td>76.79±2.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20000</td>
<td>15</td>
<td>78.03±1.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50000</td>
<td>8</td>
<td>82.05±1.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000</td>
<td>54</td>
<td>79.63±1.42</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5000</td>
<td>39</td>
<td>76.74±1.15</td>
</tr>
<tr>
<td>5</td>
<td>82.24±1.50</td>
<td>10000</td>
<td>24</td>
<td>79.11±1.59</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20000</td>
<td>18</td>
<td>78.70±1.49</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50000</td>
<td>8</td>
<td>83.53±9.28</td>
</tr>
</tbody>
</table>
Table 5-3. Accuracy comparison between PCA-1NN classifier and JSPCA-1NN classifier for the WPBC dataset.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\text{Acc}_{\text{PCA}}$</th>
<th>$C_{\text{JSPCA}}$</th>
<th># selected features</th>
<th>$\text{Acc}_{\text{JSPCA}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>68.14±9.75</td>
<td>2000</td>
<td>20</td>
<td>67.69±8.15</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>16</td>
<td></td>
<td>64.11±7.28</td>
</tr>
<tr>
<td>2</td>
<td>71.15±12.03</td>
<td>2000</td>
<td>6</td>
<td>67.20±9.31</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>26</td>
<td></td>
<td>66.22±9.21</td>
</tr>
<tr>
<td>3</td>
<td>63.09±12.28</td>
<td>2000</td>
<td>5</td>
<td>63.09±6.42</td>
</tr>
<tr>
<td></td>
<td>3000</td>
<td>5</td>
<td></td>
<td>63.11±7.45</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>5</td>
<td></td>
<td>64.67±7.66</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>29</td>
<td></td>
<td>63.62±13.50</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>28</td>
<td></td>
<td>61.65±13.24</td>
</tr>
<tr>
<td>4</td>
<td>61.04±5.72</td>
<td>2000</td>
<td>13</td>
<td>65.59±8.67</td>
</tr>
<tr>
<td></td>
<td>3000</td>
<td>8</td>
<td></td>
<td>65.29±8.47</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>7</td>
<td></td>
<td>60.77±9.89</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>33</td>
<td></td>
<td>62.62±6.26</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>25</td>
<td></td>
<td>64.57±9.29</td>
</tr>
<tr>
<td>5</td>
<td>61.54±6.36</td>
<td>2000</td>
<td>16</td>
<td>68.07±7.48</td>
</tr>
<tr>
<td></td>
<td>3000</td>
<td>14</td>
<td></td>
<td>66.52±8.49</td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>6</td>
<td></td>
<td>61.41±10.33</td>
</tr>
</tbody>
</table>
Table 5-4. Accuracy comparison between PCA-1NN classifier and JSPCA-1NN classifier for the ionosphere dataset.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\text{Acc}_{PCA}$</th>
<th>$C_{JSPCA}$</th>
<th># selected features</th>
<th>$\text{Acc}_{JSPCA}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>68.67±5.80</td>
<td>25</td>
<td>70.33±5.75</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>21</td>
<td>70.68±10.61</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>73.43±6.98</td>
<td>2000</td>
<td>71.79±11.94</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3000</td>
<td>69.47±7.32</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4000</td>
<td>72.33±4.58</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>1000</td>
<td>21</td>
<td>70.68±10.61</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>2000</td>
<td>21</td>
<td>71.79±11.94</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>73.43±6.98</td>
<td>3000</td>
<td>71.20±5.99</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000</td>
<td>71.46±5.68</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>83.48±5.50</td>
<td>2000</td>
<td>77.44±7.35</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3000</td>
<td>78.58±8.75</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4000</td>
<td>79.11±11.26</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>2000</td>
<td>32</td>
<td>82.58±5.05</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>3000</td>
<td>28</td>
<td>81.71±7.49</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>82.88±5.90</td>
<td>2000</td>
<td>81.16±5.34</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3000</td>
<td>82.58±5.72</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4000</td>
<td>79.76±8.47</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>2000</td>
<td>22</td>
<td>82.58±5.05</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>3000</td>
<td>28</td>
<td>81.71±7.49</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>87.46±4.90</td>
<td>2000</td>
<td>85.47±4.26</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3000</td>
<td>82.61±5.63</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4000</td>
<td>7</td>
<td>85.76±5.51</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>2000</td>
<td>13</td>
<td>85.47±4.26</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>3000</td>
<td>9</td>
<td>87.72±7.29</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>87.46±4.90</td>
<td>4000</td>
<td>86.02±8.18</td>
<td></td>
</tr>
</tbody>
</table>
Table 5-5. Accuracy comparison between PCA-1NN classifier and JSPCA-1NN classifier for the mushroom dataset.

<table>
<thead>
<tr>
<th>k</th>
<th>$\text{Acc}_{\text{PCA}}$</th>
<th>$\text{C}_{\text{JSPCA}}$</th>
<th># selected features</th>
<th>$\text{Acc}_{\text{JSPCA}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>84.58±1.78</td>
<td>67</td>
<td>10000</td>
<td>99.34±0.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>34</td>
<td>30000</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>9.383±0.78</td>
<td>20</td>
<td>50000</td>
<td>99.90±0.97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>77</td>
<td>10000</td>
<td>99.20±0.25</td>
</tr>
<tr>
<td>2</td>
<td>98.70±0.30</td>
<td>27</td>
<td>50000</td>
<td>98.50±0.55</td>
</tr>
<tr>
<td></td>
<td></td>
<td>90</td>
<td>10000</td>
<td>99.88±0.12</td>
</tr>
<tr>
<td>3</td>
<td>99.99±0.04</td>
<td>52</td>
<td>50000</td>
<td>99.70±0.19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>94</td>
<td>10000</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>99.96±0.06</td>
<td>55</td>
<td>50000</td>
<td>99.40±0.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>74</td>
<td>30000</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>99.99±0.04</td>
<td>32</td>
<td>50000</td>
<td>99.90±0.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>35</td>
<td>90000</td>
<td>100</td>
</tr>
</tbody>
</table>
CHAPTER 6
CONSTRAINED SUBSPACE CLASSIFIERS

Feature extraction techniques transform the input data into a set of meta-features that extract relevant information from input data for classification. One popular technique called Principal Component Analysis (PCA), finds a set of linearly uncorrelated variables called principal components from a set of observations of possibly correlated variables. PCA removes redundancy by transforming the data from a higher dimensional space into an orthogonal lower dimensional space. This transformation is performed in a way that the first principal component accounts for the maximum possible variance, and each succeeding component having decreasing values of variance. The number of retained principal components is usually less than or equal to the number of original variables and are determined using several criteria like the eigenvalue-one criterion, scree test, proportion of variance accounted for, etc.

Local Subspace Classifier (LSC) [97] utilizes PCA to perform classification. During the training phase, a lower dimensional subspace is found for each class that approximates the data. In the testing phase, a new data point is classified by calculating the distance of the point to each subspace and choosing the class with minimal distance. Although LSC is simple and relatively easy to implement, it has its own limitations. LSC finds the subspaces for each class separately without the knowledge of the presence of another class. While each subspace approximates the data well, however these projections may not be ideal from a classification perspective. In this chapter, we construct another classifier called Constrained Subspace Classifier (CSC) that accounts for the presence of other another class while finding the local subspaces. LSC formulation is modified to include the relative angle between the subspaces and is solved efficiently using alternate optimization techniques. The performance of CSC on publicly available datasets is evaluated and compared with LSC.
6.1 Local Subspace Classifier (LSC)

Consider a binary classification problem. Let the matrices $X_1 \in \mathbb{R}^{p \times m}$ and $X_2 \in \mathbb{R}^{p \times n}$ be given, whose columns represent the training examples of two classes $C_1$ and $C_2$ respectively. The number of samples in $C_1$ and $C_2$ are given by $m$ and $n$ respectively. The number of features is given by $p$. Local Subspace Classifier (LSC) [97] attempts to find two subspaces separately, one for each class that best approximates the data. Let $U_1 = [u_1^{(1)}, u_2^{(1)}, \ldots, u_k^{(1)}]_{p \times k}$ and $U_2 = [u_1^{(2)}, u_2^{(2)}, \ldots, u_k^{(2)}]_{p \times k}$ represent orthonormal bases of two $k$-dimensional linear subspaces $S_1$ and $S_2$ that approximate classes $C_1$ and $C_2$ respectively. We assume the dimensionality of subspaces $S_1$ and $S_2$ to be same and equal to $k$ without loss of generality. $S_1$ and $S_2$ attempt to capture maximal variance in classes $C_1$ and $C_2$ respectively by optimizing the following optimization problems:

$$\begin{align*}
\max_{U_1 \in \mathbb{R}^{p \times k}} & \quad \text{tr}(U_1^T X_1 X_1^T U_1) \\
\text{subject to} & \quad U_1^T U_1 = I_{k \times k}
\end{align*}$$

The solution to the optimization problem (6–1) is given by eigenvectors corresponding to $k$ largest eigenvalues of matrix $X_1^T X_1$ [60].

