OPTIMIZATION MODELS AND ALGORITHMS FOR SAMPLE-PRESERVED CLASSIFICATION AND CLUSTERING

BY YA-JU FAN

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Industrial and Systems Engineering

Written under the direction of

Wanpracha (Art) Chaovalitwongse

and approved by

New Brunswick, New Jersey

May, 2010

ABSTRACT OF THE DISSERTATION

Optimization Models and Algorithms for Sample-Preserved Classification and Clustering

by Ya-Ju Fan

Dissertation Director: Wanpracha (Art) Chaovalitwongse

This dissertation presents the development of new optimization models and algorithms for sample-preserved classification and clustering. A sample-preserved method keeps some or all of the existing samples when training a rule for classification or clustering, and continues to use them in the testing or predicting phase. Developing a sample-preserved method provides the capability of analyzing time series data due to the largely applied similarity measures on time series.

A proposed sample-preserved classification technique, called Support Feature Machine (SFM), finds an optimal combination of features that gives the best classification based on the nearest neighbor rule. It keeps all baseline samples of the selected features in the predicting phase. Variations of SFM models are also presented. In addition, the bilinear program sample-preserved k-median (BPSPKM) clustering algorithm is introduced. While the original k-median problem can be solved by a simple and efficient bilinear program algorithm, it does not have the sample-preserved property, and only works with the 1-norm distance. The sample-preserved k-median (SPKM) clustering method is formulated as an integer programming problem, which is very hard to solve. A bilinear program algorithm is herein proposed in order to obtain local optimal solutions of the SPKM clustering method, as well as a new sequential search algorithm that can solve the SPKM clustering more efficiently. Finally, a novel feature space sample-preserved k-median (FSSPKM) clustering algorithm is proposed, as well as feature selection methods tailor made for such clustering technique.

The experimental results show that the original k-median clustering fails to classify time series data due to the lack of the sample-preserved property, and the utilization of time series similarity measures. The sample-preserved medians can avoid having invalid values in some application domains and can be used to represent the samples in the clusters. The BPSPKM clustering algorithm with the Euclidean distance is suggested for clustering attribute (non-time series), univariate time series and multivariate time series data sets. Furthermore, the proposed feature selection methods consider the distances between cluster centers and cluster densities. The results show that the proposed algorithms outperform other feature selection techniques used in the original k-median methods.

Acknowledgements

Many people have contributed to the completion of this dissertation. I owe my deepest gratitude to all those people who helped me make this dissertation possible.

First I would like to thank my advisor Wanpracha (Art) Chaovalitwongse, who provided me sufficient research resources so I can efficiently conduct large experiments. He was always there to listen and to give advice. He taught me how to present research ideas and how to write a convincing paper.

I am also greatly indebted to teachers in the past. This is a great opportunity to express my respect to my first advisor in the United States, Emeritus Professor Stephen M. Robinson. He supported me generously and provided me with a lot of opportunities in the University of Wisconsin-Madison. He is my role model of being a professor. He showed me how to best organize teaching material and give challenging practices. He also taught me how careful research statements should be examined.

I am also thankful to Michael C. Ferris, who introduced me to the area of optimization, algorithms and application on data mining. He taught me how to approach a research problem and discover efficient algorithms. He also showed me how to be a dependable researcher.

I am grateful to Andrew J. Miller for his lectures on related topics that helped me improve my knowledge in the area. He also showed me how new ideas can be made through experiences.

I would like to thank David W. Coit for being a friend and teaching me how to survey research topics and find useful ideas.

I would also like to thank the rest of my thesis committee: Elsayed A. Elsayed who gave me valuable advice in pursuing a Ph.D. degree and helped me overcome many crisis situations. I am thankful to Susan Albin for her encouragement and challenging questions. I am also thankful to Hoang Pham, who helped me set up dissertation requirements. I am grateful to Endre Boros, who gave insightful comments and reviewed my work on a very short notice.

I am pleased to thank Kerem Akartunali and Rikhardur Einarsson who studied with me in Madison and taught me how to work hard and play hard. I also enjoyed hot chocolate breaks with them after class. They helped me adjust to the new environment in the United States. Kerem also helped me with my dissertation ideas from Australia. He encouraged me when I was frustrated and kept me in a right direction.

A special thank goes to my friend, Elan Borenstein, for helping me complete the final research of this dissertation as well as the key ideas that lie behind it. More importantly, he took me to enjoy chicken wings every Wednesday night after working hard, and taught me how to skate and snowboard to reduce stress! Without his encouragement and constant guidance, I could not have finished this dissertation.

I would like to thank Zhe Liang for sharing an office with me, maintaining a printer and bringing a brand new coffee machine, which allowed me to work in a very nice office for four years. He also gave me insightful suggestions on my research whenever I need them.

I would also like to thank Chun-An Chou for managing servers and updating software in the lab, which helped me run programs easily and efficiently.

Let me thank the following friends. They helped me stay focused on my graduate study. Their friendship is very important to me and I deeply appreciate their care and support. Muhammad Dayhim had invited me to countless events since I was in Rutgers, which provided me with many happy and relaxing times. Davood Golmohammadi gave me guidelines to solve unsolvable problems concerning a Ph.D. student. He valued my work and encouraged me to continue. Ali Rezvani, Ibrahim Gultekin, Alper Almaz, Andrew Rodriguez and Tsvetan Asamov helped me stay sane through these difficult years. Haleh Valian, Jing-Yi Zheng, and Yada Zhu were always there for me. Thank you all for being my friends!

Finally, I would like to express my heartfelt gratitude to my family. This dissertation is dedicated to my parents for their unconditional love and support. My brother Po-Chun Fan always loves me and helps support me financially when I went to Taiwan for vacation. My sister Tiffany Ya-Ting Fan constantly encourages me and believes in me no matter what.

Ya-Ju Fan

Piscataway, NJ

April 2010

Dedication

To my parents.

Table of Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

This dissertation focuses on developing optimization models and algorithms for multivariate data classification and clustering with a sample-preserved property.

A data set can be one of three types: attribute (or non-time series) data, single (or univariate) time series data, or multivariate (multidimensional or multi-attribute) time series data. The term non-time series is referred to static data and used to distinguish them from time series data. Non-time series data are those whose samples can be represented as real vectors, called *attribute* data. Time is not a factor, but there are other attributes that are varied. For example, a non-time series data set may contain attributes: age, weight and height, which are real-valued. An instance may be represented by a vector that contains these three elements. *Single* time series data have values of a fixed attribute, but vary with time. For example, stock prices vary with time and form a time series. *Multivariate* time series (MTS) data have both properties; having multiple attributes, and varying with time. Each individual sample is a matrix with information from two characteristics: features (spacial property) and time (temporal property). For example, sensor data may consist of multiple series of observations from recordings of multiple sensors over a period of time. Each sensor counts an attribute (or a feature) of a time series. A combination of these recordings forms a multivariate time series. Note that both attribute data sets and MTS data sets are multivariate.

A data set may contain a number of distinct groups (classes). For example, samples in a two-group data set may be labeled with “positive” or “negative”. If class labels are used in a learning technique, it is said to be *supervised.* A classifier is designed according to sample labels. Data *classification* can be done by training a classifier that can separate labeled samples into their belonging groups, and then applying the classifier to assign suitable labels on other samples whose class labels are unknown.

On the other hand, *unsupervised* learning does not use any class labels. It tries to find unknown groups in a collection of samples without categories (classes). Data *clustering* is one such method that groups "similar" observations together without knowing their class labels. The unsupervised methods can be used to find features that will then be useful for categorization (classification). It can also provide valuable exploratory data analysis in the early stages of an investigation, and gain some insight into the structure of the data for preprocessing.

This dissertation deals with both classification and clustering techniques through optimization models for multivariate data analysis. The proposed techniques are sample-preserved methods. A sample-preserved method keeps some or all of the existing samples when training a rule for classification or clustering, and continues to use them in the testing or predicting phase. Baseline samples are the samples whose class labels are known, and are taken as a base for measurement. Methods that keep the baseline samples have the sample-preserved property. A simple sample-preserved classification method is the k-nearest neighbor rule, which uses all baseline samples when classifying new ones. Using this method, the "pattern" of a sample is kept in the model. Developing a sample-preserved method especially provides the capability of analyzing time series data due to the largely applied similarity measures on time series. Keeping the "pattern" of a sample makes it easy to evaluate its similarity to others.

1.1 Motivation

Medical diagnosis is the process of identifying a medical condition or disease by its symptoms and from evaluated results of various diagnostic procedures. By taking each patient as an instance, those symptoms and evaluated results of patients can be viewed as their attributes (or features), thus forming a medical data set. The medical data set typically consists of two groups of patients; one is considered abnormal (or "positive", with a certain symptom or disease) and the other is considered normal (or "negative", without a certain symptom or disease). Physicians make decisions according to those symptoms and evaluated results.

A classification technique constructs a decision rule (classifier) that can be used as a tool for medical diagnosis. The classifier is trained according to labeled data whose class labels are known. It can then be used to classify unlabeled samples (patients) whose class labels are not known.

Today's clinical testings and experiments to diagnose patients have resulted in massive data sets, which physicians have to mine so they can accurately diagnose the patients. These data ranges from simple blood pressure and heart rate, to magnetic resonance imaging (MRI) and electroencephalogram (EEG) waveforms. In such situations, physicians need a tool to quickly analyze the medical data signals, and detect the patterns that can be used to identify the causes of the symptoms or diseases.

A clustering technique groups similar instances together, which can speed up the labeling process for a large, unlabeled data set. It may also be helpful for gaining some insight into the structure of the data. In addition, feature selection in clustering may help reduce the size of a data set and improve the performance of a classifier. Moreover, similarity measures are especially useful for analyzing time series types of data. Clustering techniques that come with such similarity measures can hence be naturally applied on time series.

1.1.1 Time Series Analysis

Time series analysis can generally be performed in two major domains: the frequency domain and the time domain (Figure 1.1). The frequency domain of time series is studied especially in the area of signal processing. Techniques such as Fourier transformation are widely used for signal classification. Extending methods of the fourier transformation, for example, are fourier kernels proposed by Vapnik (1998) [92], and the time-frequency kernel given by Davy et al. (2002) [29]. In frequency domain, behavior of time series data is approximated by a frequency function. Then the analysis is performed based on the estimated function. In this way, frequency of the temporal properties of time series is extracted. However, some important temporal properties

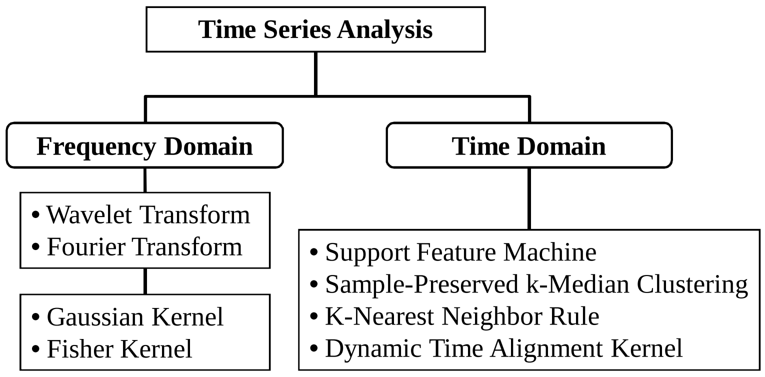

Time Series Analysis
Frequency Domain
Time Domain
• Wavelet Transform
• Fourier Transform
• Gaussian Kernel
• Fisher Kernel
• Support Feature Machine
• Sample-Preserved k-Median Clustering
• K-Nearest Neighbor Rule
• Dynamic Time Alignment Kernel

Figure 1.1: Two Major Domains of Time Series Analysis and Examples of Their Techniques.

may be lost after fourier transformation, since only the frequency is considered.

This dissertation does not include the area of frequency domain. Instead it is focused on the time domain, where the temporal properties of time series is examined directly. More detailed time series analysis techniques in time domain is addressed in Chapter 3.

1.1.2 Optimization Models

Many classification and clustering techniques based on optimization models have been developed for time series data sets and attribute (non-time series) data sets. The most well known optimization model for classification is the Support Vector Machine (SVM), which was originally designed for attribute data sets. It projects samples in a high dimensional space, and constructs an optimal hyperplane that separates the samples according to their labels. There exists many hyperplane that can separate most of the samples. The term "optimal" indicates that the hyperplane is located at the direction in the way that the distance (called *margin*) from the hyperplane to the closest samples from the two groups is the largest possible one. The optimal hyperplane is a classifier that can be used to discriminate unlabeled data. SVM has been successfully applied on breast cancer diagnosis and prognosis by a linear programming formulation [70]. To classify a single time-series data set, there are techniques especially developed to capture the temporal properties of time series samples. For instance, time series can be embedded in a vector space, then the time series data set can be treated as an attribute data set for classification [44]; or it can be dynamically aligned with time sequentially in the projecting function of SVM, and then an SVM classifier is constructed [85]. However, SVM does not have the sample-preserved property. Its advantage is that it does not need any baseline sample when classifying a new sample. Only an optimal hyperplane is needed to label a new sample.

A well known classification technique that has the sample-preserved property is the Logical Analysis of Data (LAD) [43]. It is especially suited for medical data sets. LAD is used to find meaningful values with a set of important features. These values are called patterns, which represent a specific group (positive or negative). A positive pattern does not appear in any negative sample. Similarly, a negative pattern can not be found in any positive sample. Combinatorial optimization techniques are used to find such patterns. Besides SVM and LAD, there are still a lot methods in the literature for classifying attribute data sets, but there is a very limited number of studies that can deal with MTS data sets.

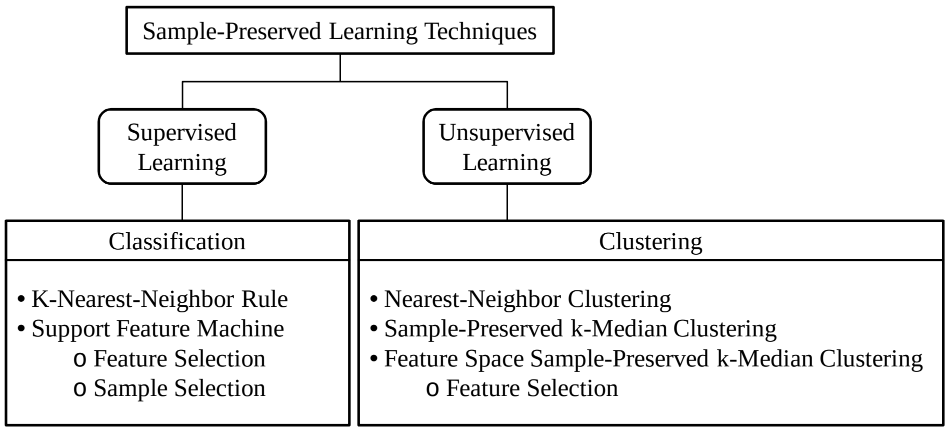
Well known clustering techniques such as the k-means clustering and the k-median clustering can be formulated as optimization models, shown in [13] and [15], respectively. However, the k-means clustering is based on squared Euclidean distances. Also, the k-median clustering is limited to 1-norm distance measure. If Euclidean distances or p-norm, $p \neq 1, \infty$ is applied, the problem becomes very hard to solve [15]. This causes a problem for time series clustering, since there are many time series similarity measures needed for time series analysis. In addition, both models are designed for attribute data, not for MTS.

MTS data sets are common in several application domains such as medical applications, financial applications and the process industry. Most of these data sets have large amounts of data and need to be interpreted. They are analyzed for classification, identification, and prediction of diseases, certain behaviors, or various functional states. Examples in medical applications are electroencephalogram (EEG), electrocardiogram (ECG), and various attribute values collected from patients that are generated from multiple sensors or measurements over a period of time. They are collected in order to provide information for medical diagnoses and treatment. In financial applications there are corporate bonds and interest rates series, housing starts and sold series. They may be used to predict possible occurrence of events, e.g., economic depression, which is still a challenging problem. In industrial plants, especially chemical, data are collected constantly in order to monitor stages of processes and to detect abnormal situations.

As described above, there are many applications related to MTS type of data. However, there is a need for the development of new methods. Therefore, the main motivation of this dissertation is to apply optimization techniques in order to construct an optimal classifier, and find optimal clusters for multivariate data sets; especially for MTS.

1.2 The Contribution of the Dissertation

This dissertation provides an optimization framework for improving multivariate data classification and clustering. Figure 1.2 shows the developed classification and clustering techniques that are applied or proposed here.



Sample-Preserved Learning Techniques
Supervised Learning
Unsupervised Learning
Classification
• K-Nearest-Neighbor Rule
• Support Feature Machine
o Feature Selection
o Sample Selection
Clustering
• Nearest-Neighbor Clustering
• Sample-Preserved k-Median Clustering
• Feature Space Sample-Preserved k-Median Clustering
o Feature Selection

Figure 1.2: Proposed Sample-Preserved Learning Techniques.

The proposed methods are especially *sample-preserved*, which means some or all of the original samples are kept when developing and applying the classifier and the clusters. The main idea that can keep the original patterns of samples is to use distance matrices. Using the distance matrices also allows variant distance measures to be chosen. Due to this flexibility as well as the property that cluster centers are existing samples, it is easy to incorporate its use with multivariate time series data as opposed to only vector space attribute data.

Suppose that there are n samples, and each has m features in a data set. An intra-class distance matrix is an $n \times m$ matrix. Each element in the matrix is an average distance from sample i to all other samples in the same class at a feature space j, for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. In addition, each element in an inter-class distance matrix is an average distance from sample i to all samples in the different class at a feature space j, for all i, j. With these distance matrices, the "shape" of data, especially time series, can be preserved. Hence, techniques using distance matrices are sample-preserved methods.

Support Feature Machine (SFM) includes these distance matrices in its optimization formulation, and is designed especially for MTS classification. It has been shown to be able to classify EEG MTS data sets [20]. In SFM optimization models, the decision to choose an optimal combination of features is made while a best classification accuracy is achieved. Values from all baseline samples at the optimally selected features are then used in the decision rule. Therefore, SFM has the sample-preserved property. Observe that class labels are incorporated in these distance matrices, so SFM is a supervised learning technique.

A sample-to-sample distance matrix is an $n \times n$ matrix. Each element in the matrix is a distance from a sample i to the other sample i', for $i, i' = 1, \ldots, n$. Note that such distance matrix is symmetric with zeros at its diagonal. Again, with such a distance matrix, the "shape" of data, especially time series, can be preserved.

Sample-preserved k-median (SPKM) clustering applies this distance matrix to group samples into k clusters. The feature space sample-preserved k-median (FSSPKM) clustering extends the distance matrix into a three-dimensional $n \times n \times m$ matrix, which contains a $n \times n$ sample-to-sample distance matrix at each feature space s, for $s = 1, \ldots, m$. Feature selection algorithms are then developed using this extended matrix. The structure of multivariate data in each feature space is examined. The best combination of features can then be found for possible improvement of classification.

1.3 Dissertation Organization

This dissertation is organized as follows:

In **Chapter 2**, techniques designed for attribute data sets are discussed. A brief review of popular classification and clustering techniques, which are based on optimization formulations is presented. Purposes of optimization models include finding logical class patterns in a data set, constructing a hyperplane classifier, and approximating likelihood function values. Since data transformation may be critical for a classification decision rule, most commonly applied kernel function and dimensionality reduction techniques are introduced. In addition, clusters can be optimized by aiming at grouping most similar samples together according to distances among samples.

In **Chapter 3**, time series analysis techniques that are commonly used, and time series classification approaches developed recently are introduced. The most important time series analysis technique is the time series similarity measure. Measures used to estimate similarity of both univariate time series and MTS are addressed. Based on a suitable similarity measure, the nearest neighbor rule is an intuitive classification method that applies similarities as distances in a space. Studies related to the improved nearest neighbor rule as well as feature extraction techniques for time series classification are included.

In **Chapter 4** and **Chapter 5**, novel optimization models for classifying multivariate data sets are presented together with its applications on real world data sets. Chapter 4 contains a description of using the intra-class and inter-class distance matrices in the classification framework of the optimization technique, SFM, and how it is based on the nearest neighbor rule. Its classification effectiveness depends on a choice of similarity measure. In SFM optimization models, a set of features is selected while a highest classification accuracy is achieved. SFM has variate formulations. The standard SFM model uses binary variables to decide which features are selected. Models with relaxation and normalization on the decision variables used for feature selection are also proposed.

Classification performances of SFM models are discussed in Chapter 5. Methods designed for evaluating classification performances are also addressed. SFM models are implemented and applied on multivariate data sets. They are attribute and MTS data sets. Sample classes (e.g., positive and negative) are used to train SFM classifiers, and then classification performances are evaluated.

Chapter 6 and **Chapter 7** contain the description and the use of the proposed clustering techniques, as well as feature selection algorithms for clustering. In Chapter 6, novel optimization models, SPKM and FSSPKM clustering techniques are proposed in order to investigate the structure of multivariate data sets. SPKM incorporates a sample-to-sample distance matrix in the whole feature space, while FSSPKM incorporates sample-to-sample distance matrices, each at a single feature space. Both methods have two advantages. One is that a distance matrix provides a flexibility of choosing a distance measure. The other is that the sample-preserved property gives the model the ability to cluster time series due to the kept "shape" of time series. Moreover, feature selection algorithms are proposed that finds possible improvements of group classification using the FSSPKM model.

In Chapter 7, both SPKM and FSSPKM clustering performances are discussed. More data sets are used to evaluate the proposed models. They are attribute data, single time series data, and multivariate time series data. Moreover, results from feature selection FSSPKM algorithms are also presented. Classification error curves and clustering error curves are compared in order to find optimal choices of features that can provide possible improvement in classification.

Finally, in **Chapter 8**, ideas for future research opportunities are presented. They are aimed at applying optimization techniques, and constructing new methods to improve sample-preserved classification and clustering. One may extend the proposed techniques into stronger pattern-based methods, such as identifying specific sample-preserved patterns in order to represent a specific group of data. One may also construct novel classification or clustering models for different analytical purposes, e.g., regression analysis or hierarchical clustering. These new methods may continue to improve machine learning techniques and contribute to the research community.

Chapter 2

CLASSIFICATION AND CLUSTERING WITH OPTIMIZATION MODELS

The goal of this dissertation is to apply optimization techniques for multivariate data classification and clustering. In the literature, many optimization models are used for classifying and clustering attribute (non-time series) data sets. An attribute data set can be represented by an $n \times m$ matrix, where n is the number of samples and m is the number of features. In this matrix, each column represents a particular feature (or attribute), while each row represents a sample in the data set with a value at each feature. Mathematically, a sample can be viewed as a vector, and each element in a vector corresponds to a specific feature. Suppose that an attribute data set consists of two disjoint groups (i.e., one group is "positive" and the other is "negative"), then each case in the data set is labeled according to the group it belongs to. If class labels are used in a learning technique, it is said to be *supervised.* A decision rule (classifier) is constructed based on the labeled data, called training data. It is then used to classify new observations in a way that is consistent with classification in the past (training).

On the other hand, *unsupervised* learning does not use any class labels. It tries to find unknown groups in a collection of samples without categories (classes). Data *clustering* is one such method that groups "similar" observations together without knowing their class labels. The unsupervised methods can be used to find features that will then be useful for categorization (classification). It can also provide valuable exploratory data analysis in the early stages of an investigation, and gain some insight into the structure of the data for preprocessing.

A brief review of popular classification and clustering techniques, which are based on optimization formulations is presented. Since data transformation may be critical for a good decision rule, most commonly applied kernel function and dimensionality reduction techniques are introduced.

2.1 Classification with Optimization Models

Optimization techniques have been widely applied to construct decision rules. In this section some of the most recent developed techniques are reviewed. Combinatorial optimization models are used in Logical Analysis of Data (LAD) [27, 10, 11]. Linear programming formulations are used to find a classification hyperplane in Support Vector Machines (SVM) [69]. Nonlinear optimization with convex objective function and linear constrains is used in a regression model, called Likelihood Basis Pursuit (LBP) model [100]. Moreover, a multi-group probability-constrained discrimination method [2] is formulated as a nonlinear programming problem with nonlinear objective functions.

2.1.1 Logical Analysis of Data (LAD)

Instead of viewing data sets as points in a vector space and constructing a hyperplane as a classifier, LAD tries to find minimal sets of features necessary for classification, such that positive and negative sample values will not coincide. In addition, combinations of feature values from the optimal set are found, which represent positive or negative samples the best. These combination of feature values form logical patterns that can be used for further classification and can be explained by human experts. Therefore, LAD is a pattern-based decision technique. Moreover, those feature values are from existing samples, so LAD is a sample-preserved method.

The original LAD technique is used only for binary data proposed in [27]. To cope with numerical data, a binarizing method is proposed in [10, 11]. As a result, LAD deals with numerical data that have been transformed into binary values. The pattern characteristics found by LAD can be easily explained, and therefore it is a useful technique in practice, especially in medical diagnosis [43].

Given a binary data set, which can be obtained by binarizing a numerical data set or is generated naturally, there may be a number of redundant attributes. The main purpose of LAD is to eliminate sets of redundant variables, and then find meaningful patterns accordingly. Two combinatorial optimization models are formulated to reach this goal. First is to find the minimal support set defined as follows.

The data set consists of two disjoint sets Ω^+ and Ω^- of d-dimensional binary vectors. A *support set* is a set S of attributes such that Ω_S^+ is disjoint from Ω_S^-, where Ω_S^+ is the projection of Ω^+ on S and Ω_S^- is the projection of Ω^- on S. Since in the data set Ω^+ and Ω^- are disjoint, the complete set $D = 1, \ldots, d$ is a support set. A *minimal support set* is a support set such that if any one of its variables is eliminated, the remaining data set will have same observations that belong to both positive and negative groups.

A combinatorial optimization model is formulated in order to find the minimal support set. Each binary decision variable y_i is associated with an attribute i, for $i = 1, \ldots, d$.

$$y_i = \begin{cases} 1, & \text{if variable } i \text{ belongs to the support set;} \\ 0, & \text{otherwise.} \end{cases}$$

Moreover, each positive observation is represented by a vector $U = (u_1, \ldots, u_d)$ and each negative observation is represented by a vector $V = (v_1, \ldots, v_d)$. Then a vector is constructed as $(w_1(U, V), \ldots, w_d(U, V))$, which is associated to a pair of a positive and a negative observation. Here,

$$w_i(U, V) = \begin{cases} 1, & \text{if } u_i \neq v_i \\ 0, & \text{otherwise.} \end{cases}$$

The condition that Ω_S^+ and Ω_S^- are disjoint and y_i's relate to a support set is equivalent to stating that for any $U \in \Omega_S^+$ and $V \in \Omega_S^-$,

$$\sum_{i=1}^{d} w_i(U, V) y_i \geq 1. \tag{2.1}$$

As a result, by minimizing the sum of y_i's over condition 2.1, the minimum size support set can be obtained. It is the set covering problem formulated as

$$\begin{aligned}
\min_{y} \quad & \sum_{i=1}^{d} y_i \\
\text{s.t.} \quad & \sum_{i=1}^{d} w_i(U,V) y_i \geq 1, \text{ for all } U \in \Omega_S^+ \text{ and } V \in \Omega_S^- \\
& y \in \{0,1\}^d.
\end{aligned}$$

After getting rid of redundant variables, the next step of LAD is to find meaningful patterns. A *positive pattern* is a subcube of the unit cube which is in Ω_S^+ and is disjoint from Ω_S^-. Similarly, a *negative pattern* is a subcube of the unit cube that lies in Ω_S^- and is disjoint from Ω_S^+. For $\omega \in \{0,1\}^d$ a positive ω-pattern is a pattern covering ω. A *maximum positive ω-pattern* P is a positive ω-pattern where the coverage ($|P \bigcap \Omega_S^+|$) is the largest. Similar definition applies to a *maximum negative ω-pattern.* The second combinatorial optimization problem is then formulated to find a maximum ω-pattern for each observation ω in the data set.

The ω-pattern is represented by a binary vector $\omega = (\omega_1, \ldots, \omega_d) \in \Omega_S^+$ is associated to an elementary conjunction C. A binary decision variable y_i, for $i = 1, \ldots, d$ is defined as follows:

i. $\omega_j = y_j = 1$ indicates that x_j is included in C.

ii. $\omega_j = 0$ and $y_j = 1$ indicate that $\bar{x_j}$ is included in C.

iii. $y_j = 0$ means that none of x_j and $\bar{x_j}$ is in C.

A toy example is when $\omega = (1,0,0,1,1,1)$ and the decision variable $y = (0,0,1,1,0,1)$, the conjunction C is $\bar{x_3} x_4 x_6$.

Consider every negative observation $\rho \in \Omega_S^-$. The definition of positive ω-pattern indicates that it should not cover any observation from Ω_S^-. Therefore, when $\rho_j \neq \omega_j$ every variable y_i should equal to 1 for at least one of those j's. That is

$$\sum_{\substack{j=1 \\ \rho_j \neq \omega_j}} y_j \geq 1, \text{ for every } \rho \in \Omega_S^-.$$

Also consider every positive observation $\sigma \in \Omega_S^+$. Each σ is covered by the ω-pattern if and only if $y_i = 0$ for all those indices j where $\sigma_j \neq \omega_j$. The objective is to find the

ω-pattern that has the maximum coverage. It is equivalent to minimize the number of σ's such that $\sigma_j \neq \omega_j$. The objective is then to minimize:

$$\sum_{\sigma \in \Omega_S^+} \prod_{\substack{j=1 \\ \sigma_j \neq \omega_j}} \bar{y_j}.$$

The following polynomial set covering problem is then formulated to find the maximum ω-pattern:

$$\begin{aligned} \min_{y} \quad & \sum_{\sigma \in \Omega_S^+} \prod_{\substack{j=1 \\ \sigma_j \neq \omega_j}} \bar{y_j} \\ \text{s.t.} \quad & \sum_{\substack{j=1 \\ \rho_j \neq \omega_j}} y_j \geq 1, \text{ for every } \rho \in \Omega_S^-. \end{aligned}$$

2.1.1.1 An Example of Logical Analysis of Data

Table 2.1: Example Data Set

Sample	F1	F2	Class Label
1	1	0	+
2	1	0	+
3	0	1	+
4	0	1	+
5	1	1	+
6	1	1	-
7	0	0	-
8	0	0	-
9	0	0	-
10	0	0	-

A data set shown in Table 2.1 contains ten samples. Each sample has values of two features (F1 and F2) and a class label. Since there are only two features in the data set and all values are binary, only four possible combinations of the feature values can appear in the data set. One can then label each combination of values with class labels of samples that contain those values. Figure 2.1 displays such labels.

Observe that both positive and negative samples contain the combination $[1, 1]$ and hence it can not be a pattern for classification. One may also observe that $[0, 0]$ only appears in negative samples. It may be a possible negative pattern. Once the number of features in a data set increases, it becomes harder to observe such patterns graphically.

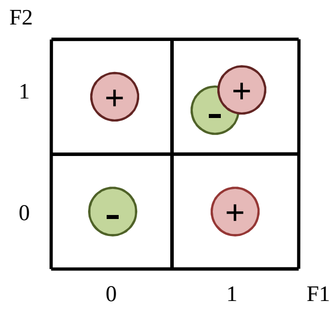


Figure 2.1: Positive and Negative Samples in a Two-Dimensional Space.

Combinatorial optimization techniques are needed to handle data sets with multiple features.

2.1.2 Support Vector Machines (SVM)

SVM is a well known classification method, which is aimed at finding an optimal hyperplane that separates labeled data into two groups, say A and B. The term "optimal" is used because a set of data of two groups may have many possible separating planes. SVM only finds one separating hyperplane that has the largest margin. The margin is defined as the minimum distance from the hyperplane to all other samples in each group. The resulting optimal hyperplane is intuitively reasonable. The hyperplane is chosen such that it is located at the place where the distance from the hyperplane to the closest samples from the two groups is the largest possible one. Note that the hyperplane derived here is under the assumption that the two groups A and B are separable. Further variants may be needed in order to apply on data sets that are not perfectly separable.

The performance of SVM relies on the projection of data so as to represent patterns in a high dimension. The strength of SVM is that with a suitable nonlinear mapping Φ to a sufficiently high dimension, data from two classes can always be separated by a hyperplane [18, 33]. Kernel functions have been used to perform such mapping. Examples of SVM related kernel functions include linear, polynomial, radial basis function (RBF) and sigmoid. In addition, Shimodaira et al. (2001) [85] gives the Dynamic Time-Alignment Kernel that can deal with time series data. An abstract example is depicted

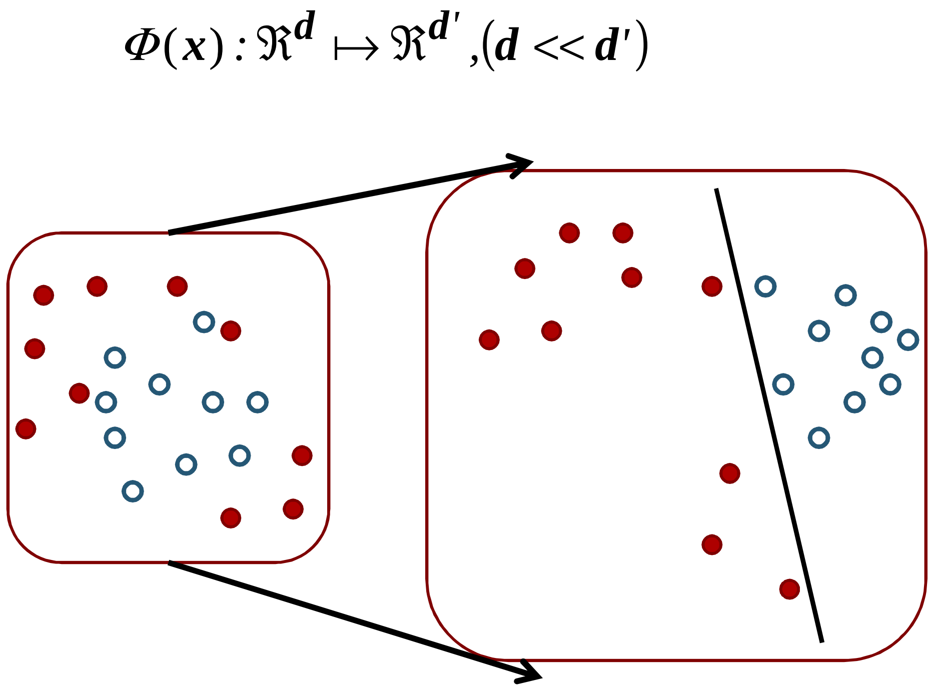


Figure 2.2: Kernel Trick.

in Figure 2.2. Suppose a set of data of two groups is impossible to be separated by any linear function, and it is in a real space of d-dimensional as shown in the left graph. The idea of a kernel function is to have a projection Φ that maps the data points into a comparably very high dimensional space, say d'-dimensional, where $d << d'$. In such a space, a linear separating plane can be constructed more easily. This is resulted by the so called kernel trick. With a kernel trick, the simple linear function can still be used for classifying data in a higher dimensional space. More detailed discussion about kernel functions can be found in Section 2.2.1.

An SVM classifier can be constructed by solving a linear programming problem. One can define a hyperplane with normal $\omega \in \mathbb{R}^d$ and express the plane as

$$x^{\mathsf{T}}\omega = \gamma,$$

where d is total number of features used to represent data cases, and $\gamma \in \mathbb{R}$ is a scalar. The objective is to decide values of (ω,γ), which reach the maximum margin. Denote the set A by the matrix $A \in \mathbb{R}^{m \times d}$ and the set B by the matrix $B \in \mathbb{R}^{k \times d}$, where m and k are the number of cases which belong to groups A and B, respectively. If the

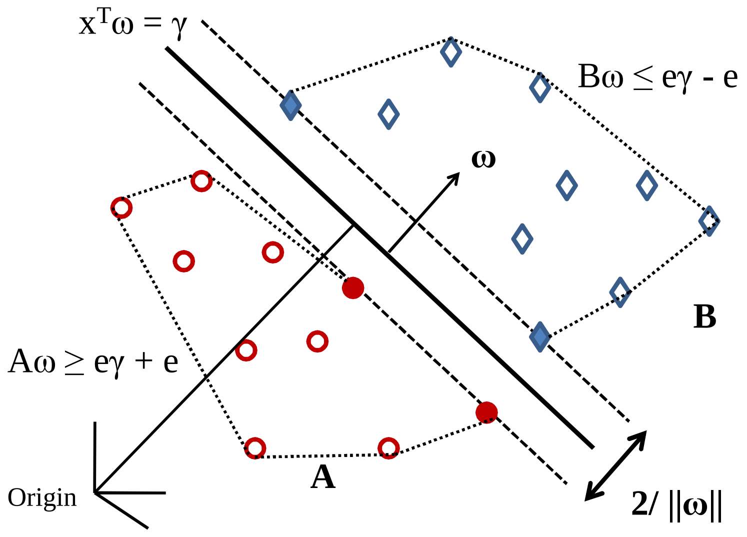


Figure 2.3: Support Vector Machines Classifier

two sets, A and B, are perfectly separable, they separately fall in two open half spaces. The set A lie in $\{x|x \in \Re^n, x^\mathsf{T}\omega < \gamma\}$ and the set B lie in $\{x|x \in \Re^n, x^\mathsf{T}\omega > \gamma\}$. Let e denote a vector of ones with arbitrary dimension. Then the following constraints must be satisfied:

$$A\omega > e\gamma, \quad B\omega < e\gamma.$$

Variables (ω,γ) can be rescaled to obtain non-strict inequalities because strict inequality constraints are not valid in linear programming formulations. To scale them, variables (ω,γ) can be divided by the positive value of $\min_{i=1,\dots,m,j=1,\dots,k}\{A_i\omega - \gamma, -B_j\omega + \gamma\}$.

Without loss of generality, the equivalent inequalities can be written as

$$A\omega \geq e\gamma + e, \quad B\omega \leq e\gamma - e. \tag{2.2}$$

Figure (2.3) shows the two inequalities with the margin, $\frac{2}{\|\omega\|}$.

In practice, most data sets are not perfectly separable. Hence, the assumption of having perfectly separable data sets for SVM is violated, and there exists no solution of (ω,γ) such that the inequalities (2.2) hold. For this reason, one tries to approximate the goal of maximizing margin by minimizing an average sum of violations. This leads to the development of robust linear programming formulation by Bennett and Mangasarian (1992) [6]. The model is given by

$$\begin{aligned} \min_{\omega,\gamma,y,z} \quad & \frac{e^T y}{m} + \frac{e^T z}{k} \\ \text{s.t.} \quad & A\omega - e\gamma - e \geq y, \\ & -B\omega + e\gamma - e \geq z, \\ & y \geq 0, z \geq 0. \end{aligned} \tag{2.3}$$

The variables y and z in the constraints of this problem satisfy the conditions:

$$y \geq \max\{0, -(A\omega - e\gamma - e)\}$$

and

$$z \geq \max\{0, (-B\omega + e\gamma - e)\}.$$

Hence, y and z are vectors containing violations of constraints (2.2). Minimizing the objective function of (2.3) leads to the minimum average violations.

2.1.3 Probability Based Separation

Optimization techniques can include probability concepts. Two techniques are discussed in this section. One is the Likelihood Basis Pursuit (LBP) model, a regression model constructed by nonlinear optimization with a convex objective function and linear constraints. A log likelihood function is incorporated in the model. Another method involves probability density function estimation. It is a multi-group probability-constrained discrimination method [2] aims at constructing an optimal separation $\{R_0, R_1, ..., R_G\}$ while maximizing the probability of correct allocation subject to constraints on the misclassification probabilities under a bound.

2.1.3.1 Likelihood Basis Pursuit (LBP) Model

LBP model is a nonparametric penalized likelihood approach for model building and feature selection [100]. It determines the probabilities of binary outcomes given labeled vectors, while automatically selecting and prioritizing important features. The log likelihood is decomposed into the sum of different functional components, such as main effects and interactions, in the setting of a tensor product reproducing kernel Hilbert space. Each functional component is represented by appropriate basis functions. The basis functions are chosen to be compatible with model building and feature selection in the framework of a smoothing spline analysis of variance (SS-ANOVA) [93].

Bernoulli data is considered in this model, whose probability distribution can be written as

$$p(x) \equiv \text{prob}(Y = 1|X = x) = \frac{e^{f(x)}}{1 + e^{f(x)}},$$

where Y takes on two values $\{0, 1\}$. f is the "logit" function with

$$f(x) = log(\frac{p(x)}{1 - p(x)}).$$

LBP forms a log likelihood as a function of f, and tries to maximize the log likelihood while minimizing the penalized basis pursuit terms. A reproducing kernel Hilbert space is constructed corresponding to an SS-ANOVA decomposition [93]. This way f varies in a high dimensional function space, which leads to a more flexible estimation. The likelihood basis pursuit estimate of f is achieved by minimizing a constructed non-linear convex objective function. After solving for the decomposition of function f, an unlabeled data vector x can be input into $f(x)$ and hence a likelihood is obtained. LBP provides weights on main effects and interactions, and those weights can also be used for feature selection. A sequential Monte Carlo bootstrap tests algorithm [28] has been applied on deciding the best threshold for feature selection [100].

2.1.3.2 Multi-Group Probability-Constrained Discrimination

Optimization techniques can be applied on solving a multi-group probability-constrained discrimination method proposed in [2]. This method considers population densities and prior probabilities when constructing a decision rule, and gives a partition $\{R_0, R_1, ..., R_G\} \subset \mathbb{R}^d$ for a given data set, where d is the number of features. Its result forms a Multi-group classification.

The objective is to construct an optimal separation $\{R_0, R_1, ..., R_G\}$ while maximizing the probability of correct allocation subject to constraints on the misclassification probabilities.

Let f_h, denote the conditional density function of group h, for $h = 1, \ldots, G$. The prior probability that a randomly selected entity is from group g is denoted as π_g, for $g = 1, \ldots, G$. Also, α_{hg}, are constants between 0 and 1 as a bound on misclassification probabilities, for $h \neq g$. The probabilistic classification model is then given by

$$\begin{aligned} \max \quad & \sum_{g=1}^{G} \pi_g \int\limits_{R_g} f_g(w)dw \\ \text{s.t.} \quad & \\ & \int\limits_{R_g} f_h(w)dw \leq \alpha_{hg} \qquad \text{for} \quad h, g = 1, \ldots, G, h \neq g. \end{aligned} \tag{2.4}$$

The optimal rule that can be used as a classification method is stated as

$$R_g = \left\{ x \in \Re^k : L_g(x) = \max_{h \in \{0,1,\ldots,G\}} L_h(x) \right\}, \qquad \text{for } g = 0, \ldots, G, \tag{2.5}$$

where

$$L_0(x) = 0, \tag{2.6}$$

$$L_h(x) = \pi_h f_h(x) - \sum_{i=1, i\neq h}^{G} \lambda_{ih} f_i(x), h = 1, \ldots, G.$$

Anderson [2] showed that there exist nonnegative constants $\lambda_{ih}, i, h \in 1, \ldots, G, i \neq h$, such that this optimal rule (2.5) holds for the probability classification model 2.4. The procedure for deriving a discriminant rule is composed of two stages:

1. The first stage is to compute two kinds of estimated values, $\hat{f}_h$ and $\hat{\pi}_h$. The values of $\hat{f}_h$ are estimated density functions, f_h; and the values of $\hat{\pi}_h$ are estimated prior probabilities, π_h, for $h = 1, \ldots, G$. There are many methods proposed for density estimation [92] which can be applied here.

2. Given the estimates of $\hat{f}'_h s$ and $\hat{\pi}'_h s$, the second stage is to estimate the optimal $\lambda'_{ih} s$. For estimating the $\lambda'_{ih} s$, there is a mixed-integer programming (MIP) approach proposed in [41], and a linear programming (LP) approach proposed in [60].

Experiments using the MIP approach demonstrated its potential to find optimal λ_{ih}'s, but it is not possible to examine large-scale problems due to the difficulty of solving large-scale MIP problems. On the other hand, the LP problems are much easier to solve than the MIP problems.

The candidate sets of $\lambda'_{ih}s$ are calculated as the proportion of training samples, i, that fell into each of the regions, h. The MIP approach uses binary variables to record whether or not each entity was allocated to each region. This approach measures the probabilities of correct classification and misclassification for any candidate set of $\lambda'_{ih}s$. The objective is to maximize a linear combination of variables representing correct allocations. The proportions of misclassified training samples were included in constraints with a bound on misclassification probabilities.

Since the LP approach gets rid of binary variables, it is not possible to incorporate proportions of misclassified training samples. As a result, modeling a priori bounds on misclassification probabilities is not possible. Instead, the LP approach provides a technique for estimating $\lambda'_{ih}s$ that balances the minimization of misclassifications and the maximization of correct classifications. The LP approach especially considers group 0, called the reserved-judgement region as shown in Equation 2.6. It is used for the training samples that can not fit in any suitable group h, $h = 1, \ldots, G$.

2.1.4 Feature Selection

Optimization models designed for classification can also be used for feature selection. Given a data set with information from many attributes (features), it is typical that not all of them are useful for pattern recognition. There may be redundant (not informative) variables that should be discarded in order to classify samples more efficiently. Table 2.2 summarizes those optimization models with brief explanations on how they can be used for feature selection.

Table 2.2: Optimization Models for Classification Used for Feature Selection.

Optimization Models	For Feature Selection
Logical Analysis of Data	The positive ω-patterns that intersect with Ω_S^+ and negative ω-patterns that intersect with Ω_S^- found by LAD optimization models actually provides feature selection.
Support Vector Machines	A hyperplane is expressed as $x^\top\omega = \gamma$ with a normal $\omega \in \mathbb{R}^d$, where d is total number of features used to represent data cases. Thus, elements in the vector ω can be viewed as feature selection.
Likelihood Basis Pursuit Model	LBP provides weights on main effects and interactions of variables. These weights prioritize the importance of features. A sequential Monte Carlo bootstrap tests algorithm has been applied on deciding best features [100].

2.2 Classification Using Data Transformations

2.2.1 Kernel Methods

Kernel methods are widely used in SVM for a better data classification. The purpose of a kernel method is to map data into a higher dimensional space, called *feature space.* In the feature space, many methods can be used to find relations in the data. The main use of a kernel function is to operate those methods in the feature space without ever computing the coordinates of the data in that space. As a result, a kernel function provides computational efficiency.

Let a training data $X = [x_1, x_2, \ldots, x_n]$ and its corresponding class label $Y = [y_1, y_2, \ldots, y_n]$ be given, where $y_i \in \{\pm 1\}$, and $x \in X$ denotes one sample case. Given two sample cases x and x', a similarity measure of x and x' is considered in the form of

$$\begin{aligned} k : X \times X &\rightarrow \mathbb{R}. \\ (x, x') &\mapsto k(x, x'). \end{aligned}$$

The function k is called a *kernel*, which returns a real number describing the similarity of x and x'. Generally k is assumed to be *symmetric*, that is, $k(x, x') = k(x', x)$ for all $x, x' \in X$. One simple similarity measure is a inner product. Let a bold face

$\mathbf{x}$ be used to denote the vectorial representation of x in the feature space. Given two vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^{d'}$, the dot product (also known as inner product) is defined as

$$\langle \mathbf{x}, \mathbf{x}' \rangle := \sum_{j=1}^{d'} [\mathbf{x}]_j [\mathbf{x}']_j,$$

where $[\mathbf{x}]_j$ is the jth element of $\mathbf{x}$ and d is the length of vector $\mathbf{x}$.

Let the space $\mathcal{H}$ denote the dot product space. Then the space $\mathcal{H}$ is called a *feature space*. The patterns in some dot product space $\mathcal{H}$ need to be illustrated in order to use a dot product as a similarity measure. It can be done by a map:

$$\begin{aligned} \Phi : X &\rightarrow \mathcal{H}. \\ x &\mapsto \mathbf{x} := \Phi(x). \end{aligned} \tag{2.7}$$

Note that whether the original patterns exist in a dot product space is unknown. Even so, a more general similarity measure obtained by applying a map (2.7) should still be considered. A similarity measure from the dot product in $\mathcal{H}$ is thus defined by embedding the data into $\mathcal{H}$ via Φ, giving

$$k(x, x') := \langle \mathbf{x}, \mathbf{x}' \rangle = \langle \Phi(x), \Phi(x') \rangle. \tag{2.8}$$

2.2.1.1 Hyperplane Classifiers with Kernel Functions

Consider a hyperplane as a classifier in some dot product space $\mathcal{H}$,

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0, \quad \text{where}\, \mathbf{w} \in \mathcal{H}, b \in \mathbb{R}. \tag{2.9}$$

Then its corresponding decision function is

$$f(\mathbf{x}) = \operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b). \tag{2.10}$$

There exists a unique optimal hyperplane, specified by the maximum margin of separation between any training point and the hyperplane. To construct such hyperplane, a vector perpendicular to the hyperplane that leads to the largest margin need to be found. To solve for $\mathbf{w}$ and b using the training data, one can form the problem:

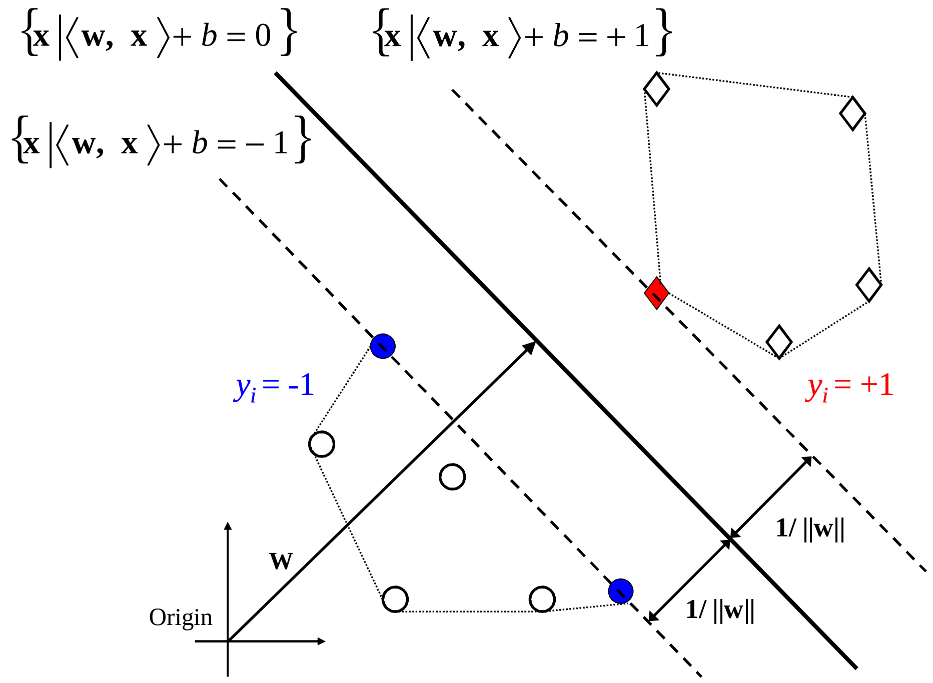

$\{\mathbf{x} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$
$\{\mathbf{x} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = +1\}$
$\{\mathbf{x} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = -1\}$
$y_i = -1$
$y_i = +1$
1/ ||w||
1/ ||w||
w
Origin

Figure 2.4: Hyperplane Classifier

$$\begin{aligned} \min_{\mathbf{w}\in\mathcal{H}, b\in\mathbb{R}} \quad & \tau(\mathbf{w}) = \tfrac{1}{2}||\mathbf{w}||^2 \\ \text{s.t.} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \mathbf{1}, \quad \text{for all } i = 1, \ldots, n. \end{aligned} \tag{2.11}$$

Since $y_i \in \{\pm 1\}$, constraints (2.11) ensure that $f(\mathbf{x}_i) = \langle \mathbf{w}, \mathbf{x}_i \rangle + b$ has the same sign as the corresponding class label y_i. That is $f(\mathbf{x}_i)$ will be $+1$ for $y_i = +1$, and -1 for $y_i = -1$. The distance from $\mathbf{x}_i$ to the hyperplane can be obtained by dividing $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$ by $\|\mathbf{w}\|$. If $||\mathbf{w}||$ were 1, then $|\langle \mathbf{w}, \mathbf{x}_i \rangle + b|$ would equal the distance from $\mathbf{x}_i$ to the hyperplane. Scaling $\mathbf{w}$ and b, (which can be done by dividing by a constant), such that closest points to the hyperplane satisfy $|\langle \mathbf{w}, \mathbf{x}_i \rangle + b| = 1$. Then the margin equals to $1/\|\mathbf{w}\|$ as shown in Figure (2.4). As a result, by satisfying (2.11) for all $i = 1, \ldots, n$ with any $\mathbf{w}$ having minimal length, the overall margin will be maximized.

This optimization problem can be solved by including the constrains in the objective using Lagrange multipliers $\lambda_i \geq 0$ ($\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_n]$). Thus it forms a Lagrangian

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\lambda}) = \frac{1}{2}||\mathbf{w}^2|| - \sum_{i=1}^{n} \lambda_i (y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1). \tag{2.12}$$

The optimal hyperplane can then be found by finding the saddle point of the Lagrangian $\mathcal{L}$. With respect to the *primal variables* $\mathbf{w}$ and b the Lagrangian is minimized, and with respect to the *dual variables* α_i it is maximized. By the Karush-Kuhn-Tucker (KKT) complementarity conditions, at the saddle point, the derivatives of $\mathcal{L}$ with respect to the primal variables $\mathbf{w}$ and b must vanish. That is

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}}\mathcal{L}(\mathbf{w}, b, \boldsymbol{\lambda}) &= \mathbf{w} - \sum_{i=1}^{n} \lambda_i y_i \mathbf{x}_i &= 0. \\ \frac{\partial}{\partial b}\mathcal{L}(\mathbf{w}, b, \boldsymbol{\lambda}) &= -\sum_{i=1}^{n} \lambda_i y_i &= 0. \end{aligned} \tag{2.13}$$

The solution vector thus has an expansion in terms of a set of the training cases.

$$\mathbf{w} = \sum_{i=1}^{n} \lambda_i y_i \mathbf{x}_i. \tag{2.14}$$

$$\sum_{i=1}^{n} \lambda_i y_i = 0. \tag{2.15}$$

There exist the patterns $\mathbf{x}_i$ as a subset of the training cases with non-zero λ_i, called *Support Vectors* (SVs). For those i such that $\lambda_i \neq 0$, Constraints (2.11) have to hold at equalities in order to satisfy the KKT complementarity conditions,

$$\lambda_i[y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1] = 0 \,\text{for all}\, i = 1, \ldots, n. \tag{2.16}$$

This shows that $\langle \mathbf{w}, \mathbf{x}_i \rangle + b = 1$ if $y_i = 1$ and $\langle \mathbf{w}, \mathbf{x}_i \rangle + b = -1$ if $y_i = -1$ for i with $\lambda_i \neq 0$. Thus, the SVs lie on the margin. All the constraints $y_i(\langle \mathbf{w}, \mathbf{x} \rangle + b) \geq 1$ of the remaining training examples (x_i, y_i) with corresponding $\lambda_i = 0$ are irrelevant. They could just as well be left out in the expansion of $\mathbf{w}$ (Equation (2.14)).

In practice, to solve the problem the primal variables $\mathbf{w}$ and b are eliminated by substituting (2.15) and (2.14) into the Lagrangian (2.12), resulting in the so-called *dual optimization problem* [83]:

$$\begin{aligned}
\max_{\boldsymbol{\lambda}\in\mathbb{R}^n} \quad & W(\boldsymbol{\lambda}) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i,j=1}^{n} \lambda_i\lambda_j y_i y_j \langle \mathbf{x_i}\mathbf{x_j}\rangle \\
\text{s.t.} \quad & \lambda_i \geq 0 \quad \text{for all} \quad i = 1,\dots,n \\
& \sum_{i=1}^{n} \lambda_i y_i = 0.
\end{aligned}$$

After obtaining the optimal $\boldsymbol{\lambda}$, $\mathbf{w}$ can be derived by (2.14). Using (2.14), the hyperplane decision function (2.10) can be written as

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{n} y_i\lambda_i\langle \mathbf{x}, \mathbf{x}_i\rangle + b\right).$$

The problem can be written in terms of the input patterns X instead of the of bold face vectors $\mathbf{x}, \mathbf{x}'$ used for the feature space. Recall the kernel expression in Equation (2.8). The decision functions becomes

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{n} y_i\lambda_i\langle \Phi(x), \Phi(x_i)\rangle + b\right) = \operatorname{sgn}\left(\sum_{i=1}^{n} y_i\lambda_i k(x, x_i) + b\right),$$

and the corresponding dual program is:

$$\begin{aligned}
\max_{\boldsymbol{\lambda}\in\mathbb{R}^n} \quad & W(\boldsymbol{\lambda}) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i,j=1}^{n} \lambda_i\lambda_j y_i y_j k(x, x_i) \\
\text{s.t.} \quad & \lambda_i \geq 0 \quad \text{for all} \quad i = 1,\dots,n \\
& \sum_{i=1}^{n} \lambda_i y_i = 0.
\end{aligned}$$

2.2.1.2 Kernel Functions

As described above, the advantage of a kernel function is to define it as $k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, and one would only need to apply the function k in the training phase without ever needing to explicitly know what the function Φ is. Two common examples of kernel functions are polynomial kernels and radial basis functions.

- A **Polynomial kernel** is defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^{\mathsf{T}}\mathbf{x}_j)^p.$$

When $d = 1$, k is a linear kernel. A quadratic kernel is given by taking $d = 2$.

- A **Radial basis function** maps the data into an infinite dimensional Hilbert space. Gaussian kernel is the one most commonly used, which is defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}.$$

  Tuning the value of σ is required to obtain a proper Gaussian kernel. It usually involves brute force search by stepping through a range of values of σ that finds optimal performance of a model with training data.

2.2.2 Dimension Reduction

Consider data points in an attribute data set having a large number of attributes. Projecting these data points in a vector space. Those data points are comparably sparse in a high dimensional vector space made by the large number of attributes, particularly if the attribute values are binary. This sparsity can lower classification accuracy and increase computational complexity [33]. Reducing the dimensionality by combining features can cope with the problem of dimensionality. One method is the Principal Component Analysis (PCA). Suppose a data set is in a d dimensional space (has d features), PCA tries to project the data set onto a space spanned by d' orthogonal directions, where $d' \leq d$. The projection is chosen where the greatest variance of data set is achieved.

PCA is not based on a probability model. The principal directions of the observed data vectors may be determined through maximum likelihood estimation of parameters [66]. A probability can thus be incorporated in PCA. The method is called the probabilistic PCA (PPCA).

In addition, effectiveness of PCA is limited by its linearity. To capture nonlinearities involved in data sets, nonlinear variants of PCA have been shown in the literature. Most popular methods is to integrate neural networks [32], and nonlinear mappings such as kernel functions [84, 79]. There are also combined probabilistic PCA and nonlinear dimension reduction, i.e., Gaussian nonlinear PCA [65] and Bayesian nonlinear PCA [61].

PCA reduces sample space dimensions in order to avoid sparsity problems and possibly to achieve better efficiency and accuracy of classification. However, PCA does not use the class labels when choosing directions in the data that have highest variance, which does not guarantee that it improves discrimination between data in different classes.

Fisher liner discriminant analysis (Fisher-LDA) utilizes the label information in searching projections. Fisher-LDA tries to maximize between-classes variances, and minimize within-classes variances while searching for directions to form a reduced projection space. In practice, linear discriminant may not be sufficient to capture data complexity. Fisher-LDA applied in the feature space can provide a supervised nonlinear feature extraction. This arrives kernel Fisher-LDA [73].

PCA is an unsupervised technique whereas Fisher-LDA is supervised. However, the supervised technique is not always better than unsupervised technique. The performance of Fisher-LDA can be worse than the PCA when applying it on the face recognition data, which is discovered in [9].

2.2.2.1 Principal Component Analysis

Principal component analysis (PCA) is a technique that searches for directions in a sample space which have largest variances. Suppose the sample space is d-dimensional. A new space is spanned by the d' ($d' \leq d$) orthogonal directions that PCA has found. Subsequently it projects the data onto the new space. The projection has the greatest variance of the data lie on the first coordinate, the second greatest variance on the second coordinate, and the d'th greatest variance on the last coordinate.