Similarly, the following optimization problem is solved to obtain the orthonormal basis $U_2$ representing $S_2$:

$$\begin{align*}
\max_{U_2 \in \mathbb{R}^{p \times k}} & \quad \text{tr}(U_2^T X_2 X_2^T U_2) \\
\text{subject to} & \quad U_2^T U_2 = I_{k \times k}
\end{align*}$$

The orthonormal basis $U_2$ is obtained by choosing eigenvectors corresponding to $k$ largest eigenvalues of matrix $X_2^T X_2$.

A new point $x$ is classified by computing distance from subspaces $S_1$ and $S_2$:

$$\text{dist}(x, S_i) = \text{tr}(I - U_i^T xx^T U_i)$$

and the class of $x$ is determined as:

$$\text{class}(x) = \arg\min_{i \in \{1, 2\}} \{\text{dist}(x, S_i)\}$$

82
Though the subspaces $S_1$ and $S_2$ approximate the classes well, these projections may not be ideal for classification tasks as each of them are obtained without the knowledge of another class/subspace. Hence, from a classification performance perspective, these subspaces may not be the best projections for the classes. In order to account for the presence of another subspace, we consider the relative orientation of the subspaces. In the following section, we show the effect of changing the relative orientation of the subspaces on classification performance through some motivating examples.

### Table 6-1. Average classification accuracies and relative angle between subspaces generated from LSC and CSC in two examples.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>$\mathcal{N}_1$</th>
<th>$\mathcal{N}_2$</th>
<th>LSC</th>
<th>CSC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mu_1$</td>
<td>$\Sigma_1$</td>
<td>$\mu_2$</td>
<td>$\Sigma_2$</td>
</tr>
<tr>
<td>EXAMPLE 1</td>
<td>$\begin{bmatrix} 3 \ 10 \end{bmatrix}$</td>
<td>$\begin{bmatrix} 4 &amp; -1.1 \ 0 &amp; 4 \end{bmatrix}$</td>
<td>$\begin{bmatrix} 7 \ 4 \end{bmatrix}$</td>
<td>$\begin{bmatrix} 4 &amp; -2.8 \ 0 &amp; 4 \end{bmatrix}$</td>
</tr>
<tr>
<td>EXAMPLE 2</td>
<td>$\begin{bmatrix} 3 \ 5 \end{bmatrix}$</td>
<td>$\begin{bmatrix} 4 &amp; -2 \ 0 &amp; 6 \end{bmatrix}$</td>
<td>$\begin{bmatrix} 10 \ 10 \end{bmatrix}$</td>
<td>$\begin{bmatrix} 5 &amp; 2 \ 2 &amp; 5 \end{bmatrix}$</td>
</tr>
</tbody>
</table>

### 6.2 Motivating Examples

We consider two examples here showing the effect of changing the relative angle between subspaces generated by LSC. The datasets are generated from two bivariate normal distributions $\mathcal{N}_1(\mu_1, \Sigma_1)$ and $\mathcal{N}_2(\mu_2, \Sigma_2)$ representing classes $C_1$ and $C_2$. Each class consists of 100 randomly generated points from $\mathcal{N}_1$ and $\mathcal{N}_2$ respectively. The parameters of $\mathcal{N}_1$ and $\mathcal{N}_2$ for the two classes are shown in Table 6-1. LSC and CSC are trained on the data with $k = 1$ and the classification accuracies are obtained via 10-fold cross validation. The subspaces obtained for each of the training folds in example 1 and example 2 are shown in Figures 6-1 and 6-2 respectively. The average classification accuracies and the average relative angle $\theta$ ($0 \leq \theta \leq \pi/2$) between the subspaces for LSC and CSC are reported in Table 6-1. In example 1, increasing the relative angle between the subspaces clearly improves the classification accuracy by 20%. However in example 2, decreasing the relative angle between the subspaces shows better
classification performance and outperforms LSC by $\approx 11\%$. These examples show that relative orientation of the subspaces should also be considered in addition to capturing maximal variance in data.

Figure 6-1. Data points generated by $\mathcal{N}_1$ and $\mathcal{N}_2$ in example 1 and the subspaces generated by LSC and CSC in each of the training folds - A) LSC, B) CSC.

Figure 6-2. Data points generated by $\mathcal{N}_1$ and $\mathcal{N}_2$ in example 2 and the subspaces generated by LSC and CSC in each of the training folds - A) LSC, B) CSC.

### 6.3 Constrained Subspace Classifier (CSC)

Constrained Subspace Classifier (CSC) finds two subspaces simultaneously, one for each class, such that each subspace accounts for maximal variance in the data in the presence of the other class/subspace. Thus, CSC allows for a tradeoff between approximating the classes well and the relative orientation among the subspaces. The relative orientation between subspaces is generally defined in terms of principal angles [71]. We briefly review principal angles between subspaces below, which are further
utilized to modify the formulation of LSC to include the relative orientation among the subspaces.

**Definition 1:** Let $U_1 \in \mathbb{R}^{p \times k}$ and $U_2 \in \mathbb{R}^{p \times k}$ be two orthonormal matrices spanning subspaces $S_1$ and $S_2$. The principal angles $0 \leq \theta_1 \leq \theta_2 \leq \theta_3 \leq \cdots \leq \theta_k \leq \pi/2$ between subspaces $S_1$ and $S_2$, are defined recursively by:

$$\cos \theta_i = \max \max_{x_i \in S_1, y_i \in S_2} x_i^\top y_i$$

subject to $x_i^\top x_i = 1$, $y_i^\top y_i = 1$, $x_i^\top x_j = 0$, $y_i^\top y_j = 0 \quad \forall j = 1, 2, \ldots, i - 1$. \hfill (6–5)

Intuitively, the first principal angle $\theta_1$ is the smallest angle between all pairs of unit vectors in the first and second subspaces. The rest of the principal angles are similarly defined. The principal angles can be computed from the Singular Value Decomposition (SVD) of $U_1^\top U_2$ [60].

$$U_1^\top U_2 = X(\cos \Theta)Y \hfill (6–6)$$

where $X = [x_1, x_2, \ldots, x_k]$, $Y = [y_1, y_2, \ldots, y_k]$, and $\cos \Theta$ is the diagonal matrix $\cos \Theta = \text{diag}(\cos \theta_1, \cos \theta_2, \ldots, \cos \theta_k)$. The cosines of the principal angles are also sometimes known as canonical correlations.

Several distance metrics between subspaces have been defined in terms of principal angles. One such distance metric, known as Projection Metric [71], is defined as:

$$d_P(U_1, U_2) = \sqrt{\sum_{i=1}^{k} \sin^2 \theta_i} \hfill (6–7)$$

The projection metric $d_P(U_1, U_2)$ can be expressed in terms of $U_1$ and $U_2$ as shown below.

**Theorem 1:** Let $U_1 \in \mathbb{R}^{p \times k}$ and $U_2 \in \mathbb{R}^{p \times k}$ be two orthonormal matrices spanning subspaces $S_1$ and $S_2$. The projection metric $d_P(U_1, U_2)$ between $S_1$ and $S_2$ is given by:

$$d_P(U_1, U_2) = \frac{1}{\sqrt{2}} \| U_1 U_1^\top - U_2 U_2^\top \|_F \hfill (6–8)$$
The projection metric is utilized to incorporate the relative orientation between subspaces in LSC. The formulation of LSC is modified as shown below to obtain the Constrained Subspace Classifier (CSC):

\[
\begin{align*}
\text{maximize} \quad & \text{tr}(U_1^T \mathcal{X}_1 \mathcal{X}_1^T U_1) + \text{tr}(U_2^T \mathcal{X}_2 \mathcal{X}_2^T U_2) - C \|U_1 U_1^T - U_2 U_2^T\|_F^2 \\
\text{subject to} \quad & U_1^T U_1 = I_{k \times k}, \quad U_2^T U_2 = I_{k \times k}
\end{align*}
\]

where the parameter \( C \) controls the tradeoff between the relative orientation of the subspaces and the approximation of the data. It is important to note here that when \( C = 0 \), CSC reduces to LSC. Additionally, for larger positive values of \( C \), the relative orientation between subspaces reduces, while for larger negative values of \( C \), the relative orientation increases.