Consider a training data set of n samples with d dimensions. Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ denote the vectors of all samples. The sample mean is

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i.$$

Then $\bar{\mathbf{x}} \in \mathbb{R}^d$ is a vector which has the smallest distances between all other samples $\mathbf{x}_i$ for all $i = 1, \ldots, n$. One tries to find a one-dimensional space (a line) running through the sample mean ($\bar{\mathbf{x}}$), and has the data projected onto the line. Let $\mathbf{e}$ denote a vector of one's with arbitrary length, and especially be in the direction of the line. Also let a be the distance of any point $\mathbf{x}$ from the mean $\bar{\mathbf{x}} \in \mathbb{R}^d$. Then the line can be expressed as

$$\mathbf{x} = \bar{\mathbf{x}} + a\mathbf{e}.$$

A data vector $\mathbf{x}$ can then be projected onto the line as

$$\mathbf{y} = \mathbf{e}^{\mathsf{T}}\mathbf{x}.$$

There will be unlimited possible lines running through the mean. Let $\mathbf{x}_i = \bar{\mathbf{x}} + a_i\mathbf{e}$. The "optimal" set of coefficients a_i and the direction of the line $\mathbf{e}$ can be found by minimizing the squared-error function

$$J(a_1, \ldots, a_n, \mathbf{e}) = \sum_{i=1}^{n} \|\mathbf{x}_i - (\bar{\mathbf{x}} + a_i\mathbf{e})\|^2.$$

Differentiating J with respect to a_i, setting the derivative to zero, and knowing that $\|\mathbf{e}\| = 1$, one obtains

$$\frac{\partial J}{\partial a_i} = 2\|\mathbf{x}_i - \bar{\mathbf{x}} - a_i\mathbf{e}\|\|\mathbf{e}\| = 0.$$

Hence,

$$a_i = (\mathbf{x}_i - \bar{\mathbf{x}})^{\mathsf{T}}\mathbf{e}. \tag{2.17}$$

Let a scatter matrix S be defined by

$$S = \sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^{\mathsf{T}}. \tag{2.18}$$

By expanding the function J and substituting a_i by the expression from Equation (2.17) into function J, one obtains

$$\begin{aligned} J(\mathbf{e}) &= \sum_{i=1}^{n} \|(\mathbf{x}_i - \bar{\mathbf{x}})^2 - 2(\mathbf{x}_i - \bar{\mathbf{x}})^{\mathsf{T}} a_i \mathbf{e} + (a_i \mathbf{e})^2\| \\ &= \sum_{i=1}^{n} \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2 - \mathbf{e}^{\mathsf{T}} S \mathbf{e}. \end{aligned} \tag{2.19}$$

The last equation applies the fact that

$$\begin{aligned} \sum_{i=1}^{n} a_i^2 &= \textstyle\sum_{i=1}^{n} a_i^2 [(\mathbf{x}_i - \bar{\mathbf{x}})^{\mathsf{T}} \mathbf{e}]^2 \\ &= \textstyle\sum_{i=1}^{n} a_i^2 \mathbf{e}^{\mathsf{T}} ((\mathbf{x}_i - \bar{\mathbf{x}}))((\mathbf{x}_i - \bar{\mathbf{x}}))^{\mathsf{T}} \mathbf{e} \\ &= \mathbf{e}^{\mathsf{T}} S \mathbf{e}. \end{aligned}$$

Minimizing J by choosing $\mathbf{e}$ also maximizes $\mathbf{e}^{\mathsf{T}} S \mathbf{e}$. Observe that the sample variance is $(\sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^{\mathsf{T}})/(n-1)$. This means that the optimal direction of $\mathbf{e}$ is where the largest sample variance is. This forms the maximization problem:

$$\begin{aligned} \max_{\mathbf{e}} \quad & \mathbf{e}^{\mathsf{T}} S \mathbf{e} \\ \text{s.t.} \quad & \|\mathbf{e}\| = 1. \end{aligned}$$

By introducing Lagrange multipliers λ to the problem, the following optimality conditions can be obtained.

$$\mathcal{L} = \mathbf{e}^{\mathsf{T}} S \mathbf{e} - \lambda \mathbf{e}^{\mathsf{T}} \mathbf{e}.$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{e}} = 2S\mathbf{e} - 2\lambda \mathbf{e} = 0.$$

$$S\mathbf{e} = \lambda \mathbf{e}. \tag{2.20}$$

Applying Equation (2.20) to the objective function gives $\mathbf{e}^{\mathsf{T}} S \mathbf{e} = \mathbf{e}^{\mathsf{T}} \lambda \mathbf{e} = \lambda$. Therefore, the optimal λ is the largest eigenvalue of the matrix S and $\mathbf{e}$ is the corresponding eigenvector.

Extending this one-dimensional result, PCA can project data onto a d' dimensional space, where $d' \leq d$. It can be done by expressing the samples as

$$\mathbf{x} = \bar{\mathbf{x}} + \sum_{j=1}^{d'} a_j \mathbf{e}_j.$$

The optimal solution will be the d' eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_{d'}$ of the scatter matrix S in (2.18) having the d' largest eigenvalues. They form the basis of the projected $\mathbf{x}$. The coordinates of the new space spanned by those orthogonal directions correspond to the largest variances of the data set. Thus, the projection can help separate the data cases. Let $E = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{d'}]$, the projected data becomes

$$\mathbf{y} = E^{\mathsf{T}} \mathbf{x}.$$

2.2.2.2 Kernel Principal Component Analysis

Applying PCA in feature space arrives to a method called kernel PCA [84]. By solving an eigenvalue problem, kernel PCA finds features in some sense which represent critical information in a feature space.

Given a training data $X = [x_1, x_2, \ldots, x_n]$. Let a bold face $\mathbf{x}$ be used to denote the vectorial representation of x in the feature space.

Consider a kernel matrix $K_{ij} := k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$.

PCA in feature space $\mathcal{H}$ aims to find eigenvectors $\mathbf{v}$ and eigenvalues λ of the so-called covariance matrix $\mathbf{C}$ in the feature space. The covariance matrix is defined as the expectation of $\mathbf{x}\mathbf{x}^{\mathsf{T}}$, which in terms of the training data X is

$$\mathbf{C} := \frac{1}{m} \sum_{i=1}^{m} \Phi(x_i) \Phi(x_i)^{\mathsf{T}}.$$

The eigenvalue equation for PCA is

$$\mathbf{C}\mathbf{w} = \lambda \mathbf{w} \tag{2.21}$$

for eigenvalues $\lambda \geq 0$ and eigenvectors $\mathbf{w} \in \mathbb{R}^n \setminus \mathbf{0}$.

Usually the function Φ projecting $x \in X$ into a very high dimensional feature space $\mathcal{H}$. This space can be arbitrarily large dimensionality. Therefore, calculating the dot product explicitly results in very expensive computational costs. A helpful method is to use the fact that all solutions $\mathbf{w}$ to Equation (2.21) with $\lambda \neq 0$ must lie in the span of $\Phi(x_1), \ldots, \Phi(x_n)$. That is

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \Phi(x_i).$$

Therefore, the solution $\mathbf{w}$ can be expanded by Φ-images, and the problem becomes to finding the values α, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\mathsf{T}$. That is a dual eigenvalue problem for the expansion coefficients:

$$n\lambda\boldsymbol{\alpha} = K\boldsymbol{\alpha}.$$

Detailed discussion about how to derive this dual problem can be found in [84]. After obtaining values of $\boldsymbol{\alpha}$, kernel PCA can extract nonlinear features from a test point x by computing the dot product between $\Phi(x)$ and the mth normalized eigenvector in feature space. The projection becomes

$$y = \langle \mathbf{v}^m, \Phi(x) \rangle = \sum_{i=1}^{n} \alpha_i^m k(x_i, x).$$

Note that by this equation, $\langle \Phi(x_i), \Phi(x_j) \rangle$ is replaced by $k(x_i, x)$, which is computationally far more efficient than taking the dot product directly in the feature space.

2.2.2.3 Fisher Linear Discriminant Analysis (Fisher-LDA)

Projecting data onto a smaller dimensional space can help avoid problems of sample sparsity in classification caused by excessive dimensionality [33]. If orthogonal directions in a sample space which have largest variances are found, then samples projected on those chosen directions will be feature independent and will have a largest variance, which are easier to be classified. PCA is an unsupervised technique whereas Fisher linear discriminant analysis (Fisher-LDA) (1936) is supervised. Fisher-LDA considers class labels while seeking directions that spans the reduced space, where the projected samples of different classes are well separated.

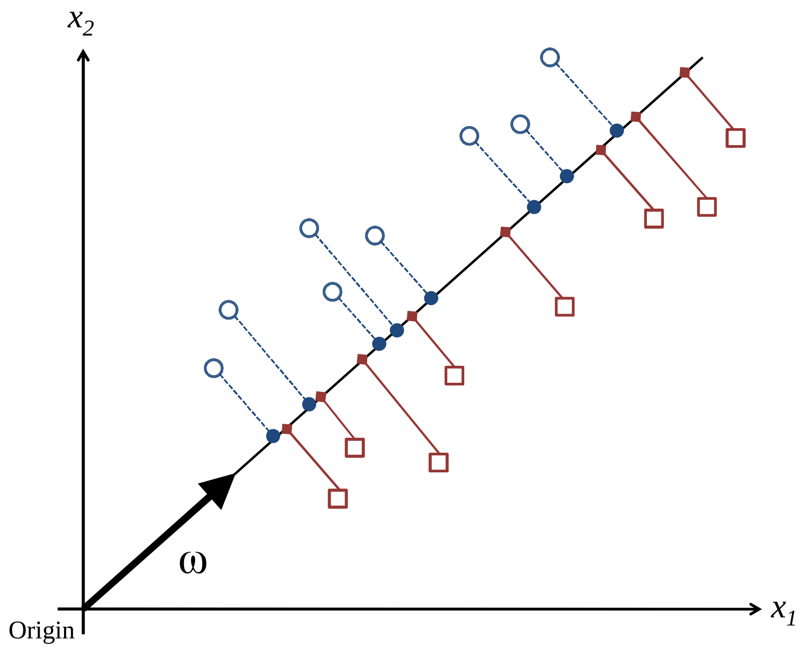


Figure 2.5: Example of PCA that provides unsupervised dimensionality reduction and fails at classification.

In some cases, an unsupervised technique such as PCA may not help for classification but destroy useful information. Consider an extreme example in a two dimensional space as shown in Figure 2.5. The goal is to classify the circles and squares without fillings. PCA finds the direction (ω) that has the largest variance. The solid circles and squares are projected samples on the direction. The resulting locations of the two groups of samples are mixed up and they are more difficult to be classified. This is what happened when applying unsupervised technique. The Fisher-LDA, which is a supervised technique, can solve such problems. In Figure 2.6, the group means are well separated with a smallest sum of possible sample variances within groups.

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be the data set that has n samples in a d-dimensional space. Let the data set have two classes, n_1 of the samples forms the subset C_1 labeled class 1 and n_2 of the samples forms the subset C_2 labeled class 2. $n = n_1 + n_2$. Also Let $\mathbf{W} = [\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_{d'}]$ be the directions that are used for projecting $\mathbf{x}$ into $\mathbf{y}$ in a reduced space, where $d' \leq d$. That is:

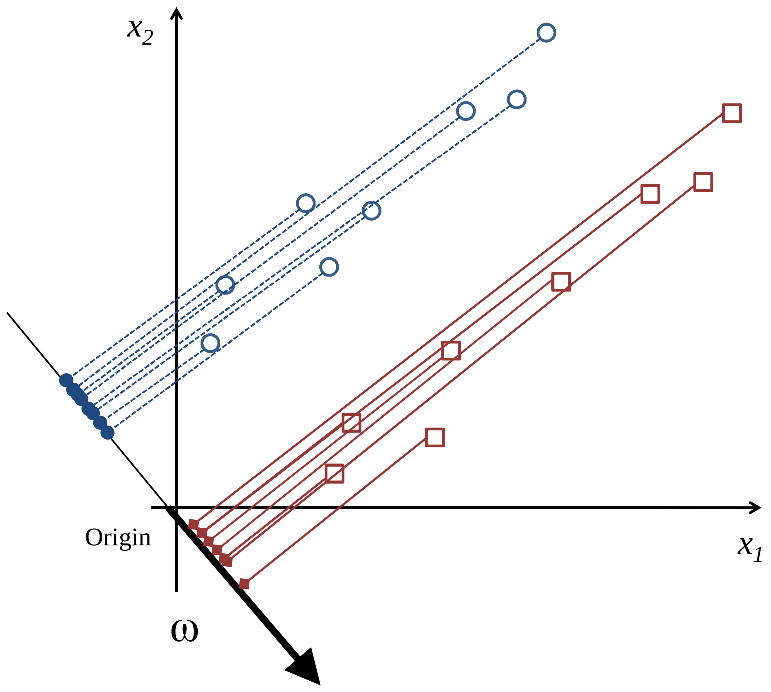


Figure 2.6: Example of Fisher-LDA that provides supervised dimensionality reduction.

$$\mathbf{y} = \mathbf{W}^\mathsf{T}\mathbf{x}.$$

Let $\bar{\mathbf{x}}^k$ be the d-dimensional sample mean of class k, for $k = 1, 2$, where

$$\bar{\mathbf{x}}^k = \frac{1}{n_k} \sum_{x \in C_k} \mathbf{x}$$

Define S_B as the "between-classes scatter matrix" and S_W as the "within-classes scatter matrix". They are:

$$\begin{aligned} S_B &= (\bar{\mathbf{x}}^1 - \bar{\mathbf{x}}^2)(\bar{\mathbf{x}}^1 - \bar{\mathbf{x}}^2)^\mathsf{T} \\ S_W &= \sum_{k=1}^{2} \sum_{x \in C_k} (x - \bar{\mathbf{x}}^k)(x - \bar{\mathbf{x}}^k)^\mathsf{T}. \end{aligned}$$

The objective of Fisher-LDA is to maximize:

$$J(\boldsymbol{\omega}) = \frac{\boldsymbol{\omega}^\mathsf{T} S_B \boldsymbol{\omega}}{\boldsymbol{\omega}^\mathsf{T} S_W \boldsymbol{\omega}}$$

The projected data can be divided into two corresponding subsets $\mathcal{Y}_1$ and $\mathcal{Y}_2$ for class 1 and class 2. Observe that the projected sample variance in its particular class k is $\frac{\sum_{y\in\mathcal{Y}_k}(y-\tilde{\mathbf{y}}^k)^2}{(n_k-1)}$, where $\tilde{\mathbf{y}}^k = \frac{1}{n_k}\sum_{y\in\mathcal{Y}_k} y$, for $k = 1, 2$. It can be shown that the objective tries to separate the class means as much as possible relative to the sum of the variances of the data in their particular classes for projected samples.

Without loss of generality, $\boldsymbol{\omega}$ can always be chosen such that the denominator is $\boldsymbol{\omega}^\mathsf{T} S_W \boldsymbol{\omega} = 1$. The maximization problem becomes a constrained optimization problem:

$$\begin{aligned} \max_{\boldsymbol{\omega}} \quad & -\frac{1}{2}\boldsymbol{\omega}^\mathsf{T} S_B \boldsymbol{\omega} \\ \text{s.t.} \quad & \boldsymbol{\omega}^\mathsf{T} S_W \boldsymbol{\omega} = 1 \end{aligned}$$

It can be solved by maximizing a corresponding Lagrangian problem with a Lagrangian multiplier $\lambda/2$:

$$\mathcal{L} = -\frac{1}{2}\boldsymbol{\omega}^\mathsf{T} S_B \boldsymbol{\omega} + \frac{1}{2}\lambda\boldsymbol{\omega}^\mathsf{T} S_W \boldsymbol{\omega}.$$

By KKT optimality conditions, the following equation has to be satisfied.

$$S_B\boldsymbol{\omega} = \lambda S_W \boldsymbol{\omega} \tag{2.22}$$

$$\Rightarrow \quad S_W^{-1} S_B \boldsymbol{\omega} = \lambda\boldsymbol{\omega}. \tag{2.23}$$

Equation (2.23) obtained by the assumption that S_W is nonsingular and thus invertible. This is an eigenvalue equation. Plugging the Equation (2.22) back into the objective J, one find

$$J(\boldsymbol{\omega}) = \frac{\boldsymbol{\omega}^\mathsf{T} S_B \boldsymbol{\omega}}{\boldsymbol{\omega}^\mathsf{T} S_W \boldsymbol{\omega}} = \lambda \frac{\boldsymbol{\omega}^\mathsf{T} S_W \boldsymbol{\omega}}{\boldsymbol{\omega}^\mathsf{T} S_W \boldsymbol{\omega}} = \lambda$$

Therefore, the largest eigenvalue of $S_W^{-1} S_B$ maximizes the objective J, and the optimal $\boldsymbol{\omega}$ is the corresponding eigenvector. The projection space of dimension d' ($d' \leq d$) can be constructed by spanning the generalized eigenvectors with the largest d' eigenvalues to provide $\mathbf{W} = [\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_{d'}]$.

2.2.2.4 Kernel Fisher-LDA

Fisher-LDA can be applied in feature space which leads to a nonlinear classification technique [73]. Let a training data $X = [x_1, x_2, \ldots, x_n]$ be given, which can be divided into 2 groups: $X_1 = [x_1^1, x_2^1, \ldots, x_{n_1}^1]$ of class 1, and $X_2 = [x_1^2, x_2^2, \ldots, x_{n_2}^2]$ of class 2. Let a bold face $\mathbf{x}$ be used to denote the vectorial representation of x in the feature space. Also, let Φ be a nonlinear mapping projecting $x \in X$ to a feature space $\mathcal{H}$:

$$\begin{aligned} \Phi : X \quad &\rightarrow \mathcal{H} \\ x \quad &\mapsto \mathbf{x} := \Phi(x). \end{aligned} \tag{2.24}$$

Let the class mean in the feature space be defined as

$$\bar{\mathbf{x}}^l := \frac{1}{n_l} \sum_{i=1}^{n_l} \Phi(x_i^l) \quad \text{for all} \quad l = 1, 2. \tag{2.25}$$

Then the between-class scatter matrix S_B^Φ and the within-class scatter matrix S_W^Φ in the feature space can be defined as

$$\begin{aligned} S_B^\Phi &= (\bar{\mathbf{x}}^1 - \bar{\mathbf{x}}^2)(\bar{\mathbf{x}}^1 - \bar{\mathbf{x}}^2)^\mathsf{T} \\ S_W^\Phi &= \sum_{k=1}^{2} \sum_{x \in X_k} (\Phi(x) - \bar{\mathbf{x}}^k)(x - \bar{\mathbf{x}}^k)^\mathsf{T}. \end{aligned}$$

The objective of Fisher-LDA in the feature space $\mathcal{H}$ is to maximize:

$$J(\boldsymbol{\omega}) = \frac{\boldsymbol{\omega}^\mathsf{T} S_B^\Phi \boldsymbol{\omega}}{\boldsymbol{\omega}^\mathsf{T} S_W^\Phi \boldsymbol{\omega}}. \tag{2.26}$$

By the theory of reproducing kernels, any solution of $\boldsymbol{\omega} \in \mathcal{H}$ must be in the span of $[\Phi(x_1), \Phi(x_2), \ldots, \Phi(x_n)]$, which are all training samples in $\mathcal{H}$. This gives an expansion for $\boldsymbol{\omega}$:

$$\boldsymbol{\omega} = \sum_{i=1}^{n} \alpha_i \Phi(x_i). \tag{2.27}$$

Plugging Equation (2.27) into the definition of $\bar{\mathbf{x}}^l$ in Equation (2.25) and combining it with the objective (2.26), one can form a new objective which is used in practice as to maximize

$$J(\alpha) = \frac{\boldsymbol{\alpha}^\mathsf{T}\mathbf{M}\boldsymbol{\alpha}}{\boldsymbol{\alpha}^\mathsf{T}\mathbf{N}\boldsymbol{\alpha}}. \tag{2.28}$$

where

$$\begin{aligned}
(\mathbf{M}^l)_i &:= \frac{1}{n_l}\sum_{j=1}^{n_l} k(x_i, x_j^l) \text{ for all } i = 1, \ldots, n, \text{ and } \ l = 1, 2, \\
\mathbf{M} &:= (\mathbf{M}^1 - \mathbf{M}^2)(\mathbf{M}^1 - \mathbf{M}^2)^\mathsf{T}, \text{ and} \\
\mathbf{N} &:= \sum l = 1^2 K_l(I - \mathbf{1}_{n_l})K_l^\mathsf{T}.
\end{aligned}$$

Here, K_l is a $n \times n_l$ matrix with $(K_l)ij := k(x_i, x_j^l)$ for each class $l = 1, 2$. I is the identity matrix and $\mathbf{1}_{n_l}$ is a matrix with all elements $1/n_l$. More detail about how to derive the objective (2.28) can be found in [73].

The problem becomes to find the eigenvectors $\boldsymbol{\omega}$ and eigenvalues $\boldsymbol{\alpha}$ of $\mathbf{N}^{-1}\mathbf{M}$. The kernel Fisher LDA gives the projection of the form

$$\langle w^\mathsf{T}\Phi(x)\rangle = \sum_{i=1}^{n} \alpha_i K(x_i, x).$$

2.3 Clustering Algorithms

Unsupervised learning tries to find unknown groups in a collection of samples without considering any class labels. Data clustering is one such method that groups “similar” observations together without knowing their class labels. Grouping similar instances together can speed up the labeling process for a large, unlabeled data set. It may also be helpful for gaining some insight into the structure of the data. In addition, feature selection in clustering may help reduce the size of a data set and improve the performance of a classifier. Moreover, similarity measures are especially useful for analyzing time series types of data. Clustering techniques that come with such similarity measures can hence be naturally applied on time series.

Throughout this section, all vectors are column vectors. Suppose a data set contains n samples, and each has m features. Let $i, j \in \{1, \ldots, n\}$, $s \in \{1, \ldots, m\}$, $p \in \{1, \ldots, k\}$. If A is a matrix, A_i is a vector which denotes the ith row of A and A_{ij} is a value of the ith row and jth column of A. Let G_p denote the set of indices of samples that are assigned to cluster p for $p = 1, \dots, k$. That is $G_p \subset \{1, 2, \dots, n\}$ and $G_p \cap G_{p'} = \emptyset$ for joint clustered subsets $p \neq p'$ for p and $p' \in \{1, \dots, k\}$. $G_1 \cup G_2 \cup \cdots \cup G_k = \{1, 2, \dots, n\}$.

2.3.1 Nearest Neighbor Clustering

A simple clustering algorithm using the nearest neighbor rule is proposed by Lu and Fu (1978) [64]. The algorithm is used to cluster patterns of sentences as an application of syntactic pattern recognition.

A set of samples $X = \{x_1, x_2, \dots, x_n\}$ is to be divided into k clusters. A threshold value, t, on the nearest-neighbor distance should be specified by the user. Then the number of clusters k will automatically be decided by the algorithm. That is the k number of clusters generated is a function of t. Fewer clusters are generated as the value of the threshold t increases. Two samples should be in the same cluster if they share neighbors. The description of the Nearest Neighbor Clustering is shown in Algorithm 2.3.1.

Algorithm 2.3.1 The Nearest Neighbor Clustering Algorithm

```
Input: Set of sample ($X = \{x_1, x_2, \dots, x_n\}$), a distance threshold ($t$).
1: Assign sample $x_1$ to cluster $C_1$. ($i = 1$ and $k = 1$)
2: for $i = 2$ to $n$ do
3:    Find the nearest neighbor of $x_i$ among the samples that are assigned to clusters.
4:    If cluster $m \in \{1, \dots, k\}$ contains the nearest neighbor, let $d_m$ be the distance
      from $x_i$ to its nearest neighbor in cluster $m$.
5:    if $d_m < t$ then
6:       Assign sample $x_i$ to $C_m$.
7:    else
8:       $k = k + 1$
9:       Assign sample $x_i$ to a new cluster $C_k$.
10:   end if
11: end for
Output: Clusters ($C = \{C_1, \dots, C_K\}$).
```

A variation of the Algorithm 2.3.1 is to consider the average distance between x_i and its q nearest neighbors. Then q is another user specified parameter.

2.3.2 QT Clustering

The quality cluster (QT clustering) algorithm is described in Algorithm 2.3.2. A distance threshold d is given by the user. The algorithm tries to calculate the number of samples within the diameter d for each sample. Then determine a cluster by the samples that have the largest number of neighbors within that diameter threshold. QT clustering is good for cluster a data set whose number of classes is unknown. The algorithm will automatically generate the optimal number of classes in the data set. It incorporates a pre-calculated distance matrix in its model, and thus provides the flexibility of choosing a distance measure.

Algorithm 2.3.2 The QT Clustering Algorithm [45]

```
Input: a set($G$), and a diameter threshold ($d$).
1:  Procedure QT_Clust($G$, $d$)
2:  if $|G| \leq 1$ then
3:      output $G$ /* Last case */
4:  else
5:      for $i \in G$ do
6:          set $flag = TRUE$
7:          set $A_i = i$ /* $A_i$ is a cluster began by $i$*/
8:          while ($flag = TRUE$) and ($A_i \neq G$) do
9:              find $j \in (G - A_i)$ such that $diameter(A_i \cup \{j\})$ is minimum
10:             if $diameter(A_i \cup \{j\}) > d$ then
11:                 set $flag = FALSE$
12:             else
13:                 set $A_i = A_i \cup \{j\}$ /* Add $j$ to cluster $A_i$ */
14:             end if
15:         end while
16:     end for
17:     identify set $C \in \{A_1, A_2, \ldots, A_{|G|}\}$ with the maximum cardinality
18:     output C
19:     call QT_Clust($G - C$, $d$)
20: end if
Output: a set of clusters.
```

2.4 Clustering with Optimization Models

Optimization models have been applied to help grouping similar samples together in clustering techniques. The original k-median problem is formulated as a concave minimization problem in [15]. A nonlinear multi-objective optimization is used for solving

k-means problem in [13]. Moreover, the sample-preserved k-median (SPKM) clustering method is formulated in integer programming formulation.

2.4.1 k-Median Clustering via Concave Minimization

Given a data set, the matrix $A \in \mathbb{R}^{n \times m}$ is used to represent all the data points where each sample is denoted as $A_i \in \mathbb{R}^m$ for $i = 1, \ldots, n$. To determine k cluster centers in the m-dimensional real space $\mathbb{R}^m$, the goal of the k-median algorithm is to assign each of the n sample points to one of the k clusters, in a way that the sum of distances of each point to the nearest cluster center is minimized.

Bradley (1997) [15] proposed an optimization model, as shown in Program (2.29), that is used to find the k desired clusters. The decision variables are the cluster centers $C_p, p = 1, \ldots, k$ in $\mathbb{R}^m$. The distance matrix $D_{ip} \in \mathbb{R}^m$ is a dummy variable that represents the bounds on the elements of the difference between point $A_i^\top$ and cluster center C_p. The objective is to minimize the 1-norm distance between $A_i^\top$ and C_p.

$$\begin{aligned} \min_{C,D} \quad & \sum_{i=1}^{n} \min_{p=1,\ldots,k} \{\sum_{s=1}^{m} D_{ips}\} \\ \text{s.t.} \quad & \\ & A_i^\top - C_p \leq D_{ip} \qquad \forall\, i, p \\ & A_i^\top - C_p \geq -D_{ip} \qquad \forall\, i, p. \end{aligned} \tag{2.29}$$

By Lemma 2.1 in [7] (Lemma 2.4.1), Program (2.29) can be formulated as a different optimization problem with a nonlinear objective function and linear constraints as shown in Program (2.33).

Lemma 2.4.1. *Given $d, t \in \mathbb{R}^k$, then*

$$\min_{p=1,\ldots,k} \{d_p\} \tag{2.30}$$

is equal to

$$\min_t \left\{ \sum_{p=1}^{k} d_p t_p | \sum_{p=1}^{k} t_p = 1, t_p \geq 0, \text{for} \quad p = 1, \ldots, k \right\}. \tag{2.31}$$

Proof. The dual of the linear program (2.31) is

$$\max_{u \in \mathbb{R}} \{u | u \leq d_p, \text{for} \quad p = 1, \ldots, k\}. \tag{2.32}$$

Observe that the maximum (2.32) is $u = \min\limits_{p=1,\ldots,k} \{d_p\}$, which is equal to (2.30). By the duality theory of linear programming, the maximum of the dual linear program (2.32) equals to the minimum of its primal linear program (2.31). This proves the equivalence. $\square$

$$\begin{array}{lll}
\min\limits_{C,D,T} & \sum\limits_{i=1}^{n}\sum\limits_{p=1}^{k}\sum\limits_{s=1}^{m} D_{ips}T_{ip} & \\
\text{s.t.} & & \\
& A_i^{\mathsf{T}} - C_p \leq D_{ip} & \forall\ i, p \\
& A_i^{\mathsf{T}} - C_p \geq -D_{ip} & \forall\ i, p \\
& \sum\limits_{p=1}^{k} T_{ip} = 1 & \forall\ i \\
& T_{ip} \geq 0 & \forall\ i, p.
\end{array} \tag{2.33}$$

The concept of Uncoupled Bilinear Program Algorithm (UBPA) [7] can be applied to solve this nonlinear optimization problem. This algorithm alternatively solves a linear program in the variable T and a linear program in the variables C and D. The algorithm will converge to a local optimal solution in a finite number of iterations.

Observe that the optimization model for k-median clustering in Program (2.29) applies efficiently only to the 1-norm distance measure. If the 2-norm is applied, the problem results in many local minima and becomes a considerably harder problem [15].

2.4.2 k-Means Clustering

The k-means clustering algorithm was introduced by J. MacQueen in (1967) [67]. The algorithm tries to find k cluster centers, $C_1, \ldots, C_k$ in $\mathbb{R}^m$, such that the choice of centers minimizes the sum of all the squared 2-norm distances from each sample A_i^{T} to its nearest cluster center C_p. The k-means clustering algorithm is displayed in Algorithm 2.4.1.

Algorithm 2.4.1 The k-Means Clustering Algorithm

Input: k = number of clusters, and an initial k samples, $C_1, \ldots, C_k$ in $\mathbb{R}^m$.

Step 1.

- Take the k samples, $C_1, \ldots, C_k$, as single-element clusters.
- Assign each of the remaining $(n - k)$ samples to the nearest cluster centroid.
- After each assignment, compute and update the centroid of the gaining cluster.

Step 2.

- Sequentially take each sample and compute its distance from the centroid of each of the clusters.
- If a sample is not in the cluster with the closest centroid, reassign the sample to the closest cluster.
- Update the centroid of both the cluster losing the sample and the cluster gaining it.

Step 3. Repeat Step 2 until no new assignments can be made.

Output: A set of cluster centers, $C_1, \ldots, C_k$.

A mathematical programming formulation for the k-means clustering can be written as Problem (2.34) as shown in Bradley (2000) [13].

$$\min_{C} \sum_{i=1}^{n} \min_{p=1,\ldots,k} \left(\frac{1}{2}\|A_i^{\mathsf{T}} - C_p\|_2^2\right). \tag{2.34}$$

2.4.3 Sample-Preserved k-Median (SPKM) Clustering in Integer Programming Formulation

The sample-preserved k-median clustering method tries to find k medians, each of which is one of the samples in the k clusters. That is, a median of the sample-preserved method is a sample whose sum of distances to all other samples in the same cluster is the smallest possible. The sample-preserved k-median clustering method is derived from the facility location (p-location) problem, which is an integer programming problem [24].

Two sets of binary decision variables are needed to formulate the problem. The first set is y_i for $i = 1, \ldots, n$, which indicates whether sample i is selected as a median.