The objective function in (6–9) can be further modified by utilizing the orthogonality constraints. Considering the projection metric term,

\[
\|U_1 U_1^T - U_2 U_2^T\|_F^2 = \text{tr}((U_1 U_1^T - U_2 U_2^T)^T (U_1 U_1^T - U_2 U_2^T))
\]

\[
= \text{tr}(U_1 U_1^T U_1 U_1^T - 2 U_1 U_2 U_2^T U_1 + U_2 U_2^T U_2 U_2^T)
\]

Utilizing \( U_1^T U_1 = I_{k \times k}, U_2^T U_2 = I_{k \times k} \),

\[
= 2k - 2 \text{tr}(U_1^T U_2 U_2^T U_1)
\]

Introducing (6–10) in the objective function of CSC, the optimization problem in (6–9) can be re-written as:

\[
\begin{align*}
\text{maximize} \quad & \text{tr}(U_1^T \mathcal{X}_1 \mathcal{X}_1^T U_1) + \text{tr}(U_2^T \mathcal{X}_2 \mathcal{X}_2^T U_2) + C \text{tr}(U_1^T U_2 U_2^T U_1) \\
\text{subject to} \quad & U_1^T U_1 = I_{k \times k}, \quad U_2^T U_2 = I_{k \times k}
\end{align*}
\]
**Algorithm:** We introduce an alternating optimization algorithm to solve (6–11). For a fixed $U_2$, (6–11) reduces to:

$$
\text{maximize } \begin{cases} 
\mathbf{U}^{1} \in \mathbb{R}^{p \times k} \\ 
\text{tr}(\mathbf{U}^{1T}(\mathcal{X}_1\mathcal{X}_1^{T} + \mathbf{C}\mathbf{U}^{2}\mathbf{U}^{2T})\mathbf{U}^{1}) 
\end{cases} 
$$

subject to $\mathbf{U}^{1T}\mathbf{U}^{1} = \mathbf{I}_{k \times k}$ (6–12)

The solution to (6–12) is obtained by choosing eigenvectors corresponding to $k$ largest eigenvalues of symmetric matrix $\mathcal{X}_1\mathcal{X}_1^{T} + \mathbf{C}\mathbf{U}^{2}\mathbf{U}^{2T}$.

Similarly, for a fixed $U_1$, (6–11) reduces to:

$$
\text{maximize } \begin{cases} 
\mathbf{U}^{2} \in \mathbb{R}^{p \times k} \\ 
\text{tr}(\mathbf{U}^{2T}(\mathcal{X}_2\mathcal{X}_2^{T} + \mathbf{C}\mathbf{U}^{1}\mathbf{U}^{1T})\mathbf{U}^{2}) 
\end{cases} 
$$

subject to $\mathbf{U}^{2T}\mathbf{U}^{2} = \mathbf{I}_{k \times k}$ (6–13)

where the solution to (6–13) is again obtained by choosing eigenvectors corresponding to $k$ largest eigenvalues of symmetric matrix $\mathcal{X}_2\mathcal{X}_2^{T} + \mathbf{C}\mathbf{U}^{1}\mathbf{U}^{1T}$.

**Termination Rules:** As the iterates approach a stationary point, it is important to detect the slowdown and stop appropriately. Additionally, it is tricky to correctly predict whether an algorithm is temporarily or permanently trapped in a region when its convergence speed has reduced. Hence, it is beneficial to have flexible termination rules. We define the following three termination rules:

- **Maximum limit $K$** on the number of iterations,
- **Relative change in $U_1$ and $U_2$ at iteration $k$ and $k+1$,**
  $$
tol_{U_1}^{k} = \frac{\|U_1^{(k)} - U_1^{(k+1)}\|_{F}}{\sqrt{n}}, \quad tol_{U_2}^{k} = \frac{\|U_2^{(k)} - U_2^{(k+1)}\|_{F}}{\sqrt{n}} \quad (6–14)
$$
- **Relative change in objective function value at iteration $k$ and $k+1$,**
  $$
tol_{F}^{k} = \frac{F^{(k+1)} - F^{(k)}}{|F^{(k)}| + 1} \quad (6–15)
$$

The algorithm for CSC is summarized in Algorithm 7.
Algorithm 7 CSC \((X_1, X_2, k, C)\)

1. Initialize \(U_1\) and \(U_2\) such that \(U_1^T U_1 = I_{k \times k}\), \(U_2^T U_2 = I_{k \times k}\).
2. Find eigenvectors corresponding to \(k\) largest eigenvalues of the symmetric matrix \(X_1 X_1^T + C U_2 U_2^T\).
3. Find eigenvectors corresponding to \(k\) largest eigenvalues of the symmetric matrix \(X_2 X_2^T + C U_1 U_1^T\).
4. Alternate between 2 and 3 until one of the termination rules are satisfied.

6.4 Comparison between LSC and CSC

The performance of CSC is evaluated on three high dimensional publicly available datasets: DLBCL [148], Breast [164] and Colon [3] with the number of samples \(\{77, 77, 62\}\) and the number of features \(\{5469, 4869, 2000\}\) respectively. The performance of CSC is evaluated for different values of \(C\), and compared with LSC. The values of \(C\) are chosen in such a way that the relative angle between the subspaces varies uniformly. The relative angle between the subspaces is evaluated in terms of the projection metric \(d_P\). The value of \(d_P\) varies between 0 and \(k\), where \(k\) is the dimensionality of the subspaces. The value of \(k\) is chosen as \(\{1, 3, 10\}\). The classification performance is evaluated using leave-one-out cross validation (LOOCV) technique. We define a relative angle parameter \(\delta\) as follows:

\[
\delta_i = d_P^{LSC} - i \Delta_+, \quad i = 1, 2, 3, \ldots, m
\]

\[
\delta_i = d_P^{LSC} - i \Delta_-, \quad i = -1, -2, -3, \ldots, m
\]

where,

\[
\Delta_+ = \frac{k - d_P^{LSC}}{m}, \quad \Delta_- = \frac{d_P^{LSC}}{m}
\]

The classification accuracies as a function of \(\delta\) for different values of \(k\) is shown in Figure 6-3. \(\delta_0\) represents the results of LSC. \(\delta_{-1}, \delta_{-2}\) corresponds to \(C > 0\); \(\delta_1, \delta_2\) corresponds to \(C < 0\). As mentioned earlier, positive values of \(C\) decrease the relative angle between the subspaces while negative values of \(C\) increase the relative angle.

The values of \(K, tol^k, tol^h_1\) and \(tol^h_2\) are chosen to be 2000, \(10^{-6}\), \(10^{-6}\) and \(10^{-6}\) respectively. For DLBCL and Colon datasets, classification accuracy is improved
by reducing the relative angle between subspaces. In the case of Breast dataset, increasing the relative angle considerably improves the classification accuracy. However, the effect of angle change is relatively small for $k = 10$ dimensional subspaces for all datasets. Additionally, in the case of Colon dataset, the change in the dimension of the subspaces does not effect the performance of CSC.

Figure 6-3. LOOCV accuracies as a function of $\delta$ for different values of $k$ for the three high dimensional datasets A) DLBCL, B) Breast and C) Colon.
CHAPTER 7
BREAST CELL CHARACTERIZATION USING RAMAN SPECTROSCOPY

Recent advances in Raman spectroscopy have spawned a surge of interest in biomedical applications of the technology, particularly in the field of oncology \([49]\). Advances in sensors, diode lasers, and other optical components, along with the reduction in the cost of this hardware, have made it possible for Raman spectrometers to become increasingly commonplace in laboratories. Raman spectroscopy has demonstrated the potential to significantly aid in the research, diagnosis and treatment of various cancers \([68, 115, 121]\). Biomedical applications of Raman spectroscopy currently under investigation range from the research laboratory bench-top to the clinical setting at the patient's bedside. Raman spectroscopic analysis of biological specimens is advantageous as it provides a spectral fingerprint rich in molecular compositional information without disrupting the biological environment; allowing in-situ biochemical observations to be made. Its ability to detect variations linked to DNA/RNA, proteins, lipid, carbohydrates, and other small molecule metabolites make it an excellent tool for monitoring biochemical changes on the cellular level. Raman spectroscopy of biological samples provides a time efficient, noninvasive method for investigating potential cancer therapeutics, yielding a significant amount of biological information from a single spectrum.

The ability to differentiate between cell and tissue types is of great importance for the research, diagnosis and treatment of cancer. In-vitro gold standard assays have relied on highly specific and expensive techniques such as immunoassays using flow cytometry, fluorescence microscopy, gene expression and protein analysis. While in-vivo and ex-vivo diagnosis has been primarily based on histological staining and morphological examination by highly trained expert pathologist. All of these conventional methods are invasive, requiring the use of exogenous agents, creating an environment non-native to cells and tissues; very few of which enable the collection of real-time
in-situ dynamic information. Raman spectroscopy is capable of overcoming these disadvantages, mainly due to its non-invasive nature and its ability to provide a global overview of the biochemical composition of dynamic biological scenarios unlike most single end-point assays. This versatile technique has been demonstrated as an effective tool, in particular for oncology-based applications as well as for other applications including investigation of stem cell differentiation, cell proliferation on tissue engineering constructs, in-vitro pharmacological and toxicological effects, among many other applications; and also numerous in-vivo animal studies and human clinical trials [103, 121–123, 126, 127, 135, 136, 151, 152, 167].

For many of the oncology applications, classification and comprehensive characterization of cell types is of great importance for the selection of therapies for use in-vivo as well as the use of the most appropriate and clinically relevant cell-based models for pre-clinical development of therapies in-vitro [98]. Advances in both the hardware (e.g. optical technologies) and the software components (e.g. data analysis methods) will allow for Raman spectroscopy to participate more readily in the research and clinical settings. In this chapter the focus is on developing a robust data analysis framework for evaluating and characterizing five commonly used breast cell lines for therapy development and breast cancer research. The aim is to develop this framework such that it can both classify cell types based on cell-line specific spectral features which may ultimately in the future allow for the potential discovery of Raman-based spectral biomarkers for identifying cancer and tumor sub-types [48].

The complexity of the spectra obtained from a biological sample makes extracting relevant information and interpreting the data challenging. The most primitive data analysis procedures used for Raman studies such as peak-by-peak analysis and peak deconvolution do not allow for extensive data extraction and are a major limitation, often involving tedious, error-prone manual analysis procedures. Such methods do not allow for complete data extraction, often only using a very limited subset of
data. Therefore, a wide variety of data analysis methods have recently begun to be increasingly employed for evaluating Raman spectra; including multivariate methods such as principal component analysis, and various machine learning-based data mining and optimization methods. Data mining and machine learning techniques are able to circumvent these pitfalls by optimizing data extraction, exposing obscured correlations and reducing classification error. Computational experiments indicate that supervised classification algorithms such as SVM and LDA are able to separate the high dimensional spectral data with high accuracy [123, 126, 127, 135, 136, 152]. These algorithms can be used in combination with dimensionality reduction or feature selection techniques in order to identify the critical regions and bands of the spectrum that allow for discrimination and classification of cell types, cellular processes and cell response to various stimuli based variations in biochemical composition. Therefore, the development and implementation of advanced data mining techniques is vital for complete, rapid and accurate data analysis. Currently, there is no single standard, optimized data analysis procedure for complete processing (pre- and post-processing) of Raman spectral data. Development of a consensus method capable of being generalizable and tunable for the many different Raman spectral datasets is particularly important in regards to fostering the clinical translation of the technology, as well as for standardizing data results for Raman-based cancer research within the flourishing field.

Raman spectral data sets are defined by high-dimensionality, frequently with limited sample sizes. Standard classification models are known to perform poorly on such high dimensional datasets due to the curse of dimensionality. Thus, implementing dimensionality reduction and/or feature selection becomes crucial prior to classification. Frequently, the spectra must be classified so as to differentiate between two or more classes of samples (e.g. classifying cell death mechanisms or cancer sub-types). In other instances, the detection, monitoring and quantification of peaks indicative of bimolecular species may be required to study particular dynamic processes or levels
of biological activity (e.g. investigating changes in DNA/RNA levels during treatment with anti-cancer agents). Classification tasks have involved both supervised and unsupervised classification techniques, while the latter has involved peak-picking combined with conventional statistical test (e.g. student’s t-test, etc.), dimensionality reduction (e.g. PCA, LDA, etc.), and feature selection techniques (e.g. Correlation-based feature selection (CFS), Markov blanket filter, etc.).

Herein, we have combined Raman spectroscopy with our novel data mining framework in order to classify and characterize, \textit{in-situ}, five breast cell lines based on differences in biochemical composition determined from the significance of the spectral features utilized for classification directly from the original feature space. Our framework, known as Fisher-based Feature Selection Support Vector Machines (FFS-SVM), allows for simultaneous feature selection and subsequent classification based on the selected features. The FFS-SVM framework is compared to several of the most commonly used methods for classification and dimensionality reduction after all spectral datasets are pre-processed by a standard pre-processing procedure. An unsupervised method based on hierarchical clustering is used to construct four prototypical cytopathological binary classification tasks, typical of tasks that might be encountered in a research or clinical setting, and which are used for testing the robustness and generalizability of the FFS-SVM framework. Finally, the top-ten most significantly discriminative features, derived directly as wavenumbers from the original feature space, are correlated to their bimolecular assignments based on the literature. These wavenumbers/features are then discussed in terms of biological relevance to the differences between the cell line groupings for each classification task. Although we focus on breast cancer cell lines in this particular study, we foresee that this framework will be applicable to virtually all cancer types and also for both research and clinical applications of Raman spectroscopy.
7.1 Experimental Methods

**Cell culture:** A panel consisting of five cell lines was assembled based on three cancerous and two non-cancerous cell lines after consideration of differences in tumor origin, morphology, tumorigenecity and select gene expression profiles. All cell lines were obtained from American Type Culture Collection (ATCC), Manassas, VA. The cells lines obtained for study were MCF7, MDA-MB-231, BT474, MCF12A, and MCF10A. Table 7-1 shows the cell lines chosen with selected characteristics. All cell lines were cultured at 37°C and 5% CO₂ in humidified cell culture incubator. Media was prepared for each of the cell lines by filtration through 0.02µm vacuum filtration system. Briefly, MDA-MB-231 media consisted of Dulbeccos Modified Eagles Medium (DMEM) with 2mM L-glutamine, 1% Non-Essential Amino Acids (NEAA), and 10% Fetal Bovine Serum (FBS); MCF7 media consisted of Minimum Eagles Medium (MEM) with Earles Balanced Salt Solution (EBSS), 2mM L-glutamine, 1mM sodium pyruvate and 10µg/ml bovine insulin, and 10% FBS; BT474 consisted of ATCC 46-X Hybricare powder prepared in sterile molecular-grade water with 1.5g/L sodium bicarbonate and 10% FBS; MCF12A and MCF10A were both grown in the same complete media which consisted of DMEM/Hams F12 50/50 with, 2.5mM L-glutamine, 15mM HEPES, 0.5mM sodium pyruvate, 1.2g/L sodium bicarbonate, 20ng/ml human Epidermal Growth Factor (hEGF), 100ng/ml cholera toxin, 10µg/ml bovine insulin, 500ng/ml hydrocortisone, and 5% horse serum.

Cells, once thawed, were grown up to log-phase (usually after 2-3 passages post-thaw) prior to plating for experimental procedures as described in the following section. Cells were passaged upon becoming confluent using 0.25% trypsin/EDTA solution for harvesting, followed by centrifugation and subsequently inoculated at the pre-determined cell-line concentrations in 75 cm² flasks. It should be noted that each cell line takes a different time period to reach confluence and also have different optimal inoculation concentrations (cells/cm²) based on growth curve characteristics. Cells were
harvested for plating for experimental procedures using 0.25% trypsin/EDTA solution to remove cells from the culture flasks and then collected by centrifugation. The cells were then diluted to the desired concentrations for inoculation onto MgF$_2$ chips and cultured in 6-well plates.

Table 7-1. Characteristics of the five ATCC breast cell lines studied. IDC: Invasive Ductal Carcinoma, AC: Adenocarcinoma, ER: Estrogen Receptor, PgR: Progesterone Receptor, HER2: HER2 Overexpression, TP53: P53 protein levels and mutational status. *MCF10A cell morphology is highly dependent on Conc. of Ca$^{2+}$ [87, 118]

<table>
<thead>
<tr>
<th>Cell Line</th>
<th>Morphology</th>
<th>Tumor Type</th>
<th>Tumorigenicity</th>
<th>ER</th>
<th>PgR</th>
<th>HER2</th>
<th>TP53</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCF7</td>
<td>Mass</td>
<td>IDC</td>
<td>Less aggressive</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+/-WT</td>
</tr>
<tr>
<td>MDA-MB-231</td>
<td>Stellate</td>
<td>AC</td>
<td>Highly aggressive</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+M</td>
</tr>
<tr>
<td>BT474</td>
<td>Mass</td>
<td>IDC</td>
<td>Highly aggressive</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MCF12A</td>
<td>Round</td>
<td>Normal</td>
<td>No</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>MCF10A</td>
<td>Cobblestone*</td>
<td>Normal</td>
<td>No</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+/-WT</td>
</tr>
</tbody>
</table>

Collection of Raman spectra from cells: For collection of Raman spectra the cells were seeded on MgF$_2$ chips (5×5×1mm) at concentrations determined to allow the cells to grow up to approximately 80% confluence after at least 24 hours incubation at 37°C and 5% CO$_2$ prior to experimental collection of spectra. The MgF$_2$ substrates are used as they exhibit a very low background Raman signal and also allow for each of the cell lines to grow with the normal morphology of that observed when grown on culture-treated plastic. The cells were grown up to slightly less than confluence, approximately 80%, but to still allow for cell-cell interactions to occur. After the cells reached approximately 80% confluence, cell-containing MgF$_2$ chips were placed in sterile Delta-T Culture Dishes and fresh complete growth media was added to the cells. The spectra were collected by a Renishaw 2000 InVia Spectrometer System coupled to a Leica Microscope with a 63x water-immersion objective (N.A. 0.90). The spectra from single cells were collected one at a time with excitation from a 785nm wavelength diode laser, using a 20s integration time, and an estimated laser intensity of 50mW at the surface of the cells. The fine focus was used to adjust the focal plane and the
laser spot size to a semi-elliptical shape of approximately $20\,\mu\text{m} \times 40\,\mu\text{m}$, which when centered over each cell allowed for the laser spot to encompass the majority of the volume of the cell for spectral acquisition. 25-40 spectra were collected from each cell line and apparent outliers were removed by visual inspection prior to pre-processing of the spectra. Figure 7-1 shows the overlay of 37 Raman spectra collected from MCF10A cell line with the standard deviation plotted as well (blue line).

![Spectral overlay](image)

Figure 7-1. Spectral overlay: Overlay of 37 spectra collected from MCF10A cell line. The standard deviation over all the spectra is also shown (blue line).