The second set is Z_{ij} for $i, j = 1, \dots, n$, which indicates whether sample j is assigned to be the cluster which has sample i as its median. A distance matrix D is included in the objective function where the element D_{ij} represents the distance from sample j to the median i for $i, j = 1, \dots, n$. Note that by this definition, D is a symmetric matrix with a zero value in all of its diagonal elements. The objective is to minimize the sum of distances from all samples to their cluster medians. The formulation for the SPKM clustering method can be written as the following Program (2.35).

$$\begin{aligned}
\min_{Z,y} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} D_{ij}Z_{ij} \\
\text{s.t.} \quad & \\
& \sum_{i=1}^{n} Z_{ij} = 1 \qquad \text{for} \quad j = 1, \dots, n \\
& Z_{ij} \le y_i \qquad \text{for} \quad i, j = 1, \dots, n \\
& \sum_{i=1}^{n} y_i \le k \\
& Z_{ij} \in \{0,1\} \qquad i, j = 1, \dots, n \\
& y_i \in \{0,1\} \qquad i = 1, \dots, n.
\end{aligned} \tag{2.35}$$

The integer programming formulation of the SPKM clustering in Program (2.35) can adapt various distance measures, because a precalculated distance matrix is incorporated. However, solving the integer programming problem is $\mathcal{NP}$-hard. Therefore, approximation algorithms for the k-median problem arise and can be found in many literatures. Studies such as the one in [24] provide approximation algorithms that involve linear relaxation of the integer programming problem in order to solve the problem easily and efficiently. Greedy local-search based solutions can be found in [23, 38]. There are also techniques based on a linear programming relaxation proposed in [68], a hierarchically greedy approach proposed in [72], and an approximation derived from a primal-dual-based algorithm with the use of Lagrangian relaxation proposed in [49].

Chapter 3

TIME SERIES ANALYSIS AND CLASSIFICATION

In this chapter, time series analysis techniques that are commonly used and time series classification approaches developed recently are introduced. The most important time series analysis technique is the time series similarity measure. Measures used for estimating similarity of both univariate time series and MTS are addressed. Based on a suitable similarity measure, nearest neighbor rule is an intuitive classification method that applies similarities as distances in a space. In the literature, there are techniques developed for improving the nearest neighbor rule. Studies related to the nearest neighbor rule approaches as well as feature extraction techniques for time series classification are included.

In general, time series related research studies can be divided into the following three prospects.

1. **Value Prediction.** Time series value prediction is the use of a model to forecast the next successive values based on known past values. This is commonly used for economic situations and control problems. Autoregressive integrated moving average (ARIMA) model is designed especially for such time series value prediction [12].

2. **Event Prediction.** Event prediction in time series is to predict the timing of upcoming events based on a history of past observations. It is different from value prediction in the sense that event prediction requires numerical features and provides patterns of a specific "event" within a segment of time. In real world applications, it involves predicting fraudulent credit card purchases, system failures, and customer churn events [75], and so on. An example is a stock market data set that includes closing prices for the stock for each day. The main interest

of event prediction is to recognize the approximate timing of future events, i.e., the points at which the stock will change the direction of its slope [42].

3. **Time Series Classification.** Time series classification tries to separate time series samples into groups. The classifier is trained based on labeled time series data, whose sample classes are known. Then it is used to classify unlabeled time series data, whose sample classes are unknown.

The focus of this dissertation in time series is on the last category: time series classification. In this chapter, analysis techniques, which can help contribute further discrimination of time series, and time series classification methods that have been successfully developed, are discussed.

Measuring similarities of time series data is essential for time series classification. Methods that can be used to estimate similarities for both univariate time series and multivariate time series (MTS) are introduced in this chapter. For single time series, there are Euclidean distance, T-statical distance (T-statistics) and dynamic time warping (DTW), etc. However, when time series has multi-attributes, in order to capture the relationships among these attributes, similarity measures for MTS without breaking up the attributes may be needed. Similarity measures developed in [51], called Kullback-Leibler discrimination information and the Chernoff information are for discrimination between multivariate time series in the multivariate non-Gaussian case.

Non-parametric analysis such as Principal Component Analysis (PCA) is unique and independent of any hypothesis about data probability distribution. PCA has been used as a dimension reduction and classification technique for attribute data, yet it can not be used to directly classify time series data. Still PCA can be used for analyzing time series in two aspects. One is to apply PCA results to measure similarities of multivariate time series [58, 86, 97]. The other is to use PCA as a tool for time series feature extraction [87].

Most intuitive technique for classifying time series is to apply the k-nearest neighbor (KNN) rule. It may be seen as a brute force technique, but it catches temporal properties of time series and has been shown to be a powerful algorithm for classifying time series. Improved KNN techniques are also included in this chapter.

There are many successful classification techniques designed especially for attribute data sets. However, they can not be directly applied on time series data sets. To be able to use these methods on time series data sets, one has to express time series data in a vector space so that they can be treated as attribute data. One can apply dynamic time warping as a distance measure and embedding time series in a vector space [44]. A parameter space is constructed by a meta-feature technique [50], where recurrent substructures of instances are discovered. The classification is then performed on the parameter space, which is also a vector space. Another technique proposed in [99] vectorizes components of a correlation coefficient matrix of MTS. Those vectorized components are then used as input features of SVM for classification.

3.1 Time Series Similarity Measures

The choice of similarity measures is very important in achieving accurate time series classification. Two types of measures can be considered. One is for single time series and the other is for multivariate time series (MTS). The measures used for single time series can be applied on MTS samples as well. It can be done by breaking each MTS sample into single time series, calculating similarity at each attribute, and then combining all measures together as shown in Figure 3.1. This method may lose the correlation information among MTS attributes, since each attribute is measured separately. Instead of breaking MTS samples, PCA-based similarity measures, such as PCA similarity factor [58] and extended Frobenius norm [97], consider all attributes in one calculation. Hence, the correlation among attributes is preserved.

3.1.1 Similarity between Two Single Time Series Samples

Single time series data have values of a fixed attribute, but vary with time. It can be viewed as a vector whose elements represent values at different time points. Euclidean distance measures distance of two points in a vector space, which is a common method used for comparing similarity. If two time series are not aligned one to one with time,

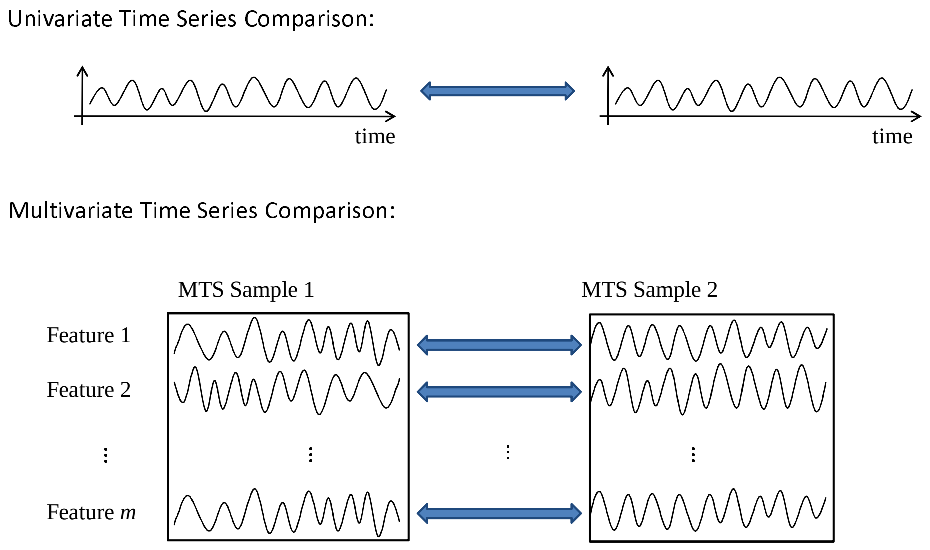

Univariate Time Series Comparison:
time
time
Multivariate Time Series Comparison:
MTS Sample 1
MTS Sample 2
Feature 1
Feature 2
Feature m

Figure 3.1: Decompose multivariate time series into multiple univariate time series.

Euclidean distance may fail to compare. In such case, dynamic time warping (DTW) can help align the two, and only the warping window restricts allowable warping paths. T-statistics is emphasized on the difference between two time series at each time point, which is composed of the average and the variance of value differences.

3.1.1.1 Euclidean Distance

The Euclidean distance is the most commonly used similarity measure. It is an average of the point-to-point differences of two time series. The Euclidean distance ($EU(x, y)$) between the two time series $x = x_1, \ldots, x_L$ and $y = y_1, \ldots, y_L$ is defined as:

$$EU(x, y) := (\sum_{i=1}^{L}(x_i - y_i)^2)^{\frac{1}{2}}.$$

Euclidean distance in a vector space is also called Euclidean norm, denoted as $\|x-y\|_2$. There are also other similar measurements, such as p-norm:

$$\|x - y\|_p := (\sum_{i=1}^{L}(x_i - y_i)^p)^{\frac{1}{p}}.$$

To be a valid norm, p has to be greater or equal to 1. If $0 < p < 1$, the resulting function will not define a norm. There is also a zero norm, which is not a true norm, and is known as the Hamming distance in the case of 2-element finite field. Only when

$p \geq 1$, p-norm is a valid distance measure. If p equals to 2, it is Euclidean norm. If p equals to 1, it is a sum of absolute values of each elements:

$$\|x - y\|_1 := \sum_{i=1}^{L} |x_i - y_i|.$$

A maximum norm is defined as:

$$\|x - y\|_\infty := \max_i \, (x_i - y_i).$$

3.1.1.2 Dynamic Time Warping (DTW) Distance

Given two time series (or vector sequences) x and y with lengths $|x| = L$ and $|y| = K$, the similarity of DTW is similar to the Euclidean distance, except that DTW considers dynamic time points. Let s denote a time point in time series x, and t in time series y. In the calculation of Euclidean distance, where $L = K$, only at time points $s = t$, values of x_s and y_t is compared, for $s = 1, \dots, L$ and $t = 1, \dots, K$. However, in DTW, at any time points (s, t), the next possible comparison can be the time points (s', t') where $s \leq s' \leq s + 1$ and $t \leq t' \leq t + 1$.

The Euclidean distance is used to measure the local distance between two vectors. $d(x_s, y_t) = (x_s - y_t)^2$ is the distance between the s^{th} point of time series x, and the t^{th} point of time series y. Subsequently, a warp path can be constructed, starting at the beginning of each time series $(1, 1)$, and finishing at the end of both time series (L, L) (shown in Figure 3.2).

Note that there can be an exponential number of warping paths that satisfy the above conditions. However, the optimal warp path is the one with a minimum warping (distortion) cost.

$$DTW(s,t) = d(x_s, y_t) + \min \begin{cases} DTW(s, t-1) \\ DTW(s-1, t) \\ DTW(s-1, t-1). \end{cases}$$

The DTW distance can be obtained by a dynamic programming approach. That is to calculate the warp path in a reverse order starting at the end of both time series.

The optimal DTW distance is $DTW(L, K)$.

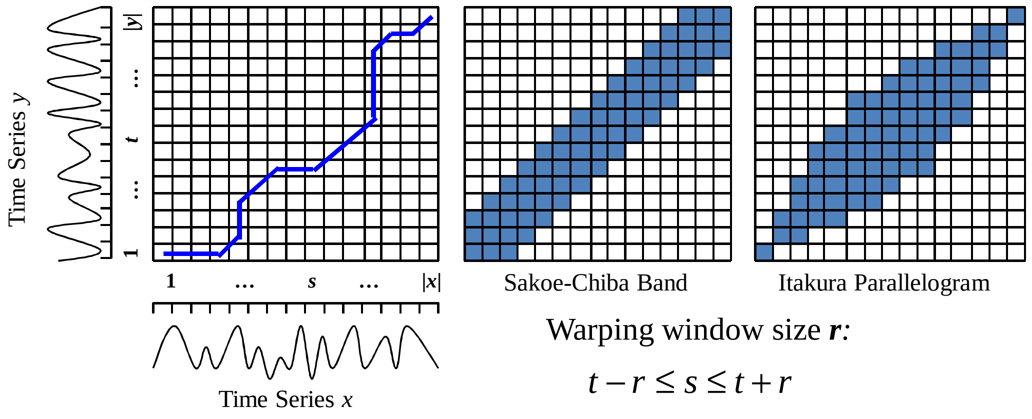


Figure 3.2: An example of warping matrix with a warp path of two time series X and Y. Two methods restricting warping paths are Sakoe-Chiba band and Itakura parallelogram.

The available cells to be evaluated for searching a warping path can be restricted by a warping window constraint. When constraints are used, the DTW algorithm finds the optimal warp path only through the warping window. Since less cells are required to be evaluated, the computation of DTW can be speed up. Also, the evaluation of the similarity can be more accurate as each step of warping is avoided jumping too far. The two mostly common used constraints are shown in Figure 3.2. The shaded areas in Figure 3.2 are called warping window, which are the cells of the cost matrix that are filled in by the DTW algorithm for each constraint. The width of each shaded area is specified by a parameter, r. Sakoe-Chiba band [82] uses a constant warping window, while Itakura parallelogram [46] linearly increases the warping window and then linearly decreases it. That is r is a linear function of warping indices.

3.1.1.3 T-Statistics

The t-statistics is a statistical analysis used to examine if two time series statistically different from each other. In particular, it assesses whether the means of two time series are statistically different from each other. The t-statistics ($T_{x,y}$) between the two time series $x = x_1, \ldots, x_L$ and $y = y_1, \ldots, y_L$ is then defined as:

$$T(x,y) = \frac{\frac{1}{L}\sum\limits_{i=1}^{L}(x_i - y_i)}{\sigma_{xy}\sqrt{L}},$$

where

$$\sigma_{xy} = \sqrt{\frac{\sum\limits_{i=1}^{L}\left[(x_i - y_i) - \frac{1}{L}\sum\limits_{i=1}^{L}(x_i - y_i)\right]^2}{L-1}}$$

is the standard deviation of the differences.

3.1.2 Similarity between Two MTS Samples

Similarity measures designed especially for MTS samples are discussed in this section. These measures do not break the MTS into separate single time series, but calculates the similarity of MTS samples both spatially and temporally at a time. Hence, they include the correlations among multiple attributes.

3.1.2.1 PCA Similarity Factor

PCA similarity factor is a similarity measure for two MTS data proposed in [58]. A MTS item can be viewed as a matrix, which contains elements with two characteristics: features and time. Hence, techniques that measure similarity of two matrices can be applied to calculate similarity of two MTS items.

First the correlation coefficient matrices of the two MTS data are obtained, and then are decomposed via singular value decomposition to derive the principal components. Consequently, the similarity of the corresponding principal components from the two MTS data are measured [86]. In the last step, Frobenius norm can be used as similarity measure between two matrices. The Frobenius norm between two eigenvector matrices sums up the cosine values of angles between the corresponding eigenvectors.

Definition 3.1.1. The PCA similarity factor (s) between two matrices, X and Y is defined as:

$$s(X,Y) = \mathrm{trace}(AB^{\mathsf{T}}AB^{\mathsf{T}}) = \sum_{i=1}^{k}\sum_{j=1}^{k}\cos^2\theta_{ij},$$

where A and B are the matrices consisting of the first k principal components of X and Y, respectively; and θ_{ij} denotes the angle between the ith principal component of X and the jth principal component of Y.

To decide the value k is usually based on heuristics, i.e. choosing the first k principal components whose variances reach 95% of total variance. The range of the PCA similarity factor, s, is from 0 to k.

3.1.2.2 Extended Frobenius Norm

Using both eigenvectors and eigenvalues of covariance matrices, a similarity measure *Eros* (Extended Frobenius norm) is proposed in [97] for k-nearest neighbor decision in multivariate time series data sets. *Eros* extends the Frobenius norm to obtain similarity for two matrices using the results of principal component analysis. The process of *Eros* to measure similarity of two multivariate time series (MTS) matrices is as follows.

1. Compute covariance matrices of the two MTS items.
2. Calculate eigenvectors and eigenvalues of the covariance matrices.
3. Measure similarities between eigenvectors (principal components) of two MTS items with weights, as which the corresponding eigenvalues obtained from the MTS items are taken.

This method does not break the MTS into separate single time series. Instead, *Eros* takes the MTS items as a whole, which includes the correlations between features, and thus results in a more meaningful similarity measure.

3.2 Time Series Classification

The most intuitive method to classify time series is to apply k-nearest neighbor (KNN) rule. However, since KNN rule is a brute force method, problems of computation efficiency have been studied. In addition, the success of KNN also depends on its training samples. Techniques used to edit training samples and to handle unbalanced data sets are also appeared in the literature.

SVM can not classify time series directly. However, properties of time series can be extracted and be viewed as attribute data for SVM to be able to classify them. In addition, dynamic time alignment kernel in SVM projects single time series in a different space in order to find a separating hyperplane. Furthermore, one can apply PCA to extract features of time series for analysis.

3.2.1 k-Nearest Neighbor (KNN) Rule

In order to identify a querying sample, intuitively the similarity of the querying sample within the training samples is considered. By observing the similarity, the class of the majority a querying sample is most similar to is what the querying sample most likely belongs to. Especially taking distance measures as the similarity measure, the classes of its nearest neighbors determine its class label. This is the KNN rule.

KNN represents a bridge between the parametric techniques that require a priori knowledge of the distributions underlying the data, and the nonparametric techniques, which presuppose the functional form of the discriminant surfaces separating the different pattern classes.

In general, for a given unlabeled time series x, the KNN rule finds the k "closest" (neighborhood) labeled time series in the training data set and assign x to the class that appears most frequently in the neighborhood of k time series. Besides the training data, the KNN rule only requires two input parameters used for classifying a new unlabeled time series; that is, the size of the neighborhood k and a similarity function used as a measure of "closeness". KNN classification algorithm is described in Algorithm 3.2.1.

KNN may be seen to be a brute force method, since all training samples are used to measure the distances from a querying sample. However, for data sets involving time such as time series, a distance measure is an important method to incorporate time for classification. Among algorithms used for time series classification, some methods applied on the same data sets are compared in [96], which has extensive literature search and suggests that one-nearest neighbor with dynamic time warping (DTW) distance is difficult to beat. Not only one-nearest neighbor with DTW, but other k nearest neighbors with three similarity measures (DTW, Euclidean and T-Statistics) are successfully applied on classifying EEG time series signals [19]. Moreover, nearest neighbor rule can be used to pair time series motifs. Instead of calculating all pairs of time series distance comparisons, by triangular inequality, one can quickly prune off pairs of time series objects by taking the lower bounds of distances provided by triangular inequalities as shown in [76].

Algorithm 3.2.1 The k-nearest neighbor Algorithm for Classification

Input: Set of training sample(C), training label (LabelC), one test sample(Q), and the number of neighbors k.

```
1:  DistK($j$) = $\infty$ for $j = 1, \ldots, k$.   % The k nearest distances
2:  Index($j$) = 0 for $j = 1, \ldots, k$.
3:  distMax = $\infty$
4:  maxJ = 1
5:  for $i = 1$ to $N$ do
6:      Calculate $d$ = Distance($C_i$,$Q$)
7:      if d < distMax then
8:          DistK(maxJ) = $d$
9:          Index(maxJ) = $i$
10:         distMax = $\max_j$(DistK)
11:         maxJ = arg $\max_j$(DistK)
12:     end if
13: end for
14: Class = $\frac{1}{k}\sum_{j=1}^{k}$ LabelC[Index(arg $\min_j$ (DistK))]
```

Output: Class.

3.2.2 Improved k-Nearest Neighbor Rules

The nearest neighbor rule has been applied on many applications in different configurations. Methods that improve the KNN rule are developed in order to gain better classification accuracy and/or to shorten execution time. Here some recent methods are addressed in the following.

1. **Editing Training Data:**

   Because KNN decision rule depends mainly on training samples, it is believed that condensing training samples can lead to a better classification and increase execution speed. Instead of making a decision based on the entire training set, a smaller set as a reference for classifying new objects, called a representation set, may be a better choice. Many studies are proposed aiming at editing the training data.

When dealing with time series, one-nearest neighbor with DTW distance is one successful technique. A numerosity reduction technique in [96] can speed up especially the one-nearest neighbor with DTW. Applying the relationship between data set size and DTW constraints, an extremely compact data set can be produced with little or no loss in accuracy.

Except taken as a similarity-based method, nearest neighbor rule can also be used as dissimilarity-based classifiers. A prototype selection technique is proposed in [78], which can improve nearest neighbor dissimilarity-based classifiers.

Many instance reduction algorithms are compared as well as new techniques are proposed in [94]. To drop "useless" samples, information from neighbor instances are obtained to create criterions. Then, samples are ordered, noise instances are filtered, or the nearest neighbor decision boundary is smoothed.

2. **Enhancing Execution Efficiency:** In [91], nearest neighbor is used for anytime classification, which has a best-so-far answer available after given a small amount of "setup time". This anytime nearest neighbor algorithm sorts the sample indices such that useful training samples will be measured early. Hence, a better classification performance can be achieved earlier.

   The basic intuition is to find the "worst" training case and index it as the last position to be evaluated. Then the worst cases remaining are repeated found and indexed as the last unoccupied position. Data editing technique is often used to find the worst cases, i.e. prototype selection or instance reduction.

   Another method to find the worst training cases is to give every training samples a rank according to its "usefulness" to the classification. Suppose a data set can be divided into G groups. Given instances x_i, for $i \in \{1, \ldots, n\}$ having x as its nearest neighbor among $\{x_j\}$, for $j \in \{1, \ldots, n\}/i$. A leave-one-out one-nearest neighbor classification can be used for the training data and the rank of each training samples can be calculated by the following formula [91]:

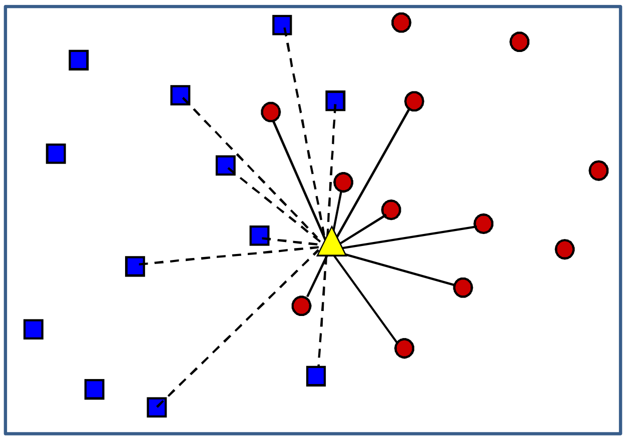
$$\text{rank}(x) = \sum_j \begin{cases} 1 & \text{if } \text{class}(x) = \text{class}(x_j) \\ -\frac{2}{G-1} & \text{otherwise.} \end{cases}$$

Many worst cases found by this ranking technique may have same ranks as one another. A further sample sorting according to distances to samples' nearest neighbor of the same class can be arranged to break the ties.

3. **Handling Unbalanced Data Sets:** When data is under-sampled, KNN method is sensitive to the number of cases selected from one of the classes (or groups) [101]. Consider an unbalanced data set such that the number of negative samples is a lot smaller than the number of positive samples. Then the k-nearest neighbors may only contain positive samples most of the time. The minority class, i.e. negative class, has only a small percentage of all samples. Hence, querying samples tends to be classified as positive because positive samples are the majority of the k-nearest neighbors.

   The problem caused by unbalanced data distributions can be solved by weighting the k nearest neighbor, proposed in [89], called neighbor-weighted KNN (NWKNN). To avoid unfair neighbor counting on unbalanced training data, NWKNN assigns a large weight for neighbors from minority group, and a small weight for neighbors from majority group.

   This dissertation proposed a method to cope with it is the nearest neighbor rule by taking the same k numbers of samples from both classes as $2k$-nearest-neighbors. This method measures distances from k nearest neighbors of each group individually. Then the querying observation is labeled as the same class as the group with shortest k nearest neighbor distance. Figure 3.3 is an example of applying this improved KNN to determine a class label for a querying sample.



d_1: Average distance to k nearest squares.
d_2: Average distance to k nearest circles.
If $d_2 < d_1$, new point is classified as a circle, Otherwise, a square.

Figure 3.3: Improved k-Nearest Neighbor Rule

3.2.3 Embedding Time Series in a Vector Space for Classification

There are many successful classification techniques designed especially for attribute data sets. However, those techniques can not be directly applied on time series data sets. Therefore, it comes an idea of transforming time series data into values in a vector space. After time series data are expressed as attribute data, they can then be classified by those classification techniques.

3.2.3.1 Embedding Time Series in a Euclidean Space

Distance based approaches for embedding univariate time series in a vector space in order to perform classification are proposed in [44]. The study shows that a successful embedding technique experimented is the Laplacian eigenmap [4, 5]. A similarity matrix using DTW distances is computed, so the Laplacian matrix can be derived from the similarity matrix. Then a generalized eigenvalue problem is constructed, and its solution provides the coordinates in the embedded space.

Once a time series data set is embedded in a Euclidean space, it can be treated as an attribute data set which many classification methods can be applied on, such as SVM.

3.2.3.2 Constructive Induction Method

A constructive induction method for classifying time series is proposed in [50] with meta-features applied to sign language recognition and ECG classification.

First, instances with recurring substructure are discovered. Then each instance can be characterized as having a set of meta-feature events, which are the recurrent substructures appropriate for the data domain. An example for the sign language recognition is to express extracted events in a two-dimensional space, called parameter space, which consists of one axis for start time of recurring events and another for duration. The parameter domain varies according to the data set that meta-features are applied to. Finally, classification techniques are used to classify those instances in the parameter domain.

3.2.3.3 Vectorizing Correlation Coefficient Matrix of MTS

A PCA-based similarity measure discussed in [98] suggests that using the correlation information among attributes of MTS data can help estimating similarity between two MTS data. Therefore, a technique proposed in [99] vectorizes components of a correlation coefficient matrix of MTS. Those vectorized components is then used as input features of SVM for classification.

First the correlation coefficient matrix for each MTS sample is computed. A correlation coefficient matrix is symmetric and its diagonal values are all one's. Therefore, to construct features for an MTS sample, the diagonal values can be eliminated and only the strict upper triangle of the correlation coefficient matrix are utilized. For an MTS sample of m attributes, the number of features that are obtained from the matrix and to be used for SVM is $\sum_{i=1}^{m-1} i = m(m-1)/2$. The vectorizing algorithm is defined in Algorithm 3.2.2.

Algorithm 3.2.2 Vectorizing an $m \times m$ correlation coefficient matrix of an MTS sample.

Input: a correlation coefficient matrix of an MTS sample with m attributes (C).

```
1: $V = [\ \ ]$; /* Initialize a vector. */
2: for $i = 1$ to $m$ do
3:     $V = [V,\quad C[i;\quad (i+1) : m]]$;
4: end for
5: return $V$;
```

Output: Vectorized correlation coefficient matrix (V).

3.2.4 Time Series Classification with Dynamic Time Alignment Kernel in Support Vector Machine

For SVM to be applicable to sequential-pattern recognition (such as time series classification), a nonlinear time alignment technique is incorporated into a kernel function, called dynamic time alignment kernel (DTAK) [85]. Because the time-alignment of two sequences is embedded in a kernel function, algorithms used to solve for SVM training and classification can still be employed without modification.

Consider a sequence of vectors $X = (x_1, x_2, \ldots, x_L)$ and $V = (v_1, v_2, \ldots, v_L)$, where x_i and $v_i \in \mathbb{R}$. Assume that the two sequences has the same length, i.e. $L = |X| = |V|$. After defining a summation of each inner product ($\circ$) between x_k and v_k, for $k = 1, \ldots, L$:

$$X \circ V = \sum_{k=1}^{L} x_k \cdot v_k,$$

an SVM classifier can be formulated as:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) \cdot \phi(x) + b = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b,$$

where n is the total numbers of samples, y_i is a class label of sample i, for $i = 1, \ldots, n$, and $y_i \in \{-1, +1\}$. Also, K is a kernel function and b is a constant. This classifier applies the kernel techniques in [83].

An alignment can be made by both linear and nonlinear time-warping function. The linear time-warping function can be the form as:

$$\psi(k) = \lceil (|X|/L)k \rceil, \quad \theta(k) = \lceil (|V|/L)k \rceil.$$

Then a new inner product operator for the linear dynamic time-alignment kernel is defined as:

$$X \circ V = \sum_{k=1}^{L} x_{\psi(k)} \cdot v_{\theta(k)}.$$

This linear warping may not be enough for time series comparison. To catch more nonlinear behavior, dynamic time warping (DTW) may be a better choice. However, instead of fining the optimal path that minimizes the accumulated distances by the original DTW, the DTAK in SVM [85] finds the optimal path that maximizes the accumulated similarity. Let $m(k)$ be a nonnegative coefficient for weighting warping path, $M_{\psi\theta}$ be the sum of all weighting coefficients, given as $\sum_{k=1}^{L} m(k)$, for normalization. Also, let Q be a positive constant used to form a warping window and to constrain on warping paths. The optimization model for the newly defined inner product (the DTAK) in SVM is formulated as follows:

$$\begin{aligned} X \circ V \quad &= \quad \max_{\psi,\theta} \frac{1}{M_{\psi\theta}} \sum_{k=1}^{L} m(k) x_{\psi(k)} \cdot v_{\theta(k)} \\ \\ \text{s.t.} \quad & \quad 1 \leq \psi(k) \leq \psi(k+1) \leq |X|, \\ & \quad \psi(k+1) - \psi(k) \leq Q, \\ & \quad 1 \leq \theta(k) \leq \theta(k+1) \leq |V|, \\ & \quad \theta(k+1) - \theta(k) \leq Q. \end{aligned}$$

This can be efficiently solved by dynamic programming. Function of $G(s,t)$ is calculated recursively starting from the largest indices of time series X and V, where $(s,t) = (|X|, |V|)$. It takes the form as:

$$G(s,t) = max \left\{ \begin{array}{l} G(s-1,t) + INP(s,t), \\ G(s-1,t-1) + 2INP(s,t), \\ G(s,t-1) + INP(s,t), \end{array} \right\},$$

where INP indicates the inter product. The optimal solution for the newly defined inner product can then be expressed as:

$$X \circ V = \frac{G(|X|, |V|)}{|X| + |V|}.$$

The DTAK in SVM can also incorporate a nonlinear mapping Φ as most kernels do. The new kernel, DTAK, is then defined as:

$$\begin{aligned} K_a(X, V) &= \Phi(X) \circ \Phi(V) \\ &= \frac{G(|\Phi(X)|, |\Phi(V)|)}{|\Phi(X)| + |\Phi(V)|}. \end{aligned}$$

Including the DTAK in SVM, one can formulate an optimization model, DTAK-SVM, for finding an optimal classifier for a time series data set. It is expressed as:

DTAK-SVM

$$\min_{w,b,\xi} \quad \frac{1}{2} W \circ W + C \sum_{i=1}^{N} \xi_i$$

$$\begin{aligned} \text{s.t.} \quad & y_i(W \circ \Phi(X^{(i)}) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0; \quad i = 1, \ldots, N. \end{aligned}$$

After solving the DTAK-SVM, optimal W and b are found. By the construction that $W = \sum_{i=1}^{n} y_i \alpha_i \Phi(x_i)$, a decision rule is obtained as follows.

$$\begin{aligned} f(X) &= \sum_{i=1}^{N} \alpha_i y_i \Phi(X^{(i)}) \circ \Phi(X) + b \\ &= \sum_{i=1}^{N} \alpha_i y_i K_z(X^{(i)}, X) + b. \end{aligned}$$

3.3 Principal Component Analysis for Time Series Feature Extraction

Nonlinear principal component analysis has been widely used for time series feature extraction. The method that provides nonlinear principal components using autoassociative neural networks is originally proposed by Kramer [57]. In the study proposed in

[87], an application of nonlinear PCA neural network for feature extraction on electrocardiogram (ECG) time series segments has been shown to be able to classify normal and abnormal states of the segments. In this application, two nonlinear principal components in feature space are constructed using autoassociative neural networks, which help produce successful classification.