**Spectral preprocessing:** All spectra are pre-processed using background subtraction, baseline correction, normalization and smoothing procedures [100, 121, 126] prior to applying dimensionality reduction and/or feature selection followed by subsequent classification. The multiple spectra collected from a biological sample can sometimes vary in wavenumber resolution. Hence, the x-axis is standardized by applying interpolation techniques to estimate values at specific and regular wavenumbers. The raw spectrum is from $601\,\text{cm}^{-1}$ to $1800\,\text{cm}^{-1}$ with intensity values at unequal intervals. In order to compare different spectra as well as perform data analysis, the spectra are standardized to have intensity values at regular intervals. An interval of
one wavenumber is chosen between 602 cm\(^{-1}\) and 1801 cm\(^{-1}\) giving a total of 1200 data points for each spectra. A simple linear interpolation method is used to find the intensity values at the predefined wavenumbers.

Noise removal is an important procedure in the data preprocessing of raw Raman spectra. First a Savitsky-Golay smoothing is performed on the standardized spectra with a 5\(^{th}\) order polynomial [143]. Then, as in accordance with the procedure outlined in [100] the fluorescence background is removed from the spectra by an automated baseline correction method. The smoothed spectra is then normalized using a standard normal variate transformation [6]. It has been shown in [2] that performing baseline correction methods prior to normalization helps in extracting robust and quantitative information from Raman spectra. The different steps are shown for the spectral preprocessing procedure in Figure 7-2.

![Figure 7-2. Spectral preprocessing: A) The raw spectrum obtained from the Raman Spectrometer. B) The smoothed & standardized spectrum along with the raw spectrum. C) Fitted Fluorescence background shown with the spectrum obtained from (B). D) The final standardized, smoothed, baseline corrected and normalized spectrum.](image-url)
The subsequent sections describe in detail the development of the FFS-SVM framework which includes a novel feature selection method called Fisher-based Feature Selection (FFS), in which the selected features, or peaks, from the average spectra of cells are ranked based on Fisher criterion. Furthermore, a method for accounting for peak position inter-class variance by peak coalescing is also incorporated. The selected features are then used for classification by C-SVM with optimization of the hyperplane, and subsequent wavenumber assignment and correlation of each feature to its biological constituent. Our frameworks performance is then compared with several popular data reduction and feature selection methods including PCA, random feature selection; as well as common classification models, such as SVM, LDA, PCA-LDA, and PCA-SVM. Classification accuracies were determined using random sub-sampling and finally classification sensitivities and specificities are also calculated.

7.2 Fisher-based Feature Selection (FFS)

The number of collected spectra is usually an order of magnitude smaller than the number of measurements for each spectrum. Several univariate [7, 44, 154] and multivariate [69, 93, 172] filter techniques have been introduced to perform feature selection in high dimensional classification problems. Univariate filter techniques assume feature independency and rank each feature according to a discriminative score, while the multivariate techniques account for feature dependencies and find an optimal subset of features based on a predefined criteria. Generally, finding an optimal subset of features is computationally intractable when the input space is reasonably large, since there are $2^n$ possibilities, where $n$ is the dimension of the input space. Though the multivariate techniques may sometimes lead to better classification performance than univariate methods, they are computationally expensive and employ heuristic methods to search for the optimal subset of features.

A comparison of the univariate and multivariate filter techniques for classification of high dimensional datasets have been studied in previous studies [75, 99]. Surprisingly,
it has been shown that the univariate selection techniques yield consistently better results than multivariate techniques and the differences are attributed to the difficulty in extracting the feature dependencies from limited sample sizes. As Raman spectral datasets are also characterized by high dimensionality and limited sample sizes, a simple univariate technique based on Fisher criterion [43] is employed which ranks the features according to their class discriminative ability.

**Peak finding:** In addition to building a classification model for the identification of cell lines, it is also important to understand and analyze the biological relevance of the selected features which contribute most significantly to the classification. Generally, in a Raman spectrum, most wavenumbers correspond to peaks that are assigned to molecular vibrational modes and represent biologically relevant molecular species. Hence, prior to ranking the features, the features are selected by considering the peaks of the average spectrum in each cell line. The set of peaks $S$ for a specific cell line are defined as local maxima given by:

$$S = \{x^* | f(x^*) \geq f(x), \forall x \in \mathcal{N}_\epsilon(x^*)\}$$  \hspace{1cm} (7–1)

where $x^*$ represents the peak location, $f(x^*)$ is the corresponding intensity value of the average spectrum and $\mathcal{N}_\epsilon(x^*)$ represents an $\epsilon$-neighborhood around $x^*$. The $\epsilon$-neighborhood is chosen by considering an average full-width at half-maximum (FWHM) of 10 cm$^{-1}$ for Raman peaks. Figure 7-3 shows the average spectra for MCF10A cells and the corresponding peaks selected for further analysis.

**Peak coalescing:** Peaks corresponding to the same biomolecular bonds may still be reported with slightly varying wavenumbers when comparing different biological samples. In order to account for this variance and therefore to create a common wavenumber reference vector for feature selection and classification tasks, peak coalescing using hierarchical clustering is employed [84]. The peak locations from different cell lines are clustered depending on their inter-distance values and choosing
Figure 7-3. Peak finding: The average spectrum for MCF10A cells and the selected peaks (shown marked with asterisks).

the number of clusters such that the total variance of all clusters is below a chosen threshold. In particular, the number of clusters $N_C$ is defined as:

$$N_C = \min \{ c \mid \sum_{j=1}^{c} s_j \leq \lambda_{th} \}$$  \(7-2\)

where,

$$s_i = \sum_{j \in C_i} (x_{ij} - \mu_i)^2$$  \(7-3\)

$C_i$ represents the index of the cluster $i$, $\mu_i$ and $s_i$ are the mean and variance of cluster $i$, $x_{ij}$ is the peak $j$ assigned to cluster $i$ and $\lambda_{th}$ is a pre-defined variance threshold. Figure 7-4 shows individual peaks chosen for MCF10A and MCF12A cell lines along with the clusters found from hierarchical clustering.

**Feature ranking**: After obtaining a common wavenumber reference vector from peak-finding and peak-coalescing, the features are ranked based on Fisher criterion [43]. For a given feature $i$, the fisher score is defined as:

$$J_i = \frac{\left( \mu^1_i - \mu^2_i \right)^2}{(\sigma^1_i)^2 + (\sigma^2_i)^2} \quad \forall i \in S,$$  \(7-4\)
where, $\mu_j^i$, $(s_j^i)^2$ and $n_j$ are the sample mean, variance and the number of data samples in class $j$ and $S$ is the set of selected peaks. Intuitively, a Fisher score would be high for a feature having high mean inter-class separation while the total within-class variance is small. Hence, it is expected that the features with high Fisher scores be part of the maximally discriminating features. Additionally, it is important to note here that the Fisher score is similar to the two-sided t-statistic estimated using pooled variance. We call this univariate filter selection technique Fisher-based Feature Selection (FFS).

### 7.3 Classification

The datasets formed from features obtained from FFS are used to train a classification algorithm. Several techniques have been proposed for binary classification problems in the literature out of which, SVM have attracted the attention of many researchers with numerous successful application in bioinformatics, finance and remote sensing among many others [23, 130, 160]. SVMs can be extended to multiclass classification problems by two commonly used techniques: one-against-one (OAO) and one-against-all (OAA).
In OAA, the \(N\)-class dataset is divided into \(N\) two-class cases, with each class being classified against the rest of the classes. The OAO approach, on the other hand, builds \(N(N-1)/2\) classification models, with every class being classified against every other class. A simple voting scheme is employed for class assignment of unknown samples. While both of these techniques are shown to perform well for several classification tasks, the performance of OAA is shown to be compromised in the case of unbalanced training datasets and the OAO approach is more computationally intensive [63]. In order to alleviate these issues, the C-SVMs are extended to multiclass classification by incorporating hierarchical clustering. The pairwise distance matrix between the average spectra of cell lines are first computed and an agglomerative hierarchical cluster tree is generated from the distance matrix. A dendrogram plot of the binary cluster tree is shown in Figure 7-5. Remarkably, but as should be expected, the largest difference occurs between the group of cancer cells and the group of non-cancer cells. Based on Figure 7-5, a classification framework is constructed from four binary classification tasks: Cancer versus Non-Cancer, MCF7 versus Other Cancers, MDA-MB231 versus BT474 and MCF10A versus MCF12A. This classification framework, in comparison with OAA and OAO approaches, reduces the computational complexity by building four cytologically relevant binary classification models for determining class membership.

### 7.4 Results and Discussion

The dataset included a total of 133 Raman spectra, from which 82 spectra were from the cancer cell lines (BT474, MCF7 and MDA-MB231) and 51 spectra from the non-cancer cell lines (MCF10A and MCF12A). The mean spectra of the five cell lines are shown in Figure 7-6. Given any two cell lines, the classification model is built in the following manner: the raw spectra from both the classes are preprocessed, the peaks are selected from the processed spectra and coalesced to remove experimental artifacts, the common reference features are ranked according to their Fisher score, the C-SVM algorithm is then trained on the dataset created with top \(k\) features chosen
with FFS, and finally the model is cross-validated using repeated random sub-sampling method [91]. The number of repetitions for cross-validation has been fixed at 100. In each repetition, 90% of the samples are used for training and the remaining 10% are used to test the classifier. The classification accuracies are reported as the average over all the classification accuracies obtained over 100 repetitions. The C parameter for SVMs has been obtained using a grid search by varying the value of C between $10^{-5}$, $10^{-4}$, ... $10^{2}$ and choosing the value that gives the maximal classification accuracy.