Chapter 4

OPTIMIZATION MODELS FOR MULTIVARIATE DATA CLASSIFICATION

From the literature, both similarity measures and feature selection have been found very important for time series discrimination. A novel optimization technique is developed for classifying multivariate time series (MTS), called support feature machine (SFM), where a set of features is selected while a highest classification accuracy is achieved. The classification framework of SFM is based on the nearest neighbor rule, whose classification effectiveness depends largely on a chosen similarity measure.

The SFM optimization model and its variations are developed for classifying multivariate time series (MTS). The optimization model is formulated by integer programming, aiming at selecting an optimal subset of features that provides the maximum classification accuracy. The classification accuracy in SFM is the percentage of correctly classified training samples under the nearest neighbor rule.

SFM selects optimal features according to nearest neighbor rule. In a data set, values at each individual feature are evaluated separately. Similarity (or distance) measures are performed to decide the nearest neighbors. In an attribute data set, single value similarity such as Euclidean distance can be used to estimate the nearest neighbors for each feature. In an MTS data set, a similarity measure used for single time series is applied at each feature. The performance of SFM may depend on the choice of time series similarity measures.

The nearest neighbor rule can be adapted in two schemes: voting and averaging. Consider a problem of the two-group classification for a multivariate data set. At each feature, the voting scheme compares the average distances from a sample to the two groups, and give one vote to the closest group. Then the sample is assigned to the group that earns the majority of votes. The majority of votes has the number of votes that is greater than half of total number of features. The averaging scheme instead adds up average distances to each group for all features, and directly compares the distances and assign the class that is the nearest. This gives two schemes of the standard SFM: voting SFM (V-SFM) and averaging SFM (A-SFM). Instead of the standard SFM model, there are other modified SFM formulations as shown in Table 4.1. More detailed description of these models is addressed in the following sections in this chapter.

The standard SFM model has two binary decision variables. One of the decision variables decides which features should be chosen. According to the optimization formulation of SFM, the other variable automatically gives the number of correctly classified training samples under the choice of a subset of features. By relaxing the decision variable used for choosing features, the feasible region of SFM optimization model becomes larger, and hence the optimal objective value of the standard SFM becomes a lower bound of the relaxed one. This relaxation provides a higher classification accuracy at the training phase. Therefore, it forms a hypothesis that the relaxed SFM (rSFM) model may also perform better. The relaxed feature selection variable gives numbers between zero and one, which can also be viewed as weights on features. The result is prioritized feature selection. Instead of only relaxing the feature selection variables, both standard SFM and rSFM can be combined, called two-stage SFM. It first solves the standard SFM with binary variables. Then in the second stage, a relaxed SFM optimization model is formulated but only includes the chosen features from the standard SFM in its constraints.

More variations of SFM models are proposed. Since the nearest neighbor rule is applied in SFM feature selection and data classification, the k-nearest neighbor (KNN) rule can also be included. In this case, the parameter k needs to be trained to select an "optimal" model. Moreover, SFM includes all training data samples in its formulation. The quality of the training samples is unknown, therefore there may be unnecessary samples or outliers in the training set. Since average distances are used in SFM formulations, outliers may skew the sample average and hence the classifier will not perform well for test set. Data clustering can help find good quality training samples as similar

Table 4.1: Support Feature Machine Models.

Standard SFM	
V-SFM	$\max\limits_{x,y\in\Theta}\{c(x,y)=e^\mathsf{T}y: \quad e^\mathsf{T}x\geq 1\},$ where $\Theta := \left\{ x\in\{0,1\}^m, y\in\{0,1\}^n : \begin{array}{ll} Ax-\left(\frac{1}{2}e^\mathsf{T}x\right)e & \leq My \\ \left(\frac{1}{2}e^\mathsf{T}x\right)e - Ax+\epsilon e & \leq M(e-y) \end{array} \right\}$
A-SFM	$\max\limits_{x,y\in\Delta}\{c(x,y)=e^\mathsf{T}y: \quad e^\mathsf{T}x\geq 1\},$ where $\Delta := \left\{ x\in\{0,1\}^m, y\in\{0,1\}^n : \begin{array}{ll} \overline{D}x-Dx & \leq M_1y \\ Dx-\overline{D}x & \leq M_2(e-y) \end{array} \right\}$
Relaxed SFM	
V-rSFM	$\max\limits_{x,y\in\Theta^R}\{f(x,y)=e^\mathsf{T}y: \quad e^\mathsf{T}x\geq 1\},$ where $\Theta^R := \left\{ x\in\mathbb{R}^m, y\in\{0,1\}^n : \begin{array}{ll} Ax-\left(\frac{1}{2}e^\mathsf{T}x\right)e & \leq My \\ \left(\frac{1}{2}e^\mathsf{T}x\right)e - Ax+\epsilon e & \leq M(e-y) \\ \mathbf{0}\leq x\leq e & \end{array} \right\}$
A-rSFM	$\max\limits_{x,y\in\Delta^R}\{f(x,y)=e^\mathsf{T}y: \quad e^\mathsf{T}x\geq 1\},$ where $\Delta^R := \left\{ x\in\mathbb{R}^m, y\in\{0,1\}^n : \begin{array}{ll} \overline{D}x-Dx & \leq M_1y \\ Dx-\overline{D}x & \leq M_2(e-y) \\ \mathbf{0}\leq x\leq e & \end{array} \right\}$
Two-Phase SFM	
V-rSFM+	$\max\limits_{x,y\in\Theta^R_s}\{f(x,y)=e^\mathsf{T}y: \quad e^\mathsf{T}x\geq 1\},$ where $\Theta^R_s := \left\{ x\in\mathbb{R}^{\lvert S_V\rvert}, y\in\{0,1\}^n : \begin{array}{ll} A^sx-(\frac{1}{2}e^\mathsf{T}x)e & \leq My \\ (\frac{1}{2}e^\mathsf{T}x)e - A^sx+\epsilon e & \leq M(e-y) \\ \mathbf{0}\leq x\leq e & \end{array} \right\}$
A-rSFM+	$\max\limits_{x,y\in\Delta^R_s}\{f(x,y)=e^\mathsf{T}y: \quad e^\mathsf{T}x\geq 1\},$ where $\Delta^R_s := \left\{ x\in\mathbb{R}^{\lvert S_A\rvert}, y\in\{0,1\}^n : \begin{array}{ll} \overline{D^s}x-D^sx & \leq M_1y \\ D^sx-\overline{D^s}x & \leq M_2(e-y) \\ \mathbf{0}\leq x\leq e & \end{array} \right\}$

samples are grouped together. With clustering results, the training samples for SFM can be more concise.

In this chapter, it is assumed that a data set contains n samples, each with m features. Another assumption of the data set is that it can be perfectly divided into two groups, that is the data set is consists of two disjoint sets. One represents the "positive" cases, while the other the "negative". This assumption holds at almost all medical data sets, since if one set of feature values appears at both groups (positive and negative), it does not make sense to include that case for classification. One may need more features with different values to be able to classify such cases.

The following notations are defined for formulating the SFM optimization models. All vectors are column vectors. Let e denote a vector of ones in a real space of arbitrary dimension. Let $\mathbf{0}$ denote a vector of zeros in a real space of arbitrary dimension. Let $|s|$ denote the length of vector s in a real space. Let $\arg\max\limits_{x \in S} f(s)$ denote the set of $f(x)$ minimizers over the set S.

4.1 Support Feature Machine (SFM) Framework

SFM is an optimization model with a mixed-integer programming formulation, aiming at selecting an optimal subset of features that provides the maximum classification accuracy. Features selected by SFM provides strong class-separability by the nearest neighbor rule. For example, in the m-dimensional space, the SFM will find $m' \leq m$ features where the nearest neighbor rule will give the best discrimination in the new m'-dimensional space.

The main advantage of SFM is its ability to select features with high classifiability. The performance of classification is determined by the available class information from the provided features. Using all features may not be necessary for good discrimination and might not be practical in clinical settings. A small number of predictive features may provide as good performance as all features do [16]. This will be helpful for medical device companies since smaller number of features would lead to less computational requirements. In addition, it can save a lot of cost on data collection and processing, and especially it is important for physicians who are required to put in lots of labor work to eye-ball and analyze huge medical data.

4.1.1 Classification Schemes

There are two classification schemes used in SFM proposed for adapting the nearest neighbor rule: voting under distance measures (called *voting scheme*) and directly comparing averaged distances (called *averaging scheme*). Each of the two schemes is also used to bind the two types of decision variables in the optimization formulations of SFM: a feature selection variable and a classification accuracy variable.

4.1.1.1 Voting Scheme

In the voting scheme, values of an average to-the-same-class (intra-class) distance and an average to-the-different-class (inter-class) distance are calculated for each training sample.

To model the voting scheme, the SFM requires an input matrix, which is an accuracy $n \times m$ matrix $A = (a_{ij})$, $i = 1, \ldots, n,\ j = 1, \ldots, m$, where n is the number of training samples and m is the number of features. In the voting scheme of nearest neighbor if a sample of a given feature has its average to-the-same-class distance smaller than its average to-the-different-class distance, then the sample is correctly classified and is voted the value one. Otherwise, it will be voted the value zero. The entry $a_{ij} = 1$ indicates that the nearest neighbor rule correctly classified training sample i at feature dimension j; 0, otherwise.

4.1.1.2 Averaging Scheme

Similar to the voting scheme, in the averaging scheme, values of an average to-the-same-class (intra-class) distance and an average to-the-different-class (inter-class) distance are calculated for each training sample. The difference between the voting scheme and the averaging scheme is that the voting scheme compares both types of distance values and provides votes on each sample at a given feature. The averaging scheme on the other hand uses both values directly and have SFM do the comparison.

To model the averaging scheme, the SFM requires two input matrices. One is an $n \times m$ intra-class distance matrix $D = (d_{ij})$, and an $n \times m$ inter-class distance matrix $\overline{D} = (\overline{d}_{ij}),\ i = 1, \ldots, n,\ j = 1, \ldots, m$. The entry of the intra-class matrix d_{ij} is the average distance between training sample i and all other training samples from the same class at feature dimension j. The entry of the inter-class matrix $\overline{d}_{ij}$ is the average distance between the training sample i and all training samples from the different class at feature dimension j.

4.1.2 Decision Variables

The standard SFM model has two binary decision variables. One is used for feature selection and the other is for recording classification accuracy. The feature selection variable decides which features should be chosen. The classification accuracy variable records the correctly classified samples according to the set of chosen features and based on one of the nearest neighbor classification schemes. Let x be a vector of length m, and y be a vector of length n. The two types of decision variables are defined as follows.

Definition 4.1.1.

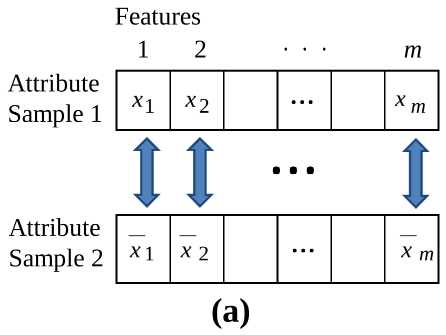
$$x_j = \begin{cases} 1, & \text{if feature } j \text{ is chosen by SFM;} \\ 0, & \text{otherwise, for } j = 1, \ldots, m. \end{cases}$$

$$y_i = \begin{cases} 1, & \text{if sample } i \text{ is correctly classified by SFM;} \\ 0, & \text{otherwise, for } i = 1, \ldots, n. \end{cases}$$

4.1.3 Nearest Neighbor Rule and Similarity Measures

The nearest neighbor rule is a very intuitive classification method, which assigns an unlabeled sample to the class whose baseline samples are on average closest to the sample. However, the classic k-nearest neighbor approach faces problems with an unbalanced class distribution. Performance of k-nearest neighbor becomes very sensitive to the number of samples in classification [101]. A possible method to avoid issue on unbalanced data sets is the neighbor-weighted k-nearest neighbor rule in [89].

Since all samples from the two classes are taken as neighbors, SFM does not have issues of imbalanced data sets. Both of its classification schemes (voting and averaging schemes) deal with imbalanced data sets under nearest neighbor rule. They are derived from to-the-same-class and to-the-different-class distances. Since class labels of training samples are known, to-the-same-class and to-the-different-class distances can be evaluated according to a chosen similarity measure. Observe that both voting and averaging schemes improve the nearest neighbor rule by avoiding issues on unbalanced data.



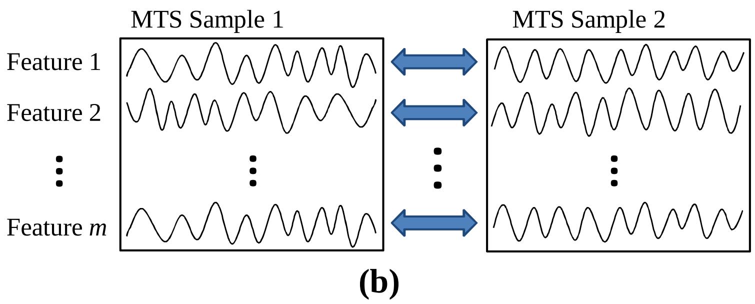


Figure 4.1: SFM Similarity Comparison: (a) Single Value Comparison for an Attribute Data Set; (b) Univariate Time Series Comparison for an MTS Data Set.

Similarity measures for single attribute values or for single time series are considered in SFM, because it evaluates each feature individually and then chooses a combination of features that gives best performance. In an attribute data set, single value similarity such as Euclidean distance can be used to estimate the nearest neighbors at each feature dimension as shown in Figure 4.1(a). An MTS sample at a feature dimension is a single time series. In an MTS data set, single time series similarity measures are applied at each feature dimension as shown in Figure 4.1(b). Examples of similarity measures for single time series can be dynamic time warping (DTW), Euclidean or T-statistics.

4.1.4 Classification Framework

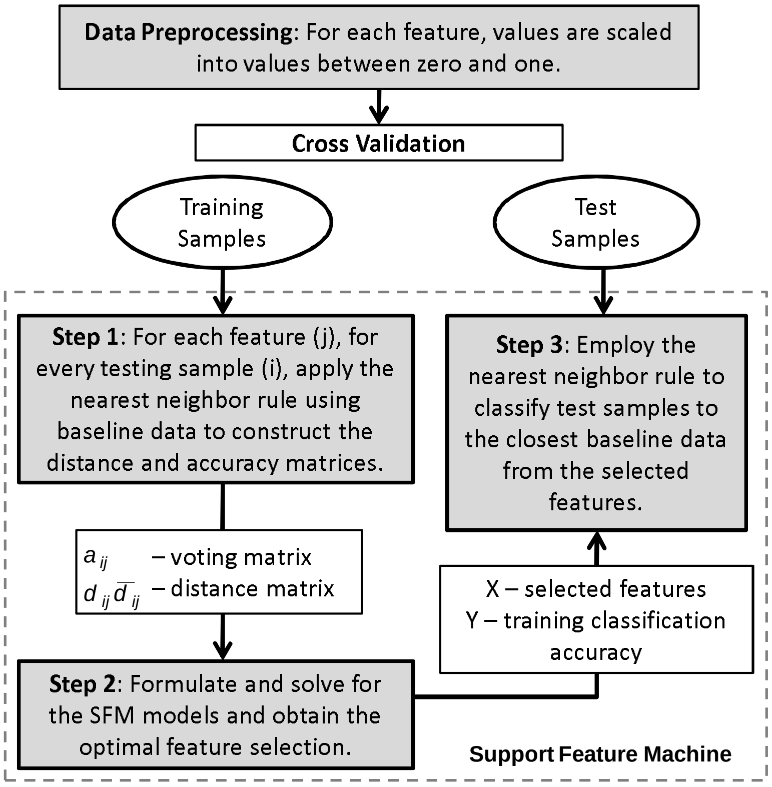
A flowchart of the proposed SFM classification framework is outlined in Figure 4.2. The framework is comprised of three key steps. In the first step, at every individual feature, the distance measure is used to generate the accuracy matrix and the distance matrices as an input to SFM optimization models. There are two parameters in the SFM classification that need to be selected to incorporate multiple decisions from all features: *majority voting* and *distance averaging.* In the second step, SFM optimization models are formulated and solved to select the best features, i.e., a subset of features that maximizes the classification accuracy. To see the performance of SFM, the last step is to classify each unlabeled data sample, whose class is treated as an "unknown", by applying the nearest neighbor rule (with majority voting or distance averaging) with the selected subset of features.

- **Step 1: Generating Accuracy and Distance Matrices from Training Data.**

  The accuracy matrix and the distance matrices generated from training data are the required inputs of SFM. An Accuracy matrix is an input of the SFM using voting scheme (voting SFM, or V-SFM), while two distance matrices, intra-class and inter-class distance matrices, are inputs of the SFM using averaging scheme (averaging SFM or A-SFM).

- **Step 2: Building Optimization Models of SFM**

  After the accuracy matrix and the distance matrices are constructed in Step 1, V-SFM and A-SFM optimization models can be formulated (for more details, see the following sections). V-SFM selects the features that gives the majority correct votes (value one's) as shown in Figure 4.3. A-SFM also tries to reach an optimal selection of features such that the sum of intra-class average distances (d_{ij}) are smaller than the sum of inter-class average distances ($\overline{d}_{ij}$) from the



Data Preprocessing: For each feature, values are scaled into values between zero and one.
Cross Validation
Training Samples
Test Samples
Step 1: For each feature (j), for every testing sample (i), apply the nearest neighbor rule using baseline data to construct the distance and accuracy matrices.
Step 3: Employ the nearest neighbor rule to classify test samples to the closest baseline data from the selected features.
a_{ij} – voting matrix
$d_{ij}\bar{d}_{ij}$ – distance matrix
X – selected features
Y – training classification accuracy
Step 2: Formulate and solve for the SFM models and obtain the optimal feature selection.
Support Feature Machine

Figure 4.2: Flowchart of the Support Feature Machine Framework

selected features (Figure 4.4). In other words, the goal is to find a subset S of all possible combinations of features ($S \subseteq \{1, 2, ..., m\}$) such that $\sum_{j \in S} d_{ij} < \sum_{j \in S} \bar{d}_{ij}$. As a result, based on the selected features, same class samples are close to each other and are away from the different class as much as possible.

- **Step 3: Applying SFM to Classify Unlabeled Data**

  After obtaining the optimally selected features by solving SFM optimization models in Step 2, test samples are treated as unlabeled samples and are classified according to those selected features. V-SFM classifies an unlabeled sample to the class with majority vote from all selected features. A-SFM classifies an unlabeled sample to the class whose baseline training samples are more similar to the sample based on the dimension of selected features. After each test sample is labeled by SFM schemes, accuracies of SFM schemes can be calculated by comparing the

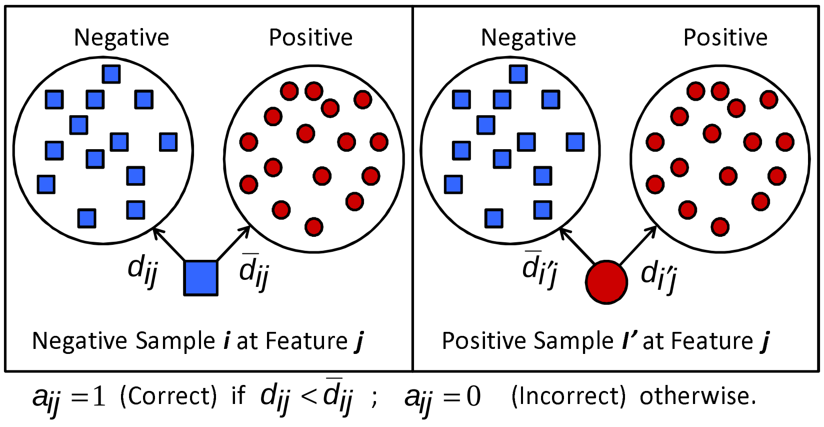

Negative
Positive
Negative
Positive
d_{ij}
$\bar{d}_{ij}$
$\bar{d}_{i'j}$
$d_{i'j}$
Negative Sample *i* at Feature *j*
Positive Sample *I'* at Feature *j*
$a_{ij} = 1$ (Correct) if $d_{ij} < \bar{d}_{ij}$; $a_{ij} = 0$ (Incorrect) otherwise.

Figure 4.3: Input Construction Process of Voting Support Feature Machine (V-SFM)

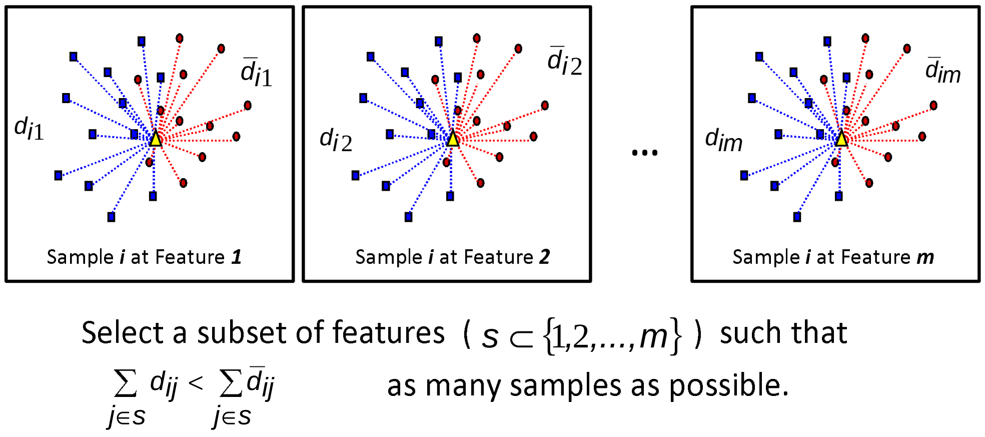

d_{i1}
$\bar{d}_{i1}$
Sample *i* at Feature *1*
d_{i2}
$\bar{d}_{i2}$
Sample *i* at Feature *2*
...
d_{im}
$\bar{d}_{im}$
Sample *i* at Feature *m*
Select a subset of features ($s \subset \{1,2,\ldots,m\}$) such that
$\sum_{j \in s} d_{ij} < \sum_{j \in s} \bar{d}_{ij}$ as many samples as possible.

Figure 4.4: Input Construction Process of Averaging Support Feature Machine (A-SFM)

labeled class with the actual class of each sample.

4.2 Standard SFM

The optimization models of SFM is to maximize the classification accuracy according to the accuracy matrix and the distance matrices constructed as described in previous section. A set of all possible solutions of an optimization problem is called a feasible region. The standard SFM has two types of feasible regions. One feasible region is for V-SFM and the other feasible region is for A-SFM, which are defined in Definition 4.2.1. In the definition, $M = \frac{m}{2}$ and $0 < \epsilon < \frac{1}{2}$, which is used to break a tie during voting. Θ is a set used in the voting scheme to ensure that when $y_i = 1$, the combination of selected features ($S = \{j \in \{1, \ldots, m\} : x_j = 1\}$) gives majority correct votes to sample i. Δ is a set used in the averaging scheme to ensure that when $y_i = 1$, sample i under the selected features S has its intra-class average distances smaller than its inter-class average distances.

Definition 4.2.1.

$$\Theta := \left\{ x \in \{0,1\}^m, y \in \{0,1\}^n : \begin{array}{ll} Ax - \left(\frac{1}{2}e^{\mathsf{T}}x\right)e & \leq My \\ \left(\frac{1}{2}e^{\mathsf{T}}x\right)e - Ax + \epsilon e & \leq M(e-y) \end{array} \right\}$$

$$\Delta := \left\{ x \in \{0,1\}^m, y \in \{0,1\}^n : \begin{array}{ll} \overline{D}x - Dx & \leq M_1 y \\ Dx - \overline{D}x & \leq M_2(e-y) \end{array} \right\}$$

Both V-SFM and A-SFM are trying to find the maximum number of correctly classified training samples based on majority voting and intra-class versus inter-class average distances, respectively. Therefore, the objective function is to maximize $\sum\limits_{i=1}^{n} y_i$, which is defined by

$$\begin{aligned} \text{V-SFM: } z_V &= \max_{x,y \in \Theta} \{c(x,y) = e^{\mathsf{T}}y : \quad e^{\mathsf{T}}x \geq 1\}, \\ \text{A-SFM: } z_A &= \max_{x,y \in \Delta} \{c(x,y) = e^{\mathsf{T}}y : \quad e^{\mathsf{T}}x \geq 1\}. \end{aligned}$$

The last constraint in the maximization model is used to avoid an empty selection. After solving these two problems, the optimal objective value z_V (or z_A) can be obtained. It represents the number of samples correctly classified based on a selection of features which is the optimal solution x by V-SFM (or A-SFM).

4.3 Relaxed SFM (rSFM) Models

SFM with a relaxed feature selection variable x is called rSFM model. Relaxing x in the sets Θ and Δ gives Θ^R and Δ^R defined as follows:

Definition 4.3.1.

$$\Theta^R \quad := \left\{ x \in \mathbb{R}^m, y \in \{0,1\}^n : \begin{array}{ll} Ax - \left(\frac{1}{2}e^\mathsf{T}x\right)e & \leq My \\ \left(\frac{1}{2}e^\mathsf{T}x\right)e - Ax + \epsilon e & \leq M(e-y) \\ \mathbf{0} \leq x \leq e & \end{array} \right\}.$$

$$\Delta^R \quad := \left\{ x \in \mathbb{R}^m, y \in \{0,1\}^n : \begin{array}{ll} \overline{D}x - Dx & \leq M_1 y \\ Dx - \overline{D}x & \leq M_2(e-y) \\ \mathbf{0} \leq x \leq e & \end{array} \right\}.$$

Again, the classification accuracy will be maximized according to voting and averaging scheme individually. Thus, the rSFM models with voting scheme (V-rSFM) and averaging scheme (A-rSFM) are defined by:

$$\begin{aligned} \text{V-rSFM: } z_V^R &= \max_{x,y \in \Theta^R} \{f(x,y) = e^\mathsf{T}y : \quad e^\mathsf{T}x \geq 1\}, \\ \text{A-rSFM: } z_A^R &= \max_{x,y \in \Delta^R} \{f(x,y) = e^\mathsf{T}y : \quad e^\mathsf{T}x \geq 1\}. \end{aligned}$$

Note that the relaxation of x's gives a larger feasible region of feature selection. Thus it is possible to find a better solution other than binary values of x. The following Lemma 4.3.1 shows that the training classification performance of rSFM is an upper bound of that of SFM.

Lemma 4.3.1. *(i) If the set Θ^R relaxes x in the set Θ, then $z^V \leq z_V^R$.*

(ii) If the set Δ^R relaxes x in the set Δ, then $z_A \leq z_A^R$.

Proof. If the set Θ^R relaxes x in the set Θ, $(x^*, y^*) \in \Theta \subseteq \Theta^R$ and $z_V = c(x^*, y^*) \leq f(x^*, y^*)$. As $(x^*, y^*) \in \Theta^R$, $f(x^*, y^*)$ is a lower bound on z_V^R, and so $z_V \leq f(x^*, y^*) \leq z_V^R$.

Similarly, if the set Δ^R relaxes x in the set Δ, $(x^*, y^*) \in \Delta \subseteq \Delta^R$ and $z_A = c(x^*, y^*) \leq f(x^*, y^*)$. As $(x^*, y^*) \in \Delta^R$, $f(x^*, y^*)$ is a lower bound on z_A^R, and so $z_A \leq f(x^*, y^*) \leq z_A^R$. □

Other than relaxing the feature selection variables x, a normalization to restrict the scale of $x's$ is introduced by adding a normalization constraint, called $\overline{rSFM}$ model.

The sum of the values of x_j is constrained to be equal to one. The resulting models are:

$$\begin{aligned}
\text{V-}\overline{rSFM}\text{: } z_V^N &= \max_{x,y\in\Theta^R}\{f(x,y) = e^{\mathsf{T}}y : \quad e^{\mathsf{T}}x = 1\},\\
\text{A-}\overline{rSFM}\text{: } z_A^N &= \max_{x,y\in\Delta^R}\{f(x,y) = e^{\mathsf{T}}y : \quad e^{\mathsf{T}}x = 1\}.
\end{aligned}$$

4.4 Two-Phase SFM

Since relaxing x provides a larger feasible region, one might think that the rSFM models may over fit the data, and may lower its testing performances. Therefore, a method is proposed to give a smaller relaxed region, which has two phases, called SFM Plus Relaxation (rSFM+) Model. Phase I solves the standard SFM, which has binary x. Then only the selected features are considered to be relaxed at Phase II. The relaxation in the Phase II is a modification of the rSFM models. It excludes the unselected features, and results a reduced relaxation model. New parameters and variables are defined for the selected features.

Definition 4.4.1. Let S_V and S_A be the index set of the selected features from V-SFM and A-SFM, respectively. $S \in \{S_V, S_A\}$. Then

$$\begin{aligned}
x^s &= (x_j)_{j\in S},\\
A^s &= (a_{ij})_{\substack{i=1,\dots,n,\\ j\in S}},\\
D^s &= (d_{ij})_{\substack{i=1,\dots,n,\\ j\in S}},\\
\overline{D^s} &= (\bar{d}_{ij})_{\substack{i=1,\dots,n,\\ j\in S}},
\end{aligned}$$

where a_{ij} is the the element of matrix A at its ith row and jth column.

According to the selected index sets S_V and S_A, the following convex sets can be defined as:

Definition 4.4.2.

$$\Theta_s \quad := \left\{ x \in \{0,1\}^{|S_V|}, y \in \{0,1\}^n : \begin{array}{ll} A^s x - (\frac{1}{2}e^\mathsf{T} x)e & \leq My \\ (\frac{1}{2}e^\mathsf{T} x)e - A^s x + \epsilon e & \leq M(e-y) \end{array} \right\}$$

$$\Delta_s \quad := \left\{ x \in \{0,1\}^{|S_A|}, y \in \{0,1\}^n : \begin{array}{ll} \overline{D^s}x - D^s x & \leq M_1 y \\ D^s x - \overline{D^s}x & \leq M_2(e-y) \end{array} \right\}$$

The relaxed model used in the Phase II of rSFM+ has the following definitions, which is the feasible region obtained by relaxing the selection variables in the selected index sets.

Definition 4.4.3.

$$\Theta_s^R \quad := \left\{ x \in \mathbb{R}^{|S_V|}, y \in \{0,1\}^n : \begin{array}{ll} A^s x - (\frac{1}{2}e^\mathsf{T} x)e & \leq My \\ (\frac{1}{2}e^\mathsf{T} x)e - A^s x + \epsilon e & \leq M(e-y) \\ \mathbf{0} \leq x \leq e & \end{array} \right\}$$

$$\Delta_s^R \quad := \left\{ x \in \mathbb{R}^{|S_A|}, y \in \{0,1\}^n : \begin{array}{ll} \overline{D^s}x - D^s x & \leq M_1 y \\ D^s x - \overline{D^s}x & \leq M_2(e-y) \\ \mathbf{0} \leq x \leq e & \end{array} \right\}$$

After defining its feasible set, the Phase II optimization models of V-SFM with relaxed x^s (V-rSFM+) and A-SFM with relaxed x^s (A-rSFM+) are given by:

$$\begin{aligned} \text{V-rSFM+:} \quad \tilde{z}_V^R &= \max_{x,y \in \Theta_s^R} \{f(x,y) = e^\mathsf{T} y : \quad e^\mathsf{T} x \geq 1\}, \\ \text{A-rSFM+:} \quad \tilde{z}_A^R &= \max_{x,y \in \Delta_s^R} \{f(x,y) = e^\mathsf{T} y : \quad e^\mathsf{T} x \geq 1\}. \end{aligned}$$

Note that the relaxation model in Phase II is used to fine tune and prioritize the features x's selected in Phase I. It is easy to observe that, in the training phase, the performance of rSFM+ is an upper bound of the standard SFM (see Lemma 4.4.1). However, due the feature selection in Phase I, the performance of rSFM+ is a lower bound of rSFM in the training. These extensions of SFM framework give a rise of how restricted the classification models are. Although theoretical relationships among these models can be drawn, they hold only in the training phase. Therefore, the relationships might not be the same in the testing phase since the classification models might overfit the training data.

Lemma 4.4.1. *1. If the set Θ_s^R relaxes x^s in the set Θ_s, then $z^V \leq \tilde{z}_V^R \leq z_V^R$.*

2. If the set Δ_s^R relaxes x^s in the set Δ_s, then $z_A \leq \tilde{z}_A^R \leq z_A^R$.

Proof. Define

$$\begin{aligned} \tilde{z}_V &= \max_{x,y \in \Theta_s} \{c(x,y) = e^\mathsf{T} y : \quad e^\mathsf{T} x \geq 1\} \\ \tilde{z}_A &= \max_{x,y \in \Delta_s} \{c(x,y) = e^\mathsf{T} y : \quad e^\mathsf{T} x \geq 1\} \end{aligned}$$

If the set Θ_s^R relaxes x in the set Θ_s, $(x^*, y^*) \in \Theta_s \subseteq \Theta_s^R$ and $\tilde{z}_V = c(x^*, y^*) \leq f(x^*, y^*)$. As $(x^*, y^*) \in \Theta_s^R$ is a lower bound on $\tilde{z}_V^R$, $\tilde{z}_V \leq f(x^*, y^*) \leq \tilde{z}_V^R$.

$\Theta_s^R \subseteq \Theta^R$, so $\tilde{z}_V^R \leq z_V^R$. Also, $\Theta_s \subseteq \Theta$, so $\tilde{z}_V \leq z_V$. In addition, the construction of Θ_s is based on the optimal solution of z_V, which implies $x_j = 0 \quad \forall j \in \{1, \ldots, m\} \backslash S_V$. Hence, $\tilde{z}_V = z_V$. As a result, $z_V = \tilde{z}_V \leq \tilde{z}_V^R \leq z_V^R$.

Similarly, $z_A = \tilde{z}_A \leq \tilde{z}_A^R \leq z_A^R$. □

4.5 *k*-Nearest Neighbor SFM

SFM feature selection and data classification is based on the nearest neighbor rule. Another variation of SFM can be made by including *k*-nearest neighbor (KNN) rule. Here *k* is another model training parameter of SFM. In this case, the parameter *k* needs to be trained to select an "optimal" model accordingly. Both the voting and averaging schemes can be modified to including the KNN rule. It can be made when generating the accuracy matrix and the average distance matrices. Instead of considering distances from a sample to all other samples in the same class or a different class, one can consider only the distances from a sample to only the *k* nearest neighbors in the same class or a different class.

4.5.1 Voting Scheme for KNN SFM

In the standard voting scheme, values of an average intra-class distance and an average inter-class distance are calculated for each training sample. To include the KNN rule in the standard SFM, only the distances to the k nearest neighbors in the same class is averaged, and only the distances to the k nearest neighbors in a different class is averaged for each sample.

To model the voting scheme, the KNN SFM also requires an input matrix, which is an accuracy $n \times m$ matrix $A = (a_{ij}),\ i = 1, \ldots, n,\ j = 1, \ldots, m$, where n is the number of training samples and m is the number of features. In the voting scheme of the KNN rule if a sample of a given feature has its average distances to the k nearest neighbors in the same class smaller than its average distances to the k nearest neighbors in the different class, then the sample is correctly classified and is voted the value one. Otherwise, it will be voted the value zero. Therefore, the entry $a_{ij} = 1$ indicates that the KNN rule correctly classified training sample i at feature dimension j; 0, otherwise.

4.5.2 Averaging Scheme for KNN SFM

Similar to the voting scheme, in the averaging scheme for KNN SFM, values of an average distance to the k nearest neighbors in the same class and an average distance to the k nearest neighbors in a different class are calculated for each training sample.

To model the averaging scheme, the KNN SFM requires two input matrices. One is an $n \times m$ KNN intra-class distance matrix $D = (d_{ij})$, and an $n \times m$ KNN inter-class distance matrix $\overline{D} = (\overline{d}_{ij}),\ i = 1, \ldots, n,\ j = 1, \ldots, m$. The entry of the intra-class matrix d_{ij} is the average distance between training sample i and the k nearest other training samples from the same class at feature dimension j. The entry of the inter-class matrix $\overline{d}_{ij}$ is the average distance between the training sample i and the k nearest training samples from the different class at feature dimension j.

4.6 Clustered SFM

Observe that SFM includes all training samples in its formulation. The quality of the training samples is unknown, there may be unnecessary samples in the training set. Since SFM classification relies mainly on distance measures, outliers may skew the sample average. Hence the classifier may not perform well as expected in the testing.

Data clustering can help finding good quality training samples and excluding outliers. With clustering results, the training samples for SFM can be more concise.

The idea of clustering training samples for SFM is to remove non-informative samples such as outliers. Clustering techniques can help with such problems since it tries to group similar samples together. There are many clustering techniques in the literature, such as k-means clustering [48]. The k-means clustering is a method that partitions n samples into k clusters, so that each sample belongs to a cluster with the nearest mean. However, such methods do not apply time series similarity measures, and the number of clusters (k) needs to be specified in advance. They are not suitable for MTS. For MTS classification, the number of clusters in each class is unknown, and a similarity measure is essential in grouping data samples into clusters. The quality cluster algorithm (QT clustering) proposed in [45] is a suggested method that satisfies these requirements. The idea of QT clustering is to find the data sample with the largest number of neighbors within the diameter d. The sample and its neighbors form a cluster. The algorithm is then operated for the rest of samples iteratively until every sample is assigned to a cluster. The algorithm will automatically generate the optimal number of classes in the data set. Moreover, it incorporates a pre-calculated distance matrix in its model, and thus provides the flexibility of choosing a distance measure. QT clustering applies a dynamic programming technique and generates an optimal number of clusters given a diameter threshold value d. However, in the application, QT clustering tends to cluster all data in one group, which does not give sufficient information for further analysis.

A sample-preserved k-median (SPKM) clustering technique is proposed in this dissertation in order to perform the unsupervised learning incorporating similarity measures. To combine the SPKM clustering with SFM, one can obtain the clusters from a clustering technique, e.g. SPKM clustering, and use the cluster centers as the training baseline for SFM as shown in Algorithm 4.6.1.

Training samples are reduced after optimal clusters are generated. Suppose $n'(n' \leq n)$ clusters are found. SFM models can be reduced. The accuracy matrix, and the average distance matrices are reduced to be of size $n' \times m$, and classification accuracy variable y has length n'. The clustered SFM only takes the cluster centers as baseline data, which saves computation time and may improve the quality of training samples.

Algorithm 4.6.1 Sample-Preserved k-Median Clustering as Sample Selection for Support Feature Machine

Input: A data set, k_1 and k_2.

1. Separate data into disjoint training and testing sets.
2. Group positive training data into k_1 clusters and negative training data into k_2 clusters.
3. Train the SFM using the k_1 positive cluster centers and k_2 negative cluster centers.
4. Use the testing set together with the $(k_1 + k_2)$ cluster centers to evaluate the performances of the nearest neighbor rule and SFM.

Output: Nearest neighbor and SFM classification accuracies.

Chapter 5

APPLICATIONS OF DEVELOPED OPTIMIZATION MODELS FOR CLASSIFICATION

A medical data set is assumed to be perfectly separable. In other words, the medical data set is a union of two disjoint sets. One set represents the “positive” group associated with patients having a specific medical condition or disease, and the other set represents the “negative ” group associated with patients who do not have the medical condition or disease. This assumption makes sense, because if one pattern appears at both positive (abnormal) and negative (normal) classes, it means that the medical condition of that pattern is unknown, and more features from diagnostic procedures are needed. When training a classifier, such patterns should be excluded or more features should be included. Thus, a perfectly separable data set can be assumed.

In this chapter, a medical data set is consisted of n observations (patients); each is represented by m features (or attributes). In other words, each data sample in the medical database is represented by an m-dimensional vector corresponding to clinical characteristics of a patient.

Classification techniques applied on medical data sets can be used as a tool for medical diagnosis. The main purpose of this chapter is to evaluate the performance of the developed classification framework, Support Feature Machine (SFM), discussed in Chapter 4. Although SFM is designed especially for MTS data sets [20], it can also be used on classifying attribute data sets. Samples in an attribute data set are often viewed as vectors. Single value similarity measures such as Euclidean distance can be applied on each feature. As a result, SFM is applicable to attribute data sets as well [36].

SFM performance is evaluated on both MTS and attribute types of real world medical data sets. The MTS data sets are consists of electroencephalogram (EEG) time series with multi-channels (described in Section 5.2.2). The attribute data sets are medical data sets acquired from the University of California Irvine (UCI) repository [3] (described in Section 5.2.1).

Classification performance of SFM is then compared with performance of other existing classification techniques, such as linear Support Vector Machines [39], nonlinear Support Vector Machines with Gaussian Kernel [40], logical analysis of data [11] and non-optimized nearest neighbor rule [22].

Note that notations for SFM formulation used in this chapter are the same as those in Chpater 4.

5.1 Background of Medical Diagnosis

Medical diagnosis is a process of identifying a medical condition or disease by its symptoms, and from results of various diagnostic procedures. By taking each patient as a sample, those symptoms and analytic results of patients can be viewed as their attributes (or features), and thus forms a medical data set. If the medical data set consists of two types of patients, one is abnormal and the other is normal, a classifier can be trained according to labeled data, whose class labels are known. It can then be used to classify unlabeled samples (patients), whose class labels are unknown. Therefore, a classification technique can be used as a tool for medical diagnosis.

Today's clinical testings and experiments to diagnose patients have resulted in massive data sets, which physicians have to mine so that they can accurately treat the patients. Those data ranges from simple blood pressure and heart rate to magnetic resonance imaging (MRI) and electroencephalogram (EEG) waveforms. In such situation, physicians do need a tool to quickly analyze the medical data signal and detect the patterns that can be used to identify the causes of the symptoms or diseases. Data mining studies are dedicated to the analysis and computational methods to deal with massive medical data [77]. Especially there are several data classification and decision making problems arising in medical diagnosis. In recent years, medical diagnosis have been shown to be examples of data classification problem in clinical settings [43, 81, 90]. To improve current medical diagnosis, data mining techniques can be used in identifying a disease from its symptoms and making a decision to diagnose a patient.

5.2 Medical Data Sets

In order to evaluate classification performance of SFM framework, it is tested on real world medical data sets. They consists of both MTS data sets and attribute data sets. The MTS data sets that the SFM is tested on are obtained from EEG time series signals with multi-channels. Note that the channels, representing where the EEG signals are obtained, are taken as features in MTS data sets. For example, univariate EEG time series has only one channel. Moreover, the attribute data sets that SFM is tested on are well-known benchmark data sets acquired from the University of California Irvine (UCI) repository [3]. They are associated with breast cancer, heart disease, diabetes, and liver disorders. Descriptions of these data sets are as follows. Note that in order to reduce the bias of scale, all the entries of all features in all data sets were linearly scaled into values in the interval of $[0, 1]$.

5.2.1 Attribute Data Sets

Four attribute data sets are well-known benchmark data sets acquired from the University of California Irvine (UCI) repository [3]. Parameters of these data sets are shown in Table 5.1. They are related to breast cancer, heart disease, diabetes and liver disorders, which are described as follows.

Table 5.1: Parameters of UCI Data Sets [3]

	# of Observations		# of Features
Data set	Positive	Negative	
Breast Cancer Wisconsin (WDBC)	212	357	30
Heart Disease (HD)	137	160	13
Pima Indian Diabetes (PID)	268	500	8
Bupa Liver Disorders (BLD)	200	145	6

The **Breast Cancer Wisconsin Diagnostic (WDBC)** data set consists of 569 instances, each with 30 real-valued features which were computed from a digitized image of a fine needle aspirate of a breast mass. These features, computed for each cell nucleus, are considered to be important characteristics for breast cancer diagnosis. Those tumor's characteristics include its radius, texture, perimeter, area, smoothness, compactness, concavity, number of concave portions of the contour, symmetry and fractal dimension. Using these characteristics, oncologists made two diagnosis outcomes: malignant (positive) or benign (negative) tumors. The WDBC data set contains 357 benign samples and 212 malignant samples.

The **Cleveland Heart Disease (HD)** data set consists of 300 instances, each with 13 select features that are believed to be a good indicator for the angiographic disease status. Those features include chest pain type (typical and atypical angina, non-anginal pain, and asymptomatic), resting blood pressure, serum cholestoral, resting electrocardiographic results (normal, abnormality, probable), maximum heart rate, indicator of exercise-induced angina, ST depression, slope of the peak exercise ST segment, number of major vessels colored by flourosopy and the main criterion that physicians use to determine the diagnosis of heart disease is the narrowing in diameter of any major blood vessel. The diagnosis was considered to be positive (presence of heart disease) if the diameter of any major vessel was narrowed by more than 50%; and negative otherwise. The HD data set contains 137 positive cases and 160 negative cases, after removing the samples with missing attribute values.

The **Pima Indians Diabetes (PID)** data set consists of 768 adult female samples, each with 8 features and the class attribute (presence of diabetes). The features were derived from the patients' insulin doses, the outcome of periodically (before and after each meal) blood tests including blood glucose measurement, meal ingestion, and exercise activity. The PID data set contains 258 positive cases and 500 negative cases.

The **BUPA Liver Disorders (BLD)** data set consists of 345 male samples, each with 6 features concerning with the patients' biological markers, the amount of daily alcoholic beverage consumption, and the class attribute (presence of liver disorders). Those biological markers include the mean corpuscular erythrocyte volume (MCV), carbohydrate-deficient transferrin (CDT), gamma-glutamyltransferase (GGT), total plasma homocysteine and folate. The BUPA data set contains 200 positive cases and 145 negative cases.

From the literature, WDBC is a clean data set on which many classification methods provide highly accurate diagnostic models. HD is not as clean as WDBC, but still reasonably predictable. On the other hand, it is known that PID and BLD are very complex data sets and difficult to classify.

5.2.2 Multivariate Time Series Data Sets

EEG signals of ten epilepsy patients who were suffering from seizures were recorded. The recordings are continuous long-term (3 to 13 days) multichannel intracranial EEG recordings. There are 26 standard channels from every patient where EEG time series are analyzed and investigated. During the recordings, 7 to 23 seizures occurred and the time when seizure occurred were marked. The recording durations and number of seizures occurred during the recordings of each patient are outline in Table 5.2.

In addition, these EEG signals are preprocessed in order to filter noises, capture essential information, and enable a classification. The measures of chaos, Short-Term Maximum Lyapunov Exponent (STLmax), has been previously shown capable of contemplating dynamical mechanisms of the brain network from EEG signals [21]. Therefore, the estimation of STLmax is calculated for all EEG recordings, which is used as the final format for classification.

Table 5.2: EEG Data Set Characteristics [20]

Patient ID	Duration of EEG (days)	♯ of seizures
1	3.55	7
2	10.93	7
3	8.85	22
4	5.93	19
5	13.13	17
6	11.95	17
7	3.11	9
8	6.09	23
9	11.53	20
10	9.65	12
Total	84.71 days	153

These recordings need to be sampled into a MTS data set before applying the SFM framework. The goal of SFM is to classify the normal and pre-seizure states of each patient. Therefore, time series segments of normal and pre-seizure states should be sampled from all 26 channels, and then a MTS data set of two groups, normal and pre-seizure will be constructed. Finally, a classification with SFM will be performed separately for each patient.

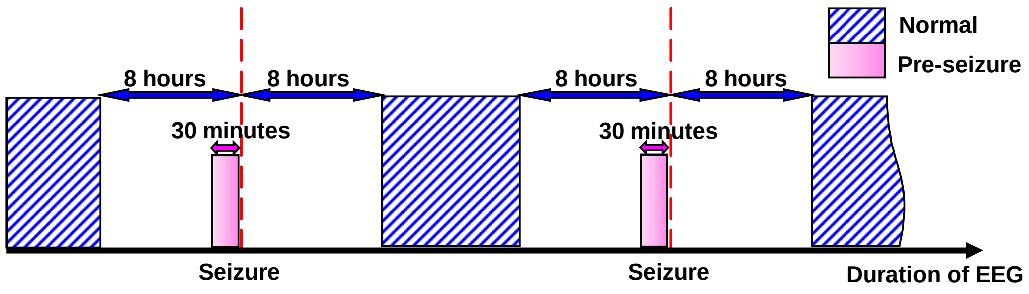


Figure 5.1: An Example of Normal and Pre-seizure EEG Segments.

To obtain MTS data, two groups (normal and pre-seizure) of five-minute EEG epochs are randomly sampled from the continuous recordings in each patient. In the STLmax format of EEG recordings, five-minute EEG epochs contains 30 values. Per seizure, five EEG epochs from each of normal and pre-seizure states are randomly and uniformly sampled. Figure 5.1 illustrates normal and pre-seizure segments of EEG duration. Normal EEG samples are selected from EEG recordings that is more than eight hours apart from a seizure. Pre-seizure EEG epochs are selected from EEG recordings during the 30-minute interval before. For example, the first patient had seven seizures; therefore, 70 EEG epochs (35 (7×5) normal and 35 pre-seizure) will be sampled from the patient's recordings. Hence, the MTS data set of the first patient has 70 samples, each sample has 26 features, and each feature is a time series of length 30 points. Finally, the ten EEG MTS data sets are obtained.

5.3 Performance Evaluation

A classification technique is aimed at employing a trained classifier (or a trained decision rule) to assign a class label to an unlabeled case. Such classifier is trained based on samples whose class labels are known. The performance of a classification technique is determined according to how test samples are classified by the trained classifier (or the trained decision rule). Therefore, a data set is usually divided into two groups. One becomes a training set, and the other becomes a test set. Note that the class labels of test set samples are treated as unknown. After the trained classifier assigns class labels to the test set samples, those sample labels can be compared with their actual ones.

In medical domain, the performance of data classification is commonly presented in terms of sensitivity and specificity, which are statistically related to type I and type II errors. Sensitivity measures the fraction of positive test samples that are correctly classified as positive. That is, $\textit{sensitivity} = TP/(TP + FN)$, where TP and FN denote number of true positives and false negatives, respectively. Specificity measures the fraction of negative test samples that are correctly classified as negative. Let FP and TN denotes number of false positives and true negatives, respectively, then $specificity = TN/(FP + TN)$. An overall accuracy is defined as $\textit{accuracy} = (TP + TN)/(TP + FP + TN + FN)$.

Results of classification performance can also be used to select the best parameter setting in a classification framework. The training phase was used to perform classification of possible parameter settings and test them in the testing phase. The parameter setting with the best performance can then be identified. Consider examples of classification techniques: SFM, support vector machine (SVM) and nearest neighbor rule (NN). Note that the parameters of the SFM framework are the values of feature selection variables $x's$. The parameters of SVM are the characteristics of the optimal hyperplane, i.e., the weighting vector ω and the scalar γ. The training for the NN technique can be skipped because there was no parameter to be trained. The best parameter is often referred to the most appropriate trade-off to maximize the detection rate ($\textit{sensitivity}$) and minimize the false alarm rate ($1 - specificity$). An ROC plot is represented with the false alarm rate along the X-axis and the detection rate along the Y-axis. The best parameter setting for each approach was selected such that it was closest to the ideal classifier (100% sensitivity and 100% specificity), which is located at the top left-hand corner of an ROC plot.

There are many possible ways to divide data into training and testing sets. In order to estimate how well the model, which is learned from some training data, can perform on future data, cross validation is an approach. It has been widely used to check sampling bias in the training and testing phases. After performing classification from several combinations of training and testing sets, an evaluation can be concluded.

5.3.1 Training and Testing: Cross Validation

Cross validation techniques are motivated by two fundamental problems in classification:

- **Performance Estimation.** Cross validation is typically used to estimate how well a trained classifier will perform on future as-yet-unseen data (testing data).
- **Model Selection.** Almost invariably, all pattern recognition techniques have one or more free parameters. By doing cross validations on different parameter settings of a classifier, performances of those settings can be compared. Then, "optimal" parameters with best performance are found.

Cross validation is a model evaluation method that does not use the entire data set when training a classifier. Part of the data is removed before training a classifier. When training is done, the removed data is used to test the performance of the learned classifier on "new" data. The output labels provided by the classifier and the actual label of the removed data (testing set) are compared. The errors it makes are accumulated, and the performance of the classification technique is evaluated.

This evaluation may depend on which data points end up in the training set and which end up in the testing set. Therefore the evaluation may be significantly different depending on how the division is made. Cross validation technique is extensively used as a method to estimate the generalization error based on "resampling". There are many choices of how to divide data into training and testing sets.

5.3.1.1 n-Fold Cross Validation

The idea of n-fold cross validation is to randomly divide all the data points into n mutually exclusive and approximately equal size subsets. All classification approaches are then trained and tested n times. Each time, one of n subsets is used as a testing set while the remaining $n-1$ subsets are used as training set. Thus, n different classification results are obtained for each training-testing configuration. The average of these results is used as the overall classification performance.

As mentioned in [74], the result from one n-fold cross validation may not be reliable. In order to have low mean squared error (MSE) and bias, at least ten repetitions of cross validations are suggested. After at least ten cross validation replications of n-fold cross validation are made, a good estimation of classification performance can be obtained.

5.3.1.2 Leave-One-Event-Out Cross Validation

In the case that a data set is obtained from many events and each event is related to multiple data points in that data set, randomly dividing all the data points into n mutually exclusive subsets may tend to over estimate the performance of a classifier. By doing so, one data point of an event may be allocated in the training set and another data point of the same event may be in the testing set. Thus, the trained classifier can easily recognize a similar pattern of that event. As a result, classifying another data point from the same event leads to an over estimated performance. The method of leave-one-event-out cross validation avoids sample correlations between training set and testing set. In such cases, leave-one-event-out cross validation gives a fair evaluation and helps avoid over estimation.

5.3.2 Selection of the Best Classification Schemes

To quantitatively select the best classification scheme as well as the best parameter settings (i.e., feature selection variables $x's$ for SFM, and weighting vector ω and the scalar γ for SVM), a statistical method, called receiver operating characteristics (ROC), can be applied. ROC is derived from the detection theory to identify the optimal parameter settings. The ROC analysis is used to indicate an appropriate trade-off that one can achieve between the detection rate (sensitivity, plotted on Y-axis) that is desired to be maximized, and the false alarm rate (1-specificity, plotted on X-axis) that is desirable to be minimized. Basically, the sensitivity and the specificity of each classification scheme for each parameter setting are calculated. By definition of ROC analysis, the best scheme is selected such that it is closest to the ideal classifier (best performance).

5.3.3 Demonstration of the Use of Leave-One-Event-Out Cross Validation

To show that the leave-one-event-out cross validation can avoid over estimating of the classification accuracy, the n-fold cross validation and the leave-one-event-out cross validation are tested on the simple and intuitive time series classification technique, the time series k-nearest neighbor (KNN) rule (discussed in Section 3.2.1). The experiment uses the EEG MTS data sets, and different k values ranging from 3 to 13 are tried (note that k should be and odd number). In addition, three time series similarity measures are applied: dynamic time warping (DTW), Euclidean and T-statistics distances.

In the case that a data set is obtained from many subjects and each subject is related to multiple data points in that data set, randomly dividing all the data points into n mutually exclusive subsets may tend to over estimate the performance of a classifier. For the EEG data sets, a classifier is trained for each individual patient. It can be observed that for each patient, the EEG MTS data set may still be over estimated because the pre-seizure points are from the same seizure occurrence. Hence, leave-one-seizure-out (LOSO) cross validation is assumed to be more suitable for performance estimation and parameter selection. Results of conducting LOSO cross validation for the time series KNN classification is summarized in this section.

Take Patient 10 as an example. Figure 5.2 illustrates an ROC plot of the KNN classification using three similarity measures (DTW, Euclidean and T-statistics), applied on Patient 10 using LOSO cross validation to train and test the classifier. To compare the results with n-fold cross validation, Figure 5.3 and Figure 5.4 illustrate

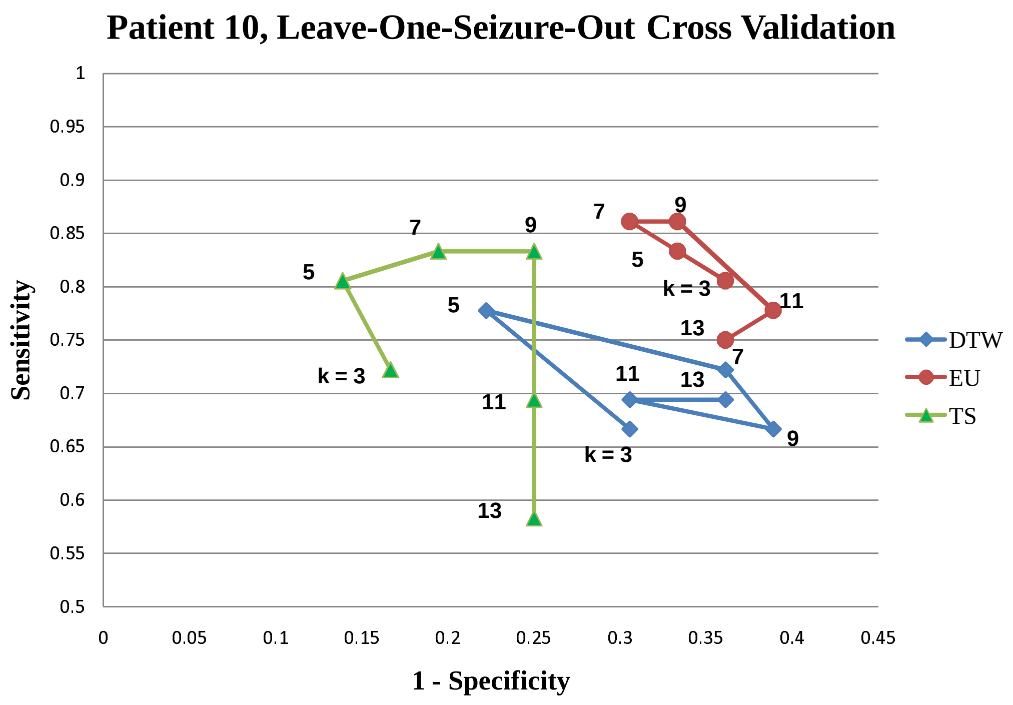


Figure 5.2: ROC plot for in Patient 10 using leave-one-seizure-out cross validation to train and test the KNN classifier with three similarity measures, dynamic time warping (DTW), Euclidean (EU) and T-statistics (TS).

an ROC plot of the KNN classification using 3-fold and 5-fold cross validation, respectively, with 10 replications to train and test the KNN classifier. Note that each point in the plot represents a value of k nearest neighbors (odd values ranging from 3 to 13) with one of the similarity measures. Comparing the results obtained by LOSO cross validation and those by n-fold cross validation, most of the paired values, *sensitivity* and $(1 - specificity)$, obtained by the n-fold cross validation are closer to the top left-hand corner of ROC plots than by the LOSO cross validation. Clearly, the n-fold cross validation over estimates the performance of KNN. Since the LOSO division only groups pre-seizure points according to the same seizure occurrence, this indicates that the method of LOSO cross validation avoids sample correlations between training set and testing set. Therefore, it avoids over estimating sensitivities. Only when the correlation between the training set and the testing set are taken out, the performance evaluation seems more reasonable.