**Classification performance:** The following presents the results obtained from our FFS-SVM classification framework when applied on our training datasets. The classification accuracies along with specificity, sensitivity and the number of extracted features for the four binary classification tasks are shown in Table 7-2. The results show that the FFS-SVM framework, combining feature selection with C-SVM, yields highly accurate classification of the cell types.
Figure 7-6. Mean spectra: The average spectra of all five cell lines.

Table 7-2. Sensitivity, specificity and average classification accuracy for the four binary classification tasks obtained from C-SVM and validated using random sub-sampling (100 repetitions).

<table>
<thead>
<tr>
<th>Classification Task</th>
<th># of features</th>
<th>Accuracy (%)</th>
<th>Sensitivity (%)</th>
<th>Specificity (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancer Vs Non-Cancer</td>
<td>38</td>
<td>99.5</td>
<td>99.8</td>
<td>98.6</td>
</tr>
<tr>
<td>MCF7 Vs Other Cancers</td>
<td>32</td>
<td>99.3</td>
<td>96.6</td>
<td>100.0</td>
</tr>
<tr>
<td>BT474 Vs MDA-MB231</td>
<td>42</td>
<td>97.4</td>
<td>91.7</td>
<td>100.0</td>
</tr>
<tr>
<td>MCF10A Vs MCF12A</td>
<td>42</td>
<td>91</td>
<td>97.1</td>
<td>62</td>
</tr>
</tbody>
</table>

The FFS-SVM framework is compared with other widely known methods for binary classification: SVM, PCA-SVM and PCA-LDA and the results are shown in Table 7-3. The first method SVM, trains the classifier with original data; PCA-SVM and PCA-LDA perform dimensionality reduction using Principal Component Analysis (PCA) and then trains SVM (or LDA [43]) on the transformed data. The number of principal components for classification are obtained by considering the top principal components that account for 70% of the total variance in the data. The top ten features obtained from Fisher criterion are used for classification in FFS-SVM framework. The classification accuracies are obtained using random subsampling technique with 100 iterations. The accuracies
obtained via FFS-SVM framework are comparable to the original accuracies reported in Table 7-2 (within 2%), thereby showing the features obtained from FFS are highly effective in discriminating the cell lines. All the methods including FFS-SVM exhibit high performance on the four classification tasks. It is interesting to note that employing SVM on the original data performs better in comparison with other methods which involve some form of dimensionality reduction. However, FFS-SVM framework, in addition to providing high classification accuracies, also yields important features which allow maximum discrimination of the cell lines or cell line groupings.

Table 7-3. Sensitivity, specificity and average classification accuracies of the four frameworks SVM, PCA-SVM, PCA-LDA and FFS-SVM for the four binary classification tasks. The classification accuracies are obtained from cross-validation using random subsampling (100 repetitions).

<table>
<thead>
<tr>
<th>Framework</th>
<th>Cancer vs. Non-Cancer</th>
<th>MCF7 vs. Other Cancers</th>
<th>BT474 vs. MDA-MB231</th>
<th>MCF10A vs. MCF12A</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>Accuracy(%) 99.2</td>
<td>100.0</td>
<td>97.6</td>
<td>93.4</td>
</tr>
<tr>
<td></td>
<td>Sensitivity(%) 100.0</td>
<td>100.0</td>
<td>94.8</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>Specificity(%) 99.4</td>
<td>100.0</td>
<td>99.5</td>
<td>80.6</td>
</tr>
<tr>
<td>PCA-SVM</td>
<td>Accuracy(%) 99.4</td>
<td>98.4</td>
<td>98.6</td>
<td>92.8</td>
</tr>
<tr>
<td></td>
<td>Sensitivity(%) 100.0</td>
<td>95.1</td>
<td>96.4</td>
<td>99.3</td>
</tr>
<tr>
<td></td>
<td>Specificity(%) 98.2</td>
<td>100.0</td>
<td>99.5</td>
<td>72.9</td>
</tr>
<tr>
<td>PCA-LDA</td>
<td>Accuracy(%) 99.5</td>
<td>98.3</td>
<td>96.4</td>
<td>85.8</td>
</tr>
<tr>
<td></td>
<td>Sensitivity(%) 99.9</td>
<td>98.9</td>
<td>88.2</td>
<td>82.8</td>
</tr>
<tr>
<td></td>
<td>Specificity(%) 98.6</td>
<td>97.6</td>
<td>99.3</td>
<td>96.6</td>
</tr>
<tr>
<td>FFS-SVM</td>
<td>Accuracy(%) 97.3</td>
<td>98.9</td>
<td>98.0</td>
<td>89.0</td>
</tr>
<tr>
<td></td>
<td>Sensitivity(%) 100.0</td>
<td>96.7</td>
<td>93.4</td>
<td>97.6</td>
</tr>
<tr>
<td></td>
<td>Specificity(%) 93.3</td>
<td>100.0</td>
<td>100.0</td>
<td>62.3</td>
</tr>
</tbody>
</table>

The relationship of classification accuracies of FFS-SVM framework to the number of features is shown in Figure 7-7 and is compared with two other frameworks: Random Feature-SVM (RF-SVM) and PCA-SVM. The first framework, RF-SVM, selects the features at random and subsequently performs classification using C-SVM. The other framework, PCA-SVM performs PCA on the data and chooses the top $k$ principal components to perform classification using C-SVM. Here, the number of principal components $k$ is varied between 1 and the total number of extracted features. The results are shown for the first classification task, Cancer versus Non-Cancer cell line
classification. The accuracies are calculated using repeated random sub-sampling. Clearly, FFS-SVM performs better than RF-SVM and yields accuracies similar to PCA-SVM. Additionally, PCA-SVM is expected to perform better than the other two frameworks since PCA utilizes all the features in estimating the principal components, while the other two methods utilize only a subset of features for classification. However, the FFS-SVM framework has the advantage in the much more efficient and effective interpretation of results, since it works in the original feature space by selecting a subset of features, while PCA works in a transformed space obtained from all the features. Therefore, FFS-SVM provides a way to classify cell lines with high accuracy (comparable to PCA-SVM) and also finds a subset of potentially biologically relevant features which maximally discriminate between the cell lines.

![Graph](image)

Figure 7-7. Classification Accuracy Comparison: The accuracies versus number of significant variables for the classification of Cancer versus Non-Cancer cell lines obtained from the three frameworks RF-SVM, FFS-SVM and PCA-SVM are shown. The x-axis shows the number of significant variables; features in the case of RF-SVM and FFS-SVM, and the principal components for PCA-SVM. All the accuracies are estimated using repeated random sub-sampling with 100 repetitions.
Biological validation of selected features: By applying our FFS-SVM framework to the four binary classification task resolved by the HCA, it was possible to develop a highly accurate classification model which can be applied on future classification of testing datasets while also yielding biological information about the differences between different groupings of the cell types. Comparison of five breast cell lines by Cuperlovic-Culf et al. (four of which were the same as used in the current study) was performed using NMR and comparing several clustering techniques and PCA; primarily focusing on lipophilic cellular extracts [30]. Remarkably, even though NMR has much higher specificity in regards to biochemical species, the Raman spectra were able to provide more accurate separation of the cell lines using HCA when discriminating among cancer versus non-cancer cell lines, as well as cancer sub-type clustering. A comparison of the results herein to those in the study by Cuperlovic-Culf et al. demonstrates the utility of the unique cell line-specific Raman spectral fingerprints and the amount of information these spectral fingerprints contain. Furthermore, this comparison exhibits the power of the FFS-SVM framework to efficiently derive distinguishing biochemical information that forms the basis for the class separation from the spectral features directly in the form of wavenumbers.

The subsequent biological analysis for each of the four HCA defined binary classification tasks based on the top ten class-discriminative selected features are discussed in the following subsections. The top ten discriminative features are discussed in terms of wavenumbers which have been correlated to biochemical species based on reference values found in the literature from other Raman spectroscopic studies of biological specimens.

Cancer versus non-cancer: The largest dissimilarity between two groups of spectra and thus cell lines was that between cancer and non-cancer cell types as can be seen in the dendrogram derived from hierarchical clustering shown in Figure 7-5. This is also the most apparent and broad classification task, resembling a typical cytological
Table 7-4. The top 10 features selected by FFS for the four binary classification tasks.