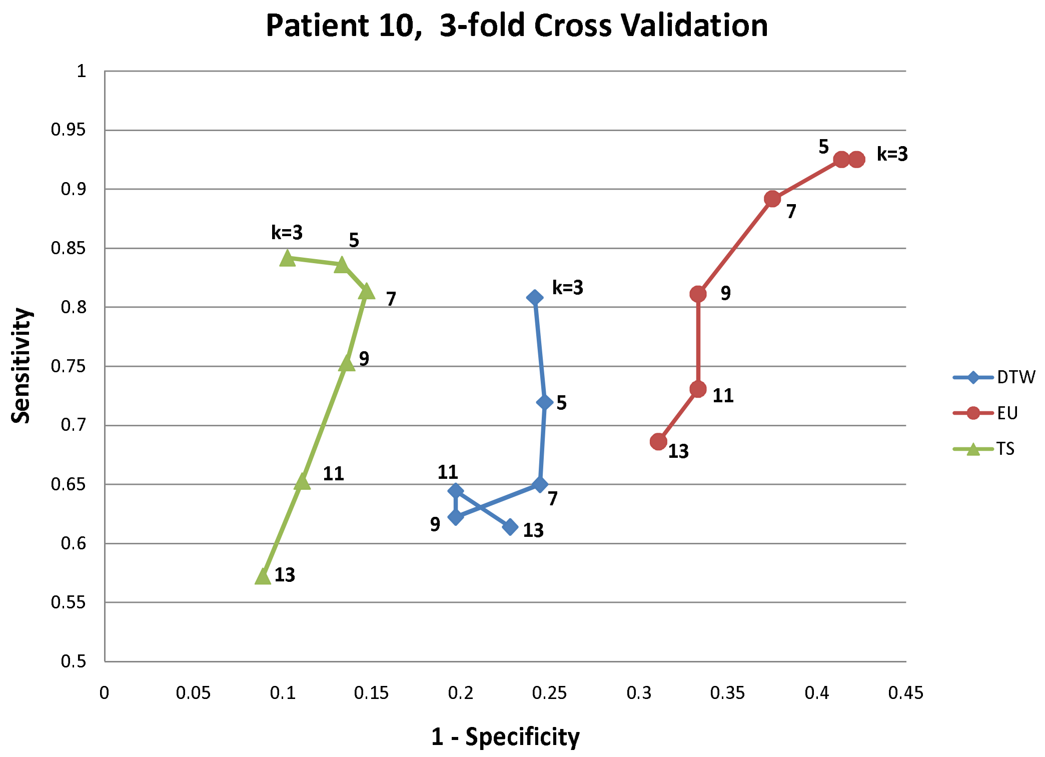


Figure 5.3: ROC plot for Patient 10 using 3-fold cross validation to train and test the KNN classifier with three similarity measures, dynamic time warping (DTW), Euclidean (EU) and T-statistics (TS).

5.4 SFM Classification on Medical Attribute Data Sets

In this section the performance of SFM framework on attribute data sets are evaluated. It is tested on four real medical data sets, which are well-known benchmark data sets acquired from the University of California Irvine (UCI) repository [3]. The four data sets are associated with breast cancer, diabetes, heart disease, and liver disorders. Since they are attribute data sets, and SFM measures sample values at each individual feature separately, a single value similarity measure is needed. The Euclidean distance measure is the most common one for measuring distances in vector space, and hence it is applied to the nearest neighbor rule that SFM is based on. To correctly estimate the classification performance of SFM, ten replications of five-fold cross validation are applied.

Training performance of the SFM models tested on the four attribute data sets is

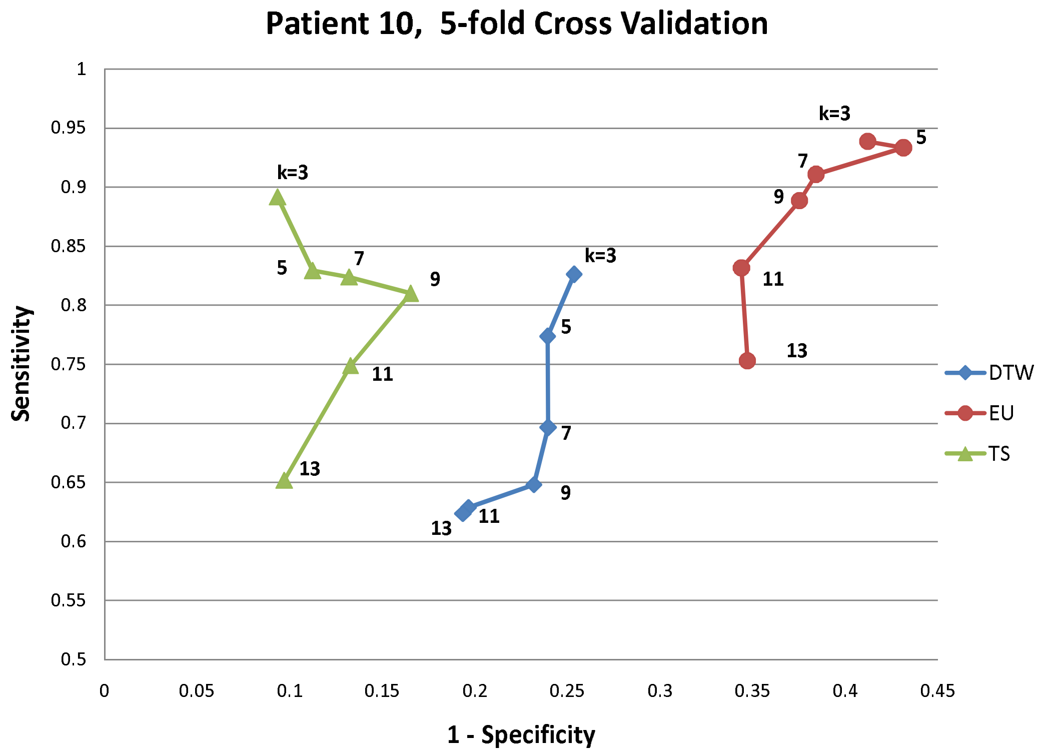


Figure 5.4: ROC plot for Patient 10 using 5-fold cross validation to train and test the KNN classifier with three similarity measures, dynamic time warping (DTW), Euclidean (EU) and T-statistics (TS).

summarized in Table 5.3. They are compared with the performance of the linear SVM and the nonlinear SVM with Gaussian kernel. Lemma 4.3.1 and Lemma 4.4.1 shows that both rSFM and rSFM+ provide classification accuracy in the training phase at least the same as the accuracy of standard SFM. Also, Lemma 4.4.1 indicate that rSFM is at least as good as rSFM+ in the training phase. These are consistent with the results shown in Table 5.3. Moreover, both of the two-phase optimization models, (rSFM+ and the normalized rSFM+) and the standard SFM give the same results in the training phase on all four data sets. The two-phase models did not improve the classification results from the standard SFM. This may suggest that SFM already captured a good balance of features already. Thus, it might not be beneficial to further relax the features selected by standard SFM. The entries in bold indicate the approaches with the two highest accuracies.

Table 5.4 shows the breakdown of accuracy results in terms of sensitivity and specificity in the training phase. Values in the table indicate that SFM approaches provide a very balanced classification performance for both positive and negative cases when compared to other approaches.

Table 5.5 presents the number of selected features of each classification approach in the training phase. From the table, it indicates that the SFM approaches drastically reduced the number of features to be used in the testing phase. This observation suggests a very significant medical implication. In practice, in order to make accurate diagnosis, physicians are not required to obtain and investigate so many test results and patients' information, which might be hard to obtain in some patients. On average, the SFM approaches reduce the number of features by a half while maintaining a very good classification performance, which is even better than SVM that uses all features. It is observed that the normalization in $V\text{-}r\overline{SFM}$ drastically reduces the number of features while maintaining a comparable classification performance.

After SFM classifiers are trained, they can be applied on testing data to estimate their performance. Table 5.6 shows the accuracy results of all approaches in the testing phase. The two-phase SFM models in training have the same results as standard SFM, testing them on unlabeled data will give the same results. Therefore, they are eliminated from the table. From the literature, WDBC is a clean data set on which many classification methods provide highly accurate diagnostic models. HD is not as

Table 5.3: Training Performance (in % of accuracy) from ten replications of 5-fold cross validation on four attribute data sets. The entries in bold are the top two performances.

Data	*LPSVM*	*V-SFM*	*V-rSFM*	$V\text{-}r\overline{SFM}$	*V-rSFM+*	$V\text{-}r\overline{SFM+}$
Set	*NLPSVM*	*A-SFM*	*A-rSFM*	$A\text{-}r\overline{SFM}$	*A-rSFM+*	$A\text{-}r\overline{SFM+}$
WDBC	98.08	97.28	97.66	91.57	97.28	97.28
	96.17	97.42	**98.97**	**98.97**	97.42	97.42
HD	85.06	86.48	**88.08**	81.68	86.48	86.48
	84.66	86.92	**89.98**	86.67	86.92	86.92
PID	77.66	75.01	75.02	73.71	75.01	75.01
	77.51	77.96	**79.66**	**79.64**	77.96	77.96
BLD	65.71	63.46	63.46	58.93	63.46	63.46
	57.97	66.43	**72.80**	**72.75**	66.43	66.43

A: Averaging; V: Voting

Table 5.4: Training Performance (in % of sensitivity and specificity) from 10 replications of 5-fold cross validation on 4 publicly available data sets. The entries in bold are the top two performances.

	LPSVM		*V-SFM*		*V-rSFM*		*V-r*$\overline{SFM}$		*V-rSFM+*		*V-r*$\overline{SFM+}$	
	NLPSVM		*A-SFM*		*A-rSFM*		*A-r*$\overline{SFM}$		*A-rSFM+*		*A-r*$\overline{SFM+}$	
Data set	Sens	Spec	Sens	Spec	Sens	Spec	Sens	Spec	Sens	Spec	Sens	Spec
WDBC	96.22	99.19	93.63	99.44	94.47	99.56	80.59	98.10	93.63	99.44	93.63	99.44
	91.31	99.05	93.76	99.59	**97.78**	**99.68**	**97.70**	**99.73**	93.76	99.59	93.76	99.59
HD	79.76	89.59	80.29	91.77	**84.06**	**91.51**	79.02	83.96	80.29	91.77	80.29	91.77
	78.12	90.27	79.14	93.59	**87.22**	**92.34**	85.34	87.81	79.14	93.59	79.14	93.59
PID	56.45	89.03	52.90	86.87	52.42	87.13	60.96	80.56	52.90	86.87	52.90	86.87
	55.84	89.13	58.67	88.29	**61.60**	**89.34**	**62.15**	**89.01**	58.67	88.29	58.67	88.29
BLD	90.90	30.97	61.44	66.26	61.66	65.95	91.63	13.83	61.44	66.26	61.44	66.26
	100	0	63.56	70.40	**78.43**	**65.05**	**77.86**	**65.71**	63.56	70.40	63.56	70.40

A: Averaging; V: Voting

Table 5.5: Average number of features selected by each classification approach in the training.

	LPSVM	*V-SFM*	*V-rSFM*	*V-r*$\overline{SFM}$
Data set	*NLPSVM*	*A-SFM*	*A-rSFM*	*A-r*$\overline{SFM}$
WDBC	30	11.6	16.2	5.4
	30	8.5	11.6	11.7
HD	13	7.4	9.6	2.3
	13	8.7	5.9	4.6
PID	8	4.3	4.3	2.0
	8	4.5	6.9	7.2
BLD	6	3.3	3.3	2.0
	6	3.7	5.8	5.8

Table 5.6: Testing Performance (in % of accuracy) from 10 replications of 5-fold cross validation on 4 publicly available data sets. The entries in bold are the top two performances.

	*LAD**	*LPSVM*	*V-NN*	*V-SFM*	*V-rSFM*	*V-r*$\overline{SFM}$
Data set		*NLPSVM*	*A-NN*	*A-SFM*	*A-rSFM*	*A-r*$\overline{SFM}$
WDBC	95.00**	**97.00**	91.60	94.99	95.10	90.51
		95.38	93.18	**96.01**	95.82	95.78
HD	83.40	82.96	80.87	82.49	82.45	72.86
		83.94	82.77	**84.92**	70.68	68.93
PID	77.20	**76.93**	63.14	72.75	72.92	73.38
		76.09	74.94	75.83	75.72	75.94
BLD	74.90**	**65.71**	38.38	58.20	58.03	56.78
		57.97	54.09	59.57	**63.28**	62.93

* LAD results is from [43], which applies 20 replications of 10-fold cross validation.

** The data sets have been cleaned before applying LAD.

Table 5.7: Testing Performance (in % of sensitivity and specificity) from 10 replications of 5-fold cross validation on 4 publicly available data sets. The entries in bold are the top two performances.

Data set	*LPSVM* / *NLPSVM*		*V-NN* / *A-NN*		*V-SFM* / *A-SFM*		*V-rSFM* / *A-rSFM*		*V-$\overline{rSFM}$* / *A-$\overline{rSFM}$*	
	Sens	Spec	Sens	Spec	Sens	Spec	Sens	Spec	Sens	Spec
WDBC	**94.10**	**98.71**	80.90	97.95	90.10	97.90	90.76	97.67	78.22	97.81
	90.29	98.40	82.18	99.72	**90.81**	**99.10**	92.93	97.53	92.46	97.76
HD	78.30	86.94	73.78	86.94	76.66	87.50	75.89	88.06	69.17	76.00
	76.99	**89.88**	76.79	87.88	**76.22**	**92.38**	70.92	70.44	68.25	69.50
PID	**55.63**	**88.34**	28.49	81.70	50.07	84.90	49.81	85.30	60.41	80.30
	53.34	**88.28**	51.46	87.50	55.56	86.70	56.46	86.00	56.94	86.10
BLD	**90.90**	**30.97**	13.95	72.07	57.05	59.79	57.00	59.44	90.00	10.97
	100	0	29.15	88.48	56.85	63.31	**69.00**	**55.38**	68.05	55.86

clean as WDBC, but still reasonably predictable. On the other hand, it is known that PID and BLD are very complex data sets and difficult to classify. The top two performances for each data set are highlighted in bold. The consistent results are shown in Table 5.6. Table 5.7 presents a breakdown of accuracy results in terms of sensitivity and specificity.

In general, the A-SFM achieved better classification accuracies than those of the V-SFM. The results in the testing phase show that the performances of SVM models and SFM models are very comparable while SFM models use far fewer features. Table 5.6 also includes results of another classification technique with feature selection designed especially for attribute type of medical data sets, called Logical Analysis of Data (LAD), from Hammer 2006 [43]. Except for the BLD data set, LAD and SFM models are also comparable. The standard SFM models seem to perform as well as the rSFM models, which consistently provided better training results. The normalization models of rSFM models seem to decrease the accuracies in most cases except the PID data set. This may imply that that the relaxation of SFM, which is easier to solve, can improve the classification accuracies. However, the numbers are not significant enough to draw such conclusion.

Although not in the attribute data sets, results in the next section (Section 5.5) will show that all variations of SFM models outperform SVM in the EEG MTS data sets. This may indicate that for attribute data sets, the Euclidean distance is not sufficient for SFM to measure the similarities.

5.5 SFM Classification on Multivariate Time Series Data Sets

In this section, the developed classification framework discussed in Chapter 4 is applied on the real EEG MTS data sets. Since the goal is to classify normal and pre-seizure states of EEG MTS data sets, sample correlations of pre-seizure cases between training set and testing set should be avoided. Therefore, leave-one-seizure-out cross validation is operated in order to correctly estimate the performance of the SFM classification framework.

Table 5.8: Accuracy from validation of EEG classification based on KNN, SVM and SFM classifiers with their best parameter settings.

	NN		KNN		SVM	SFM	
Patient	Acc.	Setting	Acc.	Setting	Acc.	Accuracy	Setting
1	80.96%	EU	85.71%	$k=3$,TS	71.43%	85.72%	A,EU
2	78.57%	DTW	83.33%	$k=7$,TS	78.57%	88.10%	A,DTW
3	77.28%	TS	85.61%	$k=3$,TS	85.61%	90.91%	A,TS
4	50.00%	TS	71.57%	$k=3$,TS	60.79%	64.71%	A,TS
5	52.94%	DTW	64.71%	$k=3$,TS	57.85%	59.81%	V,DTW
6	72.22%	DTW	92.60%	$k=3$,TS	79.63%	83.34%	A,DTW
7	61.60%	TS	88.41%	$k=3$,TS	70.29%	68.12%	A,TS
8	71.06%	TS	88.60%	$k=5$,TS	66.67%	74.56%	A,TS
9	54.17%	DTW	72.50%	$k=3$,TS	56.67%	60.84%	A,TS
10	66.67%	EU	83.34%	$k=5$,TS	69.45%	76.39%	V,DTW
Average	65.14%		80.94%		68.74%	73.21%	

A = averaging. V = voting. Acc. = accuracy.
EU = Euclidean. TS = T-statistics. DTW = dynamic time warping.

Table 5.8 shows performance of SFM classification on the ten EEG MTS data sets. The performance of the nearest neighbor rule is also included, which is what SFM based on without optimal feature selection. Besides, KNN and SVM classification techniques are tested on the EEG MTS data sets.

Note that the average values in the table are weighted and calculated according to the numbers of seizures of each patients. The classification performance of SFM obtained after the leave-one-seizure-out cross validation yields an accuracy of 73.21%, while without optimal feature selection, the nearest neighbor rule only gives an overall accuracy of 65.14%. The SFM feature selection and ensemble classification does improve the performance of the nearest neighbor rule. The classification performance of SVM yields an accuracy of 68.74%, which indicate that SFM outperforms SVM. The best classification performance for the EEG MTS data sets is provided by KNN rule, which results in an overall accuracy of 80.94%.

Table 5.9: Classification ability for normal and pre-seizure EEG states in terms of accuracies based on nearest neighbor (NN), KNN and SFM classifiers with best parameter settings. These values are obtained by leave-one-seizure-out and including test samples in the training phase. As a result, they are the upper bounds on discrimination accuracies for each classifier.

	NN		KNN		SFM	
Patient	Accuracy	Setting	Accuracy	Setting	Accuracy	Setting
1	80.96%	A,EU	85.72%	$k = 9$,EU	92.86%	A,EU
2	78.57%	V,EU	85.71%	$k = 9$,TS	100.00%	V,TS
3	77.28%	A,TS	85.61%	$k = 3$,TS	100.00%	A,EU
4	50.00%	A,TS	71.57%	$k = 3$,TS	88.24%	A,TS
5	52.94%	A,DTW	64.71%	$k = 3$,TS	89.22%	V,TS
6	72.22%	A,DTW	92.60%	$k = 3$,TS	98.15%	A,DTW
7	61.60%	A,TS	88.41%	$k = 3$,TS	86.96%	V,DTW
8	71.06%	A,TS	88.60%	$k = 5$,TS	100.00%	A,DTW
9	54.17%	A,DTW	72.50%	$k = 3$,TS	94.17%	A,DTW
10	66.67%	A,EU	83.34%	$k = 5$,TS	95.83%	A,DTW
Average	64.49%		81.05%		94.95%	

A = averaging. V = voting.
EU = Euclidean. TS = T-statistics. DTW = dynamic time warping.

To show how SFM can be further improved, the classification ability in terms of accuracies of SFM on each EEG MTS data set are first computed as shown in Table 5.9. For a simple comparison with other techniques, the classification abilities of the KNN classifier are also computed, as well as the nearest neighbor rule, which is a method that no training is required. These values are obtained by leave-one-seizure-out sample division and including the test samples directly to construct the classifier without consider the training phase. They are the upper bounds on discrimination accuracies for each classifier.

The results show that SFM has very high classification ability in terms of accuracy. There is a large gap between the SFM accuracy from the cross validation and its corresponding upper bounds. This means that there are large space for improving the SFM classification ability. Hence, more improvements for training the SFM model will need to be investigated. Variations of SFM optimization models may be tried on in

Table 5.10: Training and testing performance (in % of accuracy) of all classification approaches tested on the EEG data set. The entries in bold are the top two performances.

EEG Classification	*LPSVM* / *NLPSVM*	*V-NN* / *V-NN*	*V-SFM* / *A-SFM*	*V-rSFM* / *A-rSFM*	$V\text{-}r\overline{SFM}$ / $A\text{-}r\overline{SFM}$
Training	**88.89**	-	78.03	78.28	72.77
	54.14	-	79.59	**83.58**	83.57
Testing	68.74	60.35	69.39	69.06	70.81
	49.13	63.83	**72.88**	72.55	**72.98**

Table 5.11: Training and testing performance (in % of sensitivity and specificity) of all classification approaches tested on the EEG data set. The entries in bold are the top two performances.

EEG	*LPSVM* / *NLPSVM*		*V-NN* / *V-NN*		*V-SFM* / *A-SFM*		*V-rSFM* / *A-rSFM*		$V\text{-}r\overline{SFM}$ / $A\text{-}r\overline{SFM}$	
	Sens	Spec	Sens	Spec	Sens	Spec	Sens	Spec	Sens	Spec
Training	**88.24**	**89.54**	-	-	76.75	79.30	79.01	77.55	72.52	73.01
	54.03	54.25	-	-	79.02	80.15	**84.32**	**82.84**	84.02	83.12
Testing	62.75	74.73	67.10	53.59	68.41	70.37	66.01	72.11	73.86	67.76
	53.59	44.66	69.28	58.39	**71.68**	**74.07**	70.37	74.73	**67.10**	**78.87**

order to find possible improvements.

The relaxed optimization models of SFM are also tested on the EEG MTS data sets. Table 5.10 shows the performance accuracy in both training and testing phases of SFM classifiers as well as the performance accuracies of linear SVM (LPSVM), nonlinear SVM with Gaussian kernel (NLPSVM), and the nearest neighbor (NN, which is the non-optimized SFM) classifiers are included.

Lemma 4.3.1 indicate that in the training phase the relaxed models of SFM (rSFM) give at least the classification accuracies of what standard SFM does. The empirical results are consistent with the theoretical results as shown in Table 5.10. Although in training phase the SVM approach outperformed the SFM approach, they were outperformed by all the SFM variations in the testing phase (see Figure 5.5). This is mainly due to the SVM approach, which may overfit the data and the optimal hyperplanes found in the training phase did not take into account the sequence correlation between the data points of time series. The temporal properties in time series were neglected so that the SVM approach could not search for time series similarity. Consequently, the patterns found in the training phase are uncorrelated to the patterns in the testing phase. The classification performance by SVM yielded an overall sensitivity at the rate of 62.75% and an overall specificity at the rate of 74.73%. The best classification performance was achieved by A-rSFM, which yielded an overall sensitivity at the rate of 67.10% and an overall specificity at the rate of 78.87%.

Table 5.11 shows the performance in terms of sensitivity and specificity in both training and testing phases of these classifiers. It indicates that SFM approaches provide a very balanced classification performance for both positive and negative cases when compared to other approaches.

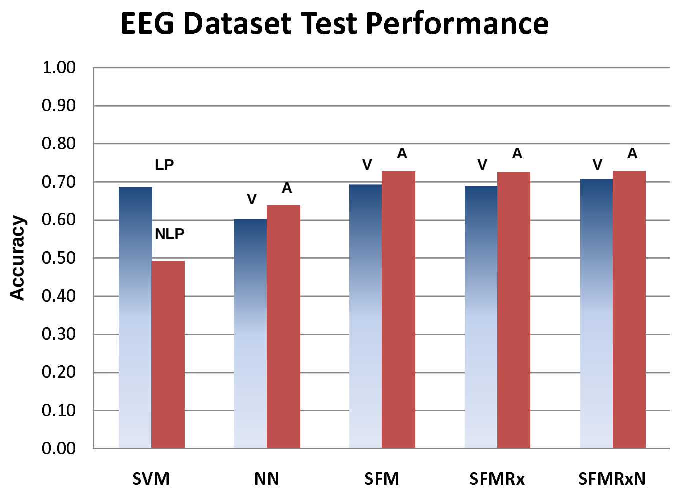


Figure 5.5: Testing performance (in % of accuracy) of all classification approaches tested on the EEG data sets.

5.6 Classification Results Using k-Nearest Neighbor SFM

Different k values, $k = 4, 8, 12, 14, 16, 20, \ldots, 100$ and all neighbors, of the KNN rule is applied on generating the accuracy matrix and the average distance matrices for SFM optimization models. Figure 5.6 to Figure 5.9 displays the training accuracies obtained from applying V-SFM and A-SFM on the four attribute data sets. Each point on a figure represent an accuracy associated with a k value used in calculating the distance matrices.

Figure 5.10 to Figure 5.13 are testing results with different k values on the four

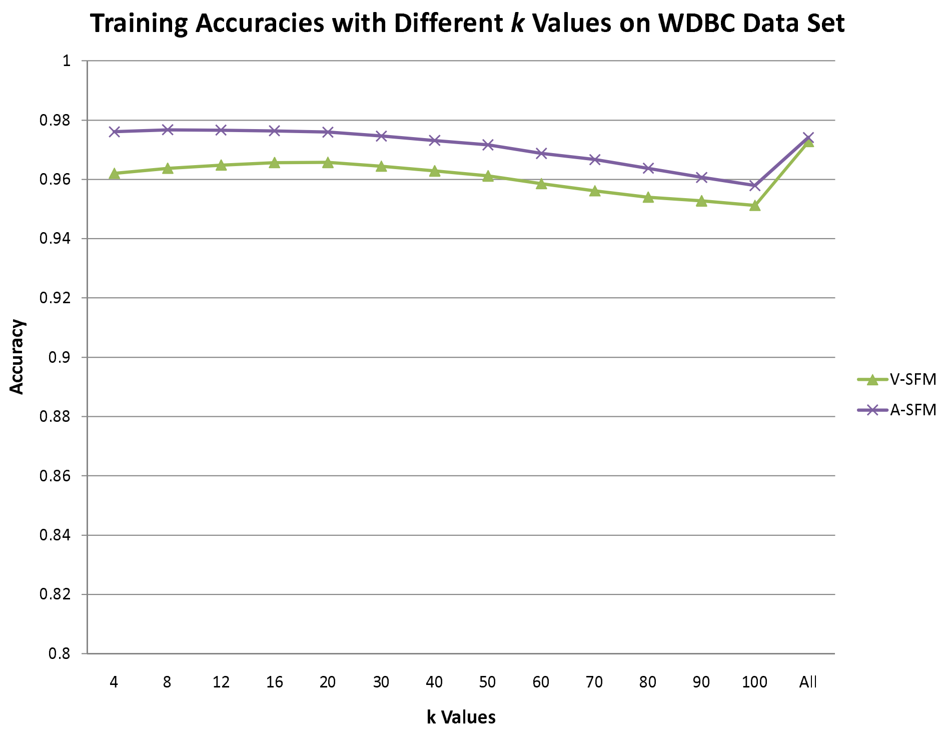


Figure 5.6: Training Accuracies of the k-Nearest-Neighbor Support Feature Machine on the Wisconsin Diagnostic Breast Cancer Data Set.

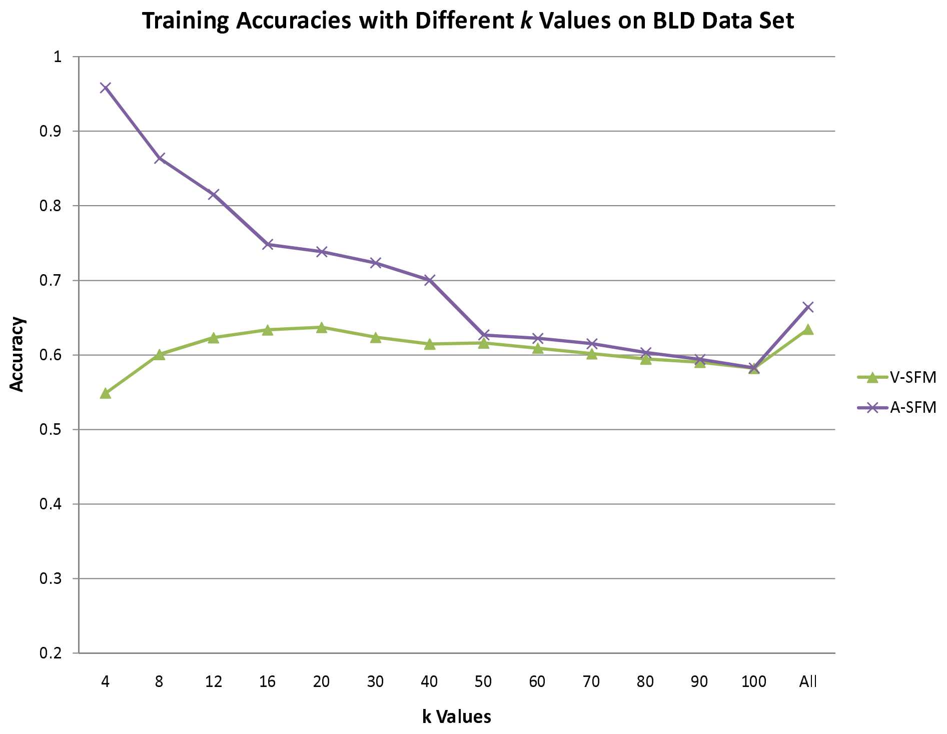


Figure 5.7: Training Accuracies of the k-Nearest-Neighbor Support Feature Machine on the BUPA Liver Disorders Data Set.

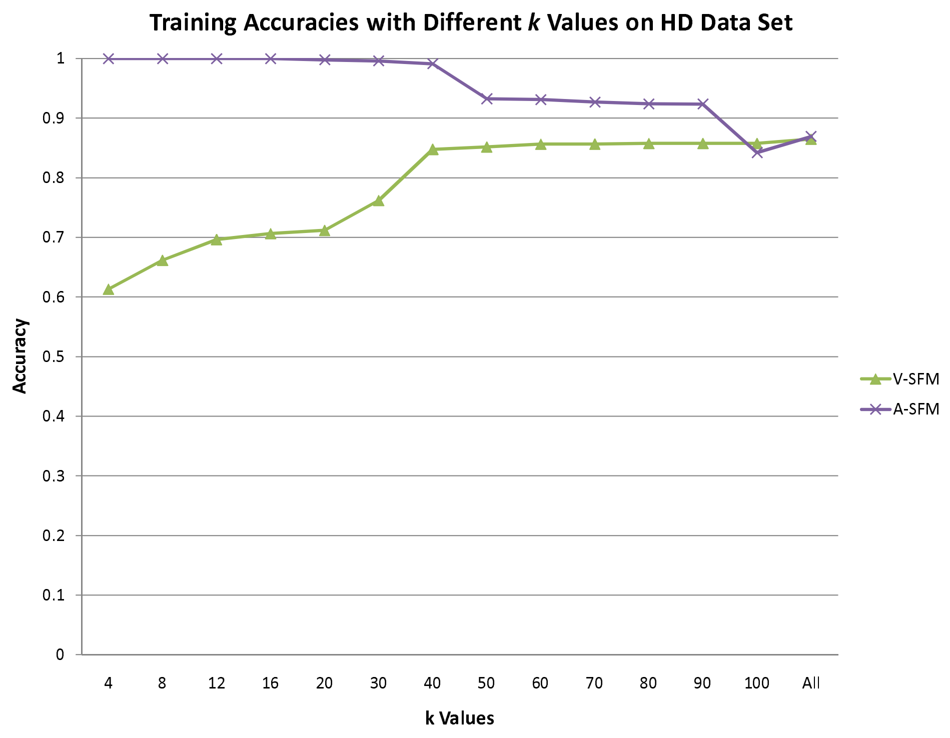


Figure 5.8: Training Accuracies of the k-Nearest-Neighbor Support Feature Machine on the Cleveland Heart Disease Data Set.