<table>
<thead>
<tr>
<th>Cancer vs. Non-Cancer</th>
<th>MCF7 vs. Other Cancers</th>
<th>BT474 vs. MDA-MB231</th>
<th>MCF10A vs. MCF12A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1047</td>
<td>811</td>
<td>823</td>
<td>1450</td>
</tr>
<tr>
<td>1341</td>
<td>986</td>
<td>1322</td>
<td>1405</td>
</tr>
<tr>
<td>1049</td>
<td>1063</td>
<td>760</td>
<td>1318</td>
</tr>
<tr>
<td>1047</td>
<td>1320</td>
<td>1156</td>
<td>941</td>
</tr>
<tr>
<td>823</td>
<td>760</td>
<td>830</td>
<td>1211</td>
</tr>
<tr>
<td>1658</td>
<td>830</td>
<td>1174</td>
<td></td>
</tr>
<tr>
<td>765</td>
<td>1085</td>
<td>1174</td>
<td>1621</td>
</tr>
<tr>
<td>1450</td>
<td>1318</td>
<td>941</td>
<td>1799</td>
</tr>
<tr>
<td>1660</td>
<td>1518</td>
<td>811</td>
<td>604</td>
</tr>
<tr>
<td>1066</td>
<td>604</td>
<td>1338</td>
<td>1129</td>
</tr>
<tr>
<td>622</td>
<td>1129</td>
<td>719</td>
<td></td>
</tr>
<tr>
<td>1086</td>
<td>785</td>
<td>967</td>
<td></td>
</tr>
</tbody>
</table>

analysis activity. The unsupervised clustering confirms that the cancer cell lines and non-cancer cell lines group together based on significant similarities and differences found within the Raman spectra. Further analysis by classification with supervised learning methods and feature selection also provides a measure of the degree of similarity to be established based upon cross-validation accuracy and the Fisher-based selected features which account for the class discrimination.

Here, the FFS-SVM framework demonstrates that Raman spectroscopy is able to detect clear differences between cancerous and non-cancerous cell lines. The random sub-sampling validation classification accuracy for classifying cancer versus non-cancer cells, after grouping all of the spectra from all cell lines together, is 97.3% with sensitivity of 100.0% and specificity of 93.3%. The top ten wavenumber features, in order of class discriminative power, are shown in the first column of Table 7-4. The selected feature at wavenumber 1047 cm\(^{-1}\) is the most discriminative and correlates closely to a glycogen peak [116]. The difference in glycogen content is an indicator of glucose consumption and characteristic for differentiating between cancer and non-cancer cell types as cancer cells are generally typified by having significantly higher glucose metabolism than non-cancer cells. Five of the top ten discriminative features (811, 823, 765, 829, and 785 cm\(^{-1}\)) all correlate to DNA and RNA vibrational modes including those of nucleic acids uracil, thymidine, and cytosine from the pyrimidine ring breathing, and the
phosphate backbone of DNA [34, 116]. Cancer cells typically have higher amounts of DNA, as well as RNA, due to their higher rates of proliferation [34, 116].

Bocklitz et al. has shown that differences in nuclei size and shape are defining differences between MCF12A and MCF7 cells when examined by Raman spectroscopy and applying SVM [13]. Additionally, Bocklitz et al. showed that two of the most distinguishing wavenumbers, which are assigned to nucleic acids, were 785 cm$^{-1}$ and 811 cm$^{-1}$. These findings are noteworthy because as shown in both the study herein and the study by Bocklitz et al., the same wavenumbers were found to be significant indicators of discrimination between cancer and non-cancer cell lines, even when including additional cancerous and non-cancerous cell lines. These specific features appearing in different studies, performed in different labs, using different instruments demonstrate the potential possibilities that the feature selected wavenumbers could be indicative of spectral biomarkers for distinguishing cancer cells from non-cancerous cells.

The wavenumber 1086 cm$^{-1}$ was also found to show significant differences between the two classes and is related to C=C stretch of fatty acid chains. The peak at 1450 cm$^{-1}$ that is indicative of lipids was also selected and has been shown in several previous studies to be related to invasive ductal carcinoma (IDC) [140, 147]. The feature at 1621 cm$^{-1}$ is due to the C=C bond vibrational mode and has been correlated to amino acids in structural proteins [116]. A difference among structural proteins was also indicated by the feature 1660 cm$^{-1}$, which is due to the Amide I vibration [52, 95, 147]. The peak at 1660 cm$^{-1}$ also originates from C=C stretch in lipids, particularly sphingolipids such as ceramide [96]. Moreover, ceramide has been shown to have elevated levels in cancerous cells, in particular those with high-level expression of enzymes regulating ceramide synthesis such as multi-drug resistant cancer cells [5, 105]. Ceramides have been shown to play a role in tumorigenesis as well as being involved in proliferation, survival and apoptosis [110]. The features 1086 cm$^{-1}$,
1450 cm\(^{-1}\), and 1621 cm\(^{-1}\) are possibly indicators of the differences in cell membrane composition, as well as cell morphology between the cancerous and non-cancerous cell lines. Again, the relevance of these features and the biological and biochemical correlations that these features play in discriminating among cancer and non-cancer cell lines could potentially lead to the discovery of 'Raman-based spectral biomarkers' for cytopathological post-classification analysis. Further studies will be needed to confirm the presence and relevance of such spectral biomarkers by investigating a large, comprehensive panel of different breast cell lines in order to identify universal Raman spectral features allowing for identification of cancerous cell types.

**MCF7 versus other cancer types:** Based on the HCA results, the second most dissimilar clustering originates from the differentiation of the MCF7 cells from the two cancerous cell types, BT474 and MDA-MB-231. The separation observed in the dendrogram of the MCF7 cells from the other cancerous cell types have an interesting biological basis due to the difference in the degree of tumorigenicity of the cancer cell lines. Subsequent classification by the FFS-SVM framework of the MCF7 cells versus the other highly invasive cell lines yields a classification accuracy of 98.9% with a sensitivity of 96.7% and specificity of 100.0% as shown in Table 7-2. Table 7-1 compares several characteristics between the MCF7, BT474, and MDA-MB-231 cell lines. From Table 7-1 and the corresponding references [87, 106, 118] it is known that BT474 and MDA-MB-231 cells are derived from highly invasive and aggressive tumors compared to the relatively low level invasivity of the MCF7 cell line. Moreover, the observed clustering therefore likely accounts for the strong similarity of the BT474 spectra and MDA-MB-231 spectra. The top-ten discriminative features selected for differentiating the MCF7 from the BT474 and MDA-MB-231 cell lines are shown in the second column of Table 7-4.

The majority of the features correlate to wavenumber assignments that are related to vibrations observed from structural proteins and the secondary protein structure;
particularly collagen, actin and the $\alpha$-helical secondary protein structure as well as amino acid residues such as valine and proline which are often found abundantly in structural proteins (e.g. collagen). The features 1066 cm$^{-1}$, 1316 cm$^{-1}$, 1322 cm$^{-1}$, 1341 cm$^{-1}$, and 1658 cm$^{-1}$ have all been previously assigned to collagen vibrational modes as well as amino acids and amide I vibrational modes [13, 81]. The peak at 1341 cm$^{-1}$ has been shown to correlate to the structural protein actin, and the majority of protein contribution of the Raman spectrum originates from collagen and actin [147]. Wavenumbers 1159 cm$^{-1}$ and 1405 cm$^{-1}$ originate from protein vibrational modes, and 622 cm$^{-1}$ is due to a specific vibrational mode of phenylalanine [34, 133]. Therefore, the composition, or more accurately the difference in composition, of these structural proteins is likely to give rise to the discrimination of the two classes of highly invasive versus non-invasive breast cell lines based on the Raman spectra. Increased levels of structural proteins, collagen and actin, have been associated with increased invasiveness and metastasis of breast cancer cells [36]. Furthermore, genes encoding for vimentin and actin have been shown to be highly expressed in such invasive cell lines as MDA-MB-231 based on a stromal-mesenchymal gene expression signature, but not expressed in the MCF7 cell line [10, 36]. The feature at 986 cm$^{-1}$ is thought to be due to the CH$_2$ stretch of lipids.

It should be noted that the feature at 1799 cm$^{-1}$ is adjacent to the end of the fingerprint range from which spectral data is collected and therefore was not evaluated. Due to the overwhelming proportion of the features related to structural protein differences between the invasive and non-invasive cell lines it is probable that the clustering and subsequent high accuracy of the cross-validation classification is due to the variation in cellular composition of the mentioned structural proteins.

**BT474 versus MDA-MB-231:** The two highly invasive cell lines, BT474 and MDA-MB-231, are shown to have the highest degree of similarity and cluster closely together based on the HCA-derived dendrogram in Figure 7-5. Even though these two
cell lines cluster with high similarity, the FFS-SVM framework is still able to classify the two cell lines from one-another with high accuracy. The classification accuracy for the classification of BT474 versus MDA-MB-231 is 98.0% with a sensitivity of 93.4% and specificity of 100.0%. This demonstrates the power of the FFS-SVM framework to be able to accurately classify and therefore differentiate among the two highly aggressive breast cancer cell subtypes despite having high clustering similarity. The top-ten discriminative features are shown in the third column of Table 7-4. BT474 and MDA-MB-231 have different morphologies and also have significantly different gene expression profiles, even though they are similar in their invasive nature [87, 118]. A comparison of selected characteristics can be found in Table 7-1. The most dominant feature, 1049 cm$^{-1}$, has been assigned to the vibrational mode of glycine. Furthermore, the feature at 760 cm$^{-1}$, which is assigned to tryptophan, and the vibrational mode at 830 cm$^{-1}$ which correlates to proline, hydroxyproline, and tyrosine which is indicative of difference of protein content. The feature at 1063 cm$^{-1}$ is strongly correlated to the C-C stretch of lipids [124]. This peak has been shown to have contributions from myristic acid, palmitic acid and other fatty acids; indicating a difference in fatty acid composition among the two cell lines. Additionally, the feature at 1129 cm$^{-1}$ is assigned to the C-C stretch of the acyl lipid backbone and correlates with palmitic acid, stearic acid and phospholipids [34, 116]. The feature at 1518 cm$^{-1}$ is assigned to the C=C vibrational mode and indicates a differences in carotenoid composition, which may be associated with membrane structure and fluidity [18, 79]. The feature at 1661 cm$^{-1}$ is associated with cis-isomer unsaturated fatty acids and may further indicate a difference in lipid composition between the cell lines. Both 1085 cm$^{-1}$ and 1318 cm$^{-1}$ are indicative of phosphodiester bond vibrational mode found in DNA [116].

Interestingly, the two IDC cell types, BT474 and MCF7, both of which are ER+ and PR+, are not found to cluster based on the expression of ER and PR as might be expected. Instead the AC sub-type, MDA-MB-231 (ER-, PR-), clusters more closely with
the BT474 cells based on the biochemical composition based on the Raman spectra. The separation of the MCF7 from the MDA-MB-231 are similar to the results from investigation of the cell lines by NMR for metabolic profiling; demonstrating again that Raman spectroscopy may provide information with similar impact as that of more commonplace metabolomic techniques [31]. Furthermore, combining Raman spectroscopy with techniques such as NMR through advanced data mining strategies, and data fusion, may allow for complimentary information to be extended from the massive datasets and therefore improve understanding of *in-vitro* breast cancer models.

**MCF10A versus MCF12A:** The MCF10A and MCF12A cluster together based on the dendrogram created by the HCA as seen in Figure 7-5, but the two cell lines have clear spectral differences as shown by the large branching distance at the level of the individual cell line clusters. The spectra were then classified by the FFS-SVM framework yielding an 89.0% classification accuracy with a sensitivity of 97.6% and a specificity of 62.3%, as shown in Table 7-2. This classification task results in the lowest classification accuracy of all the classification tasks attempted with the FFS-SVM framework. The high level of similarity between the two cell lines may give rise to the relatively low classification accuracy for this task. Furthermore, both cell lines are grown in the same media, under the same conditions and have similar doubling time, growth rates and thus most likely metabolic profiles. The top ten features selected for discriminating between the two non-cancerous cell lines are listed in the fourth column of Table 7-4. Again, the feature at 1047 cm\(^{-1}\), which is assigned to glycogen, is found to provide the greatest significant discrimination between the two classes. Differences in glucose consumption between the two cell lines may account for this difference, and further evidence of differences in carbohydrate content are provided by the feature at 941 cm\(^{-1}\) [116]. Although, as shown by Cuperlovic-Culf *et al.*, the metabolomic profile of these cell lines are quite similar in regards to lipophilic fractions based on NMR analysis [30]. In spite of this, several of the features listed for this binary
classification task have assignments related to fatty acids and lipids including 714 cm\(^{-1}\), 967 cm\(^{-1}\), and 1156 cm\(^{-1}\) (carotenoids) \cite{116}. Differences in the protein content based on selected features 1211 cm\(^{-1}\) and 1338 cm\(^{-1}\) may originate from the slightly different morphologies (Table 7-1). The features at 811 cm\(^{-1}\) and 1320 cm\(^{-1}\) correlate to the phosphodiester bond stretch of RNA and guanine vibrational modes respectively \cite{116}. The differences between the two non-cancerous cell lines are not very well defined by the Raman spectra as evidenced by the difficulty in accurate classification by the FFS-SVM framework. As these two cell lines are relatively similar and are both non-invasive, non-cancerous cell lines, with similar expression profiles it might be expected that this would be the most challenging classification task.
CHAPTER 8
CONCLUSION

In this work, we presented an overview of feature selection and classification techniques in the context of high dimensional datasets. Due to technological advances in the last decade, high dimensional datasets have been prevalent in many applications arising from fields like Bioinformatics, e-Commerce and Computer Vision. Several traditional classification algorithms are known to perform poorly on such high dimensional datasets due to curse of dimensionality. Additionally, in many biomedical applications, it is important to extract features that contribute to the differences among the classes. Hence, our present work focused on building scalable efficient classification models for high dimensional datasets that would also be able to extract important features in addition to accurately predicting the class of unknown samples.

A new class-specific embedded feature selection method called Sparse Proximal Support Vector Machines (sPSVMs) is proposed in Chapter 4. sPSVMs learns a sparse representation of PSVMs by first casting it as an equivalent least squares problem and then inducing sparsity using the $l_1$-norm on the coefficient vector. An efficient algorithm based on alternate optimization techniques is proposed. Numerical experiments on several publicly available datasets verify the good classification performance and efficiency of sPSVMs. Feature selection stability of sPSVMs is also studied and compared with other univariate filter techniques. Moreover, sPSVMs successfully eliminate more than 98% features irrelevant to classification for high dimensional datasets and show consistency in the feature selection process. Additionally, our method offers the advantage of interpreting the selected features in the context of classes.

Chapter 5 explores the possibility of combining feature selection with subspace learning for dimensionality reduction in high dimensional datasets. Current approach involves transforming the problem formulation to an equivalent convex relaxation and
using efficient alternate optimization techniques to solve it optimality. Though the current approach is efficient, it limits explicit control in the level of sparsity. Hence, we propose an alternate approach of solving the original problem to local optimality using Clayley transformation. This approach, in addition to being efficient, also allows for explicit control in the level of sparsity which is crucial while performing classification tasks on high dimensional datasets.

Constrained subspace classifiers (CSC) for high dimensional datasets are discussed in Chapter 6. CSC improves on an earlier proposed subspace classifier Local Subspace Classifier (LSC). In addition to approximating the classes well by local subspaces, CSC also accounts for the relative angle between the subspaces by utilizing the projection metric. An efficient alternate optimization technique is proposed. CSC has been evaluated on publicly available datasets and is compared with LSC. The improvement in classification accuracy shows the importance of considering the relative angle between subspaces while approximating the classes. Additionally, CSC seems to be effective for lower dimensional subspaces.

A high dimensional data application of characterizing Breast cell lines using Raman Spectroscopy is presented in Chapter 7. We developed a robust method capable of classifying Raman spectra at different levels of cell line groupings based on the biological characteristics contained within the spectra. In order to identify the biological characteristics contained within a Raman spectrum and which are used as the basis for classification, a method for selecting and extracting these features directly from the spectra is needed. A method employing a Fisher-based criterion for selecting the features is implemented for classification while being mindful of the curse of dimensionality, and thus tunable parameters were incorporated into the framework allowing for the number of features to be selected by the user as well. The purpose of developing a generalizable feature selection technique for Raman spectroscopy is based on the idea that spectral biomarkers for cancer and cancer sub-types may
likely exist. This assumption is based upon the fact that other spectroscopic methods, which produce complex, rich spectra, such as NMR and mass spectrometry, are used to discover and identify cancer biomarkers regularly. Therefore, Raman spectroscopy should also have the potential to yield biomarkers, although these biomarkers being different in nature, due to differences in specificity and sensitivity, thus being more comparable to a fingerprint.

Traditionally, feature selection has been employed prior to classification to handle high dimensional data classification problems. However, recently, the focus has been shifted to applying regularization techniques to traditional classification algorithms for better generalization performance. Though the progress made so far is encouraging, we believe that high dimensional data classification would continue to be an active area of research as technological innovations continue to evolve and become more effective.
REFERENCES


BIOGRAPHICAL SKETCH

Vijay Pappu was born in Visakhatnam, Andhra Pradesh, India. He received a bachelor’s degree in Mechanical Engineering from Indian Institute of Technology (IIT), Chennai, India in 2006. He joined Rutgers, The State University of New Jersey, in 2006 for a Master in Science in Mechanical Engineering, where he specialized in Computational Fluid Mechanics. After finishing master’s program at Rutgers in 2008, he joined MathWorks Inc., in Natick, MA as an Application Support Engineer. In 2010, he joined the Industrial and Systems Engineering (I.S.E) doctoral program at University of Florida (UF). He received a Master of Science in 2011 and Ph.D. in 2013 from I.S.E. He also received Master of Science in Management from Warrington School of Business at UF in 2013. His research interests include applications of data mining and machine learning techniques to real world problems. His work has been published in peer-reviewed journals and conference proceedings. He is the editor of a special issue for Energy Systems and has also co-edited two volumes published by Springer book series. He has co-organized two conferences at UF related to applications in Biomedicine and Smart Grid systems. He was the Captain and the President of Gator Cricket Club at UF in 2011, during which UF Cricket team was ranked 2nd in the nation. He was awarded the Outstanding International Student award from UF International Center in 2011, Scholar Athlete award from department of RecSports at UF in 2012 and the honorable mention award for Seth Bonder Scholarship for Applied Operations Research in Health Services, INFORMS in 2012. He is student member of Institute of Operations research and Management Science (INFORMS) and Institute of Electrical and Electronics Engineers (IEEE).