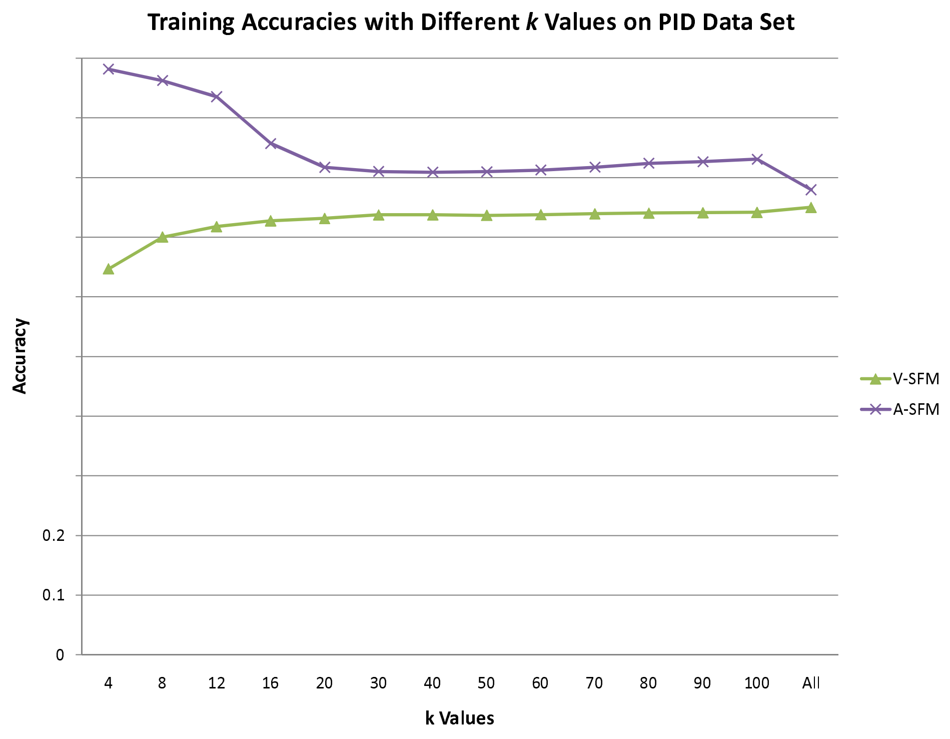


Figure 5.9: Training Accuracies of the k-Nearest-Neighbor Support Feature Machine on the Pima Indians Diabetes Data Set.

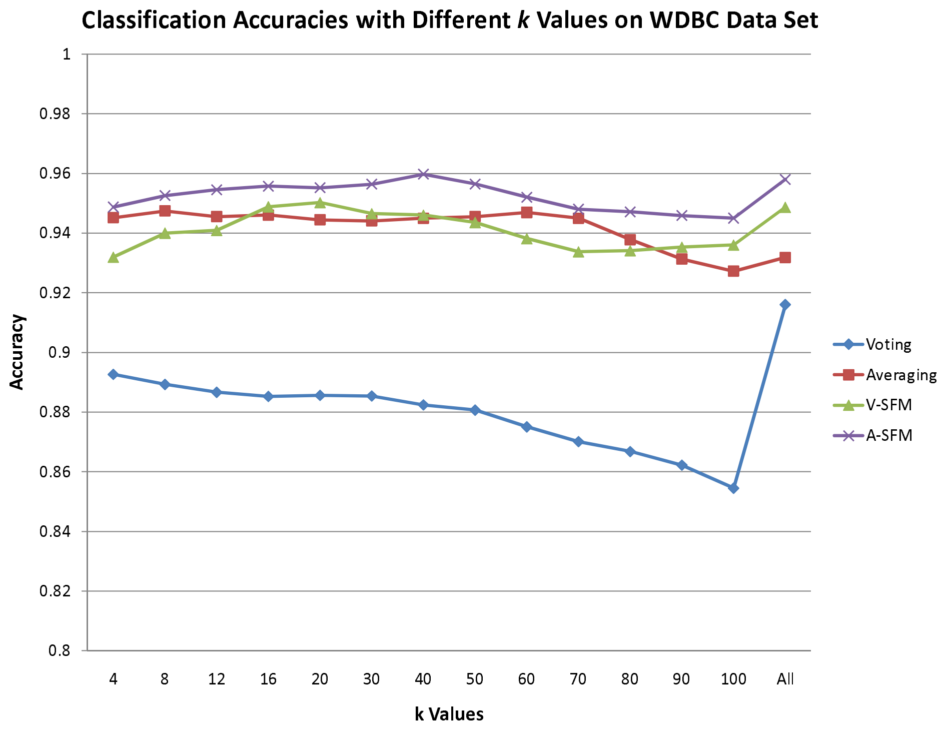


Figure 5.10: Testing Accuracies of the k-Nearest-Neighbor Support Feature Machine on the Wisconsin Diagnostic Breast Cancer Data Set.

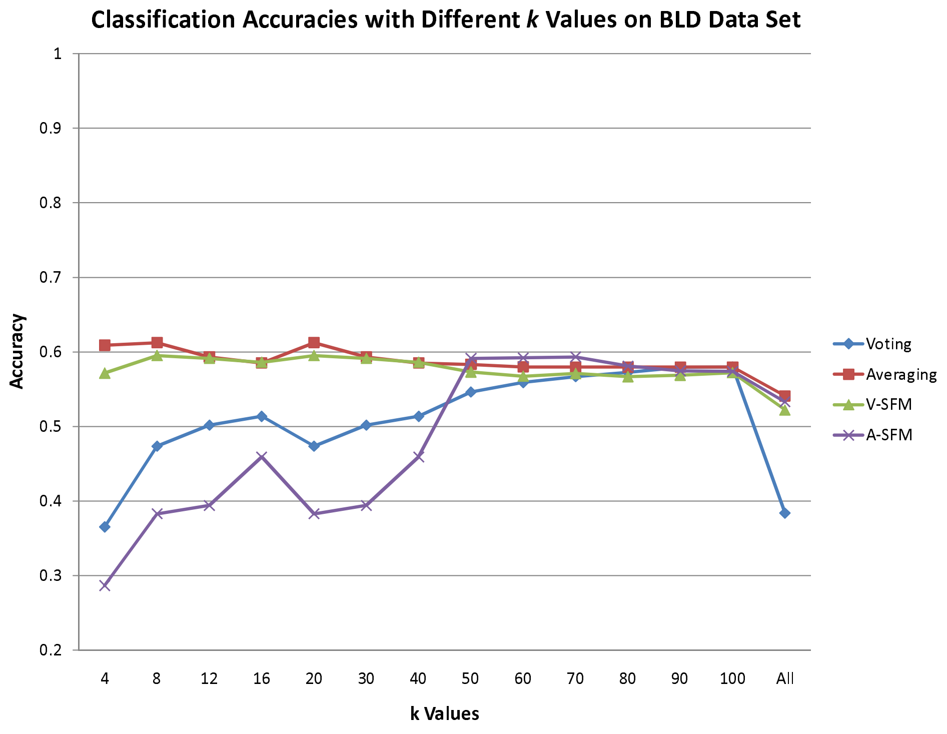


Figure 5.11: Testing Accuracies of the k-Nearest-Neighbor Support Feature Machine on the BUPA Liver Disorders Data Set.

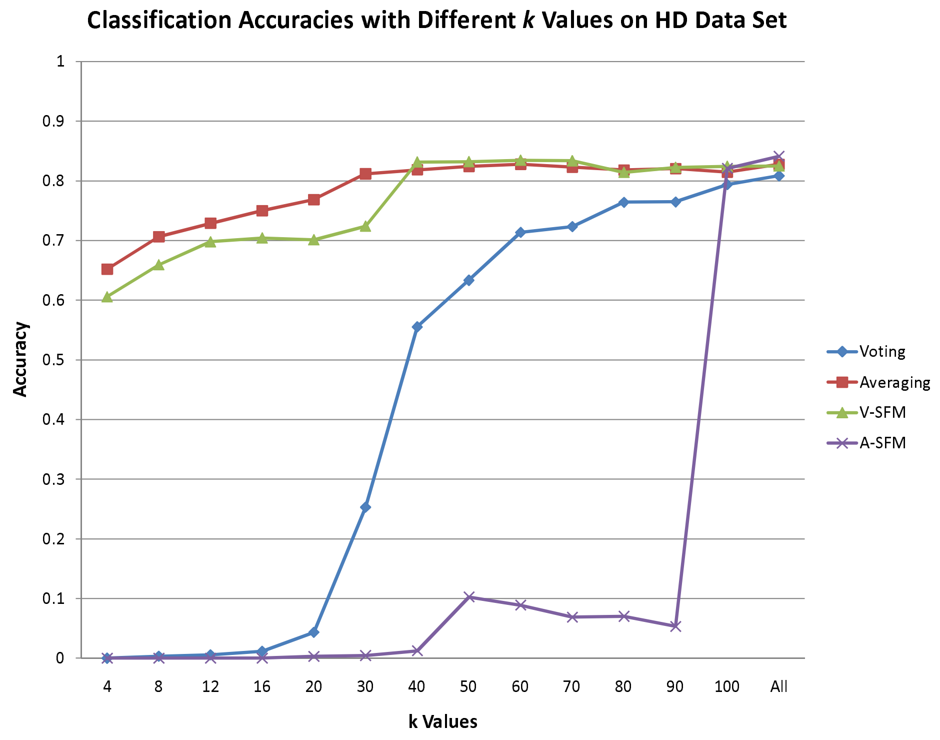


Figure 5.12: Testing Accuracies of the k-Nearest-Neighbor Support Feature Machine on the Cleveland Heart Disease Data Set.

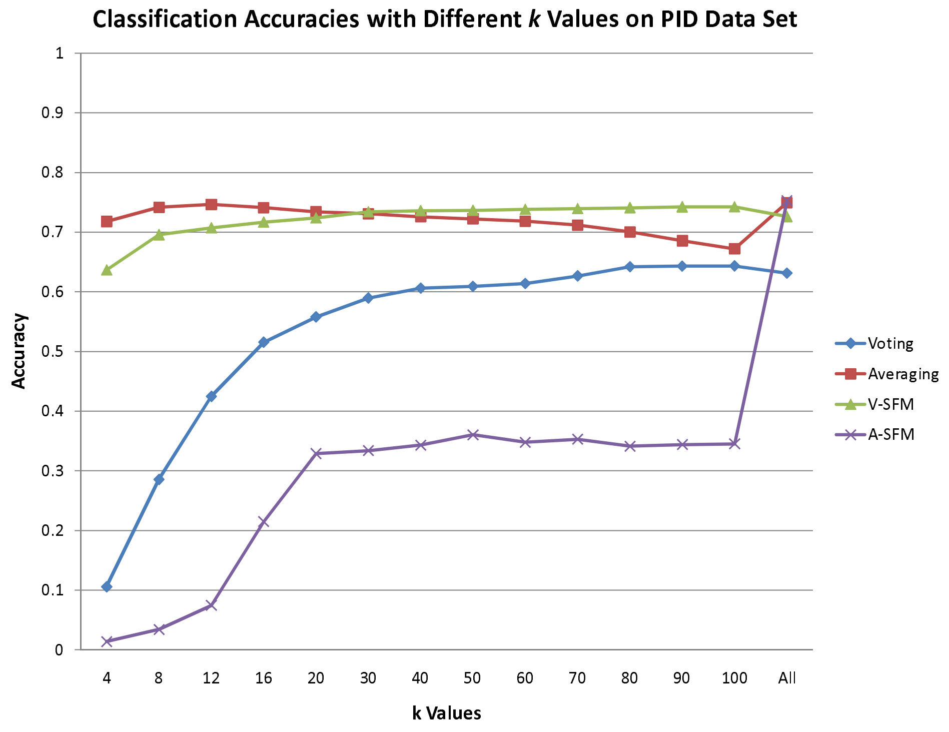


Figure 5.13: Testing Accuracies of the k-Nearest-Neighbor Support Feature Machine on the Pima Indians Diabetes Data Set.

Table 5.12: Performance (%) of sample-preserved k-median in SFM on WDBC data set using ten times of five-fold cross validation. Numbers of positive and negative training samples without sample selection are 170 and 286, respectively.

Methods	Sensitivity	Specificity	Accuracy	Best Setting	
				k_1	k_2
V-NN	80.90	97.95	91.60	N/A	N/A
A-NN	82.18	99.72	93.18	N/A	N/A
V-SFM	90.10	97.90	94.99	N/A	N/A
A-SFM	90.81	99.10	96.01	N/A	N/A
V-kMNN	87.22	96.63	93.13	3	5
A-kMNN	91.00	97.77	95.25	2	5
V-kMSFM	80.21	93.56	88.59	3	6
A-kMSFM	91.48	96.08	94.37	2	5

attribute data sets. These results indicate that using all samples is necessary for SFM to provide its best performance. In addition, the average scheme alone without SFM gives quite consistent high accuracy in the training, and is not significantly affected by the change of k values. However, the average scheme of SFM fails to classify the data when not all neighbors are included. It seems that the average scheme with optimal feature selection loses its ability to capture the difference of the average distances between the two classes in the data. It is better to use the average scheme without feature selection.

Moreover, the voting scheme alone without SFM fails to classify the data if fewer than 50 neighbors are considered in the HD data set and 20 in PID data set. However, the voting scheme of SFM gives very consistent high accuracy in both the training and testing phases. This may indicate that the voting scheme with optimal feature selection can capture the differences of the two classes in the data sets.

5.7 Classification Results Using Clustered SFM

One problem with SFM is that it carries all training samples while performing classification in the testing phase, which is very inefficient compared with SVM and most classification models. Hence, the idea of reducing the number of carried samples for SFM is to only keep most representative samples among the training samples while classifying unknown samples. The sample-preserved k-median (SPKM) clustering may be helpful in choosing such training samples.

Table 5.13: Performance (%) of sample-preserved k-median in SFM on HD data set using ten times of five-fold cross validation. Numbers of positive and negative training samples without sample selection are 110 and 128, respectively.

Methods	Sensitivity	Specificity	Accuracy	Best Setting	
				k_1	k_2
V-NN	73.78	86.94	80.87	N/A	N/A
A-NN	76.79	87.88	82.77	N/A	N/A
V-SFM	76.66	87.50	82.49	N/A	N/A
A-SFM	76.22	92.38	84.92	N/A	N/A
V-kMNN	83.25	80.00	81.50	6	5
A-kMNN	83.94	85.00	84.53	6	6
V-kMSFM	77.33	76.88	77.10	4	6
A-kMSFM	72.96	81.88	77.77	2	5

Table 5.14: Performance (%) of sample-preserved k-median in SFM on PID data set using ten times of five-fold cross validation. Numbers of positive and negative training samples without sample selection are 214 and 400, respectively.

Methods	Sensitivity	Specificity	Accuracy	Best Setting	
				k_1	k_2
V-NN	28.49	81.70	63.14	N/A	N/A
A-NN	51.46	87.50	74.94	N/A	N/A
V-SFM	50.07	84.90	72.75	N/A	N/A
A-SFM	55.56	86.70	75.83	N/A	N/A
V-kMNN	43.27	74.40	63.54	6	4
A-kMNN	53.03	86.40	74.74	5	3
V-kMSFM	72.33	72.60	72.53	2	4
A-kMSFM	69.04	74.40	72.53	3	4

Table 5.15: Performance (%) of sample-preserved k-median in SFM on BLD data set using ten times of five-fold cross validation. Numbers of positive and negative training samples without sample selection are 160 and 116, respectively.

Methods	Sensitivity	Specificity	Accuracy	Best Setting	
				k_1	k_2
V-NN	13.95	72.07	38.38	N/A	N/A
A-NN	29.15	88.48	54.09	N/A	N/A
V-SFM	57.05	59.79	58.20	N/A	N/A
A-SFM	56.85	63.31	59.57	N/A	N/A
V-kMNN	58.50	26.21	44.93	2	5
A-kMNN	71.00	42.07	58.84	2	6
V-kMSFM	61.00	46.90	55.07	2	6
A-kMSFM	68.50	39.31	56.23	5	3

Table 5.12 to Table 5.15 shows the performance on the four attribute data sets. It is clear that adapting the sample selection technique does not improve the classification accuracies in all four data sets. Only nearest neighbor rule is improved. After decreasing the number of training samples for SFM and shrinking the number of features by SFM, the classification accuracies are reduced. SFM combining with chosen samples failed to classify. One major reason could be that the number of training samples is reduced while the features are shrank, which gives too few information for both the voting and averaging schemes of nearest-neighbor rule to distinguish new samples. However, the improved performance on the nearest neighbor rule may indicate that the SPKM clustering gives good selection of training samples.

The results motivates the new feature selection algorithms in the feature space SPKM clustering that are proposed in this dissertation, discussed in Section 6.3.1.

Chapter 6

OPTIMIZATION MODELS FOR MULTIVARIATE DATA CLUSTERING

Clustering is an unsupervised method for data analysis, in which the group labels of data points are not known during the training of clusters. It is concerned with partitioning data into k disjoint subsets, called clusters, such that the data points are more "similar" among each other within a cluster than to data points in other clusters. Appropriate measures of "similarity" between two data points may depend on the underlying characteristics of the data set itself. One may employ the 1-norm or Euclidean distance measures for data in a metric space [15]. One may use the dynamic time warping distance for time series data [8, 53].

Given a data set of $\mathbb{R}^{n \times m}$ and an integer k. The goal of the k-median clustering algorithm is to determine k cluster centers in m-dimensional real space $\mathbb{R}^m$ so that each of the n sample points is assigned to one of the k clusters, in a way that the sum of the distances of each point to the nearest cluster center is minimized. These k cluster centers are called medians. The k-means clustering is similar. The objective of k-means clustering is to minimize the sum of squared Euclidean distances from each point to its cluster center.

Solution ideas used for k-means problems can usually be applied to the k-median problems [1, 15]. Many efficient algorithms designed for k-group clustering problems require a "guess" of initial cluster centers, and then try to adjust cluster centers iteratively until they converge to a local minimum. The most commonly used technique to solve the k-means in practice is developed by Lloyd (1982) [63]. The Lloyd's algorithm is based on the observation that the optimal location of a center is at the centroid of the associated cluster [35]. At initial stage, a starting solution is randomly chosen. Then the

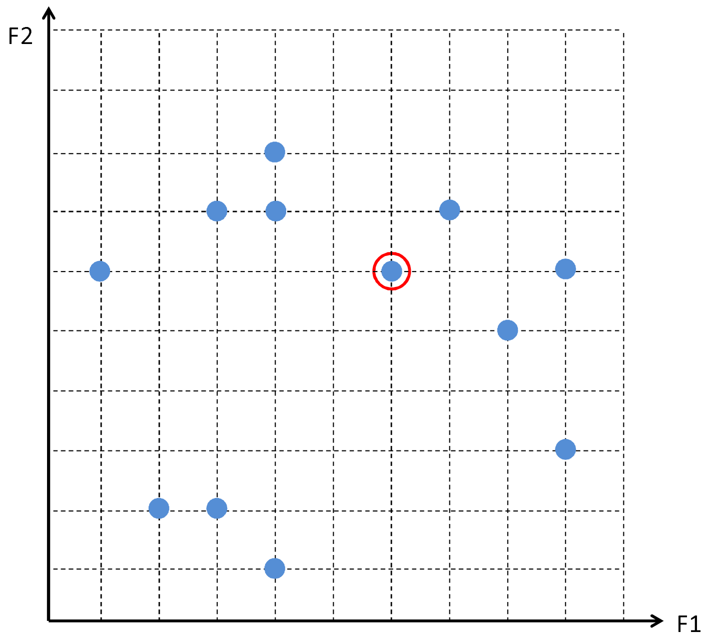


Figure 6.1: An example of a sample-preserved median in a cluster.

algorithm determines new clusters and moves every center to the centroid of a cluster at each stage until some convergence condition is met. A filtering algorithm proposed in [52] gives a simple and efficient implementation of Lloyd's algorithm. Methods in [15] obtains the k medians by an algorithm that solves a concave optimization problem, which also requires an initial guess of centers. Studies in [56] have found that a good cluster center initialization can improve such algorithms. In [62], a global k-means method especially involves an algorithm for choosing the initial clustering centers. A technique in [17] refines initial points based on an efficient technique for estimating the modes of a distribution, which allows the iterative algorithm to converge to a "better" local optimal solution.

The k-median problem is also similar to the uncapacitated facility location problem in its goal [24]. It is usually known as a p-center location problem or node selection problem in a graph [25]. In this dissertation, such a k-median method is called the sample-preserved k-median (SPKM) method. Figure 6.1 is an example of a sample-preserved median in a cluster. The highlighted red circle points to the sample that has the smallest sum of the Euclidean distances to all other samples in the cluster. The sample-preserved property is critical especially when not all values are valid in some application domain. In such a case, a cluster median is better to be an existing sample in order to contain valid values. The original k-median as shown in [15] chooses a medium value at each of the m dimension. Those medians may not be any of the existing samples, which are not sample-preserved. Instead, SPKM method chooses centers from existing samples by considering the distances between samples in all the m dimensions simultaneously. By formulating the k-median problem as a facility location problem, one can benefit from its flexibility of choosing a distance measure. The cost matrix used in facility location can be taken as a distance matrix for the SPKM method. Since the distance matrix is pre-calculated, the model dose not depend on the choice of distance measures. Any applicable similarity measure can be used in the SPKM method.

Similarity measures have been largely applied on analysis of time series data sets [31, 91, 54]. Therefore, clustering techniques incorporating variant distance measures may be helpful for the analysis of time series data. SPKM method has this property. The SPKM method that can incorporate arbitrary time series similarity measures, it is formulated as an integer programming (IP) problem that is $\mathcal{NP}$-hard [23]. Hence, approximation algorithms have been developed for SPKM. Greedy local-search based solutions can be found in [23, 38]. There are also techniques based on a linear programming relaxation proposed in [68], a hierarchically greedy approach proposed in [72], and an approximation derived from a primal-dual-based algorithm with the use of Lagrangian relaxation proposed in [49].

Although approximation algorithms are developed for the SPKM problem, their formulations are not as simple and easy as the method proposed for the original (non-sample-preserved) k-median problem in [15]. Its formulation is a dual minimization problem, and it comes with a bilinear program algorithm that efficiently approximates the solution for the original k-median problem. Only two simple linear programs are needed and the algorithm alternately solves them until a local optimal solution is found. However, the original k-median method is not sample-preserved. Figure 6.2 is an example of an original median in a cluster. All samples are projected onto each of the feature spaces and the resulting median is a combination of median values from each of the feature spaces. Those values do not necessarily come from existing sample values.

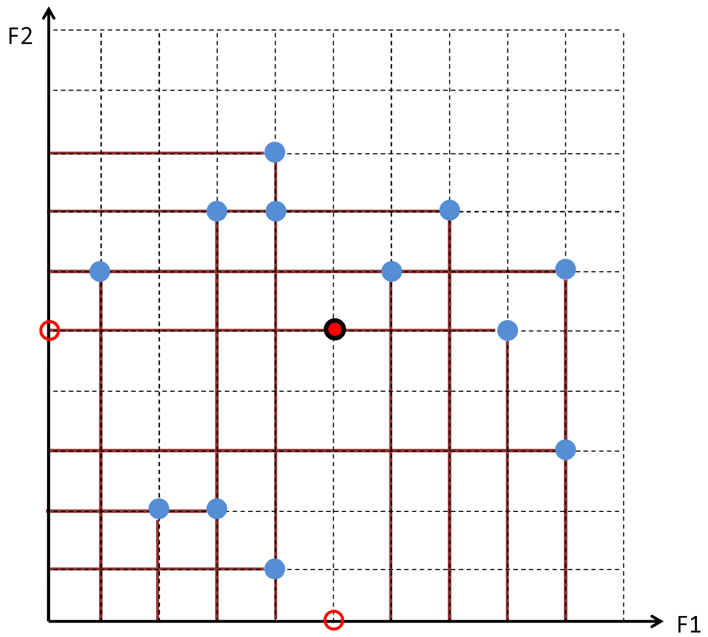


Figure 6.2: An example of the original median in a cluster.

Observe also that it does not allow a flexible choice of distance measure. Only the 1-norm distance can be used in order to find a fast efficient solution. If the 2-norm is applied, the problem results in many local minima and becomes a considerably harder problem [15]. Hence, it can hardly be applied on time series data.

Therefore, an efficient algorithm for solving the SPKM clustering problem is needed. In this Chapter, two new k-median algorithms are proposed to satisfy such need. They are SPKM methods, and each is formulated as a 0-1 integer programming problem. The first method is called the Bilinear Program Sample-Preserved k-Median (BPSPKM) method, which applies a sample-to-sample distance matrix in a "whole" m-dimensional space. The second method is called the Feature Space Sample-Preserved k-Median (FSSPKM) method, which adapts a sample-to-sample distance matrix at "each" of the m feature spaces.

Both method can be solved by the same idea of the bilinear program algorithm used for solving the original k-median method in [7, 15]. The proposed bilinear program SPKM algorithm contains two linear programs that give binary solutions under a condition that is easy to be met. The SPKM solution can be approximated by solving the two programs alternately until the algorithm converges to a local optimal.

The bilinear program SPKM algorithm generalizes the original k-median algorithm to accommodate different distance measures using linear programming formulations. Furthermore, equivalent SPKM algorithms without using any linear programs are proposed for both BPSPKM and FSSPKM, which is even more efficient to use.

Designing especially for the original k-median algorithm in [15], Mangasarian (2004) [71] proposed a feature selection clustering algorithm. The proposed feature selection technique shows possible improvements of classification by using smaller number of features. Therefore, feature selection criteria tailor made for the FSSPKM clustering are proposed, which utilize the distance matrix at each feature space for the revised k-median technique so that the sample-preserved property is kept.

Throughout this chapter, all vectors are column vectors. Suppose a data set contains n samples, and each has m features. Let $i, j \in \{1, \ldots, n\}$, $s \in \{1, \ldots, m\}$, $p \in \{1, \ldots, k\}$. If A is a matrix, A_i is a vector which denotes the ith row of A; and A_{ij} is a value of the ith row and jth column of A. Let G_p denote the set of indices of samples that are assigned to cluster p for $p = 1, \ldots, k$. That is $G_p \subset \{1, 2, \ldots, n\}$ and $G_p \cap G_{p'} = \emptyset$ for joint clustered subsets $p \neq p'$ for p and $p' \in \{1, \ldots, k\}$. $G_1 \cup G_2 \cup \cdots \cup G_k = \{1, 2, \ldots, n\}$.

6.1 Bilinear Program Sample-Preserved k-Median (BPSPKM) Clustering

In this section, a bilinear program sample-preserved k-median (BPSPKM) clustering algorithm is proposed that generalizes the original k-median algorithm to adapt different distance measures using LP formulations. BPSPKM is an algorithm that provides local optimal solutions to the IP problem of the SPKM clustering.

All cluster centers obtained by the SPKM method are existing samples, thus the "shape" of time series is retained in those cluster centers. However, the SPKM method is formulated in IP problem, which is not easy to solve. Therefore, a BPSPKM clustering algorithm is proposed that contains two LP problems. The algorithm adapts a precalculated distance matrix that forms a sample-preserved model and can then be solved by the bilinear program algorithm, which is used to solve the original k-median method easily and efficiently as shown in [7, 15]. By this way, the proposed BPSPKM keeps the major benefits of the two existing k-median clustering methods. It has the flexibility of choosing a distance measure as what the SPKM method has due to its sample-preserved property and the incorporated distance matrix. Hence, it can be easily applied on time series type of data. It can also avoid the problem of having median values not valid in some application domains. Having sample-preserved medians can guarantee that the obtained center values are valid. Moreover, it has the efficiency in obtaining an approximated solution as the original k-median method proposed in [15] due to the two linear programs in its model.

Let D denote a distance matrix whose element D_{ij} represents the distance from sample i to sample j for $i, j = 1, \ldots, n$. The objective is to find k samples as cluster medians so that the sum of the distances from all samples to their closest cluster medians is minimized. A decision variable $\bar{X} \in \{0, 1\}^{n \times k}$ is used to determine the optimal cluster medians. If $\bar{X}_{ip} = 1$, sample i is determined to be a cluster median of cluster p, for $i = 1, \ldots, n$ and $p = 1, \ldots, k$. Otherwise, $\bar{X}_{ip} = 0$. Let X be the relaxation of the binary variable $\bar{X}$. It will be shown that the proposed algorithm gives binary solutions by solving the relaxation of the integer program under a certain condition.

Program (6.1) is reformulated as a dual minimization problem. In the optimal solution to the inner minimization, the minimum distances from each sample to each of the k cluster centers are found, which can be viewed as cluster assignment. The overall objective function tries to find the optimal cluster centers such that the sum of the distances from all samples to their nearest cluster medians is minimized. The overall solution to Program (6.1) provides the optimal samples that represent the k cluster medians.

$$
\begin{aligned}
\min_{X} \quad & \sum_{i=1}^{n} \min_{p=1,\ldots,k} \{\sum_{j=1}^{n} D_{ij} X_{jp}\} \\
\text{s.t.} \quad & \\
& \sum_{i=1}^{n} X_{ip} = 1 && p = 1,\ldots,k \\
& \sum_{p=1}^{k} X_{ip} \leq 1 && i = 1,\ldots,n \\
& 0 \leq X_{ip} \leq 1 && i = 1,\ldots,n, p = 1,\ldots,k.
\end{aligned}
\tag{6.1}
$$

To solve the optimization problem in Program (6.1), the objective function is reduced into a single minimization problem. The idea of Lemma 2.1 in [7] (Lemma 2.4.1) is applied and another equivalent optimization problem is formed with a single nonlinear objective function and linear constraints as shown in Program (6.2).

Lemma 2.4.1 indicates that solving Program (6.1) is equivalent to solving Program (6.2). The objective function (6.2a) contains the element-by-element multiplication of two decision variables X_{ip} and T_{ip} for $i = 1,\ldots,n$ and $p = 1,\ldots,k$. Therefore, to satisfy the equivalent condition stated in Lemma 2.4.1, Constraints (6.2b), (6.2c) and (6.2d) are included in the formulation.

$$
\min_{X,T} \quad \sum_{i=1}^{n} \sum_{p=1}^{k} \left\{ \sum_{j=1}^{n} D_{ij} X_{jp} \right\} T_{ip} \tag{6.2a}
$$

s.t.

$$
\sum_{p=1}^{k} X_{ip} \leq 1 \qquad i = 1,\ldots,n
$$

$$
0 \leq X_{ip} \leq 1 \qquad \text{for } i = 1,\ldots,n, p = 1,\ldots,k
$$

$$
\sum_{p=1}^{k} T_{ip} = 1 \qquad \text{for } i = 1,\ldots,k \tag{6.2b}
$$

$$
\sum_{i=1}^{n} X_{ip} T_{ip} = 1 \qquad \text{for } p = 1,\ldots,k \tag{6.2c}
$$

$$
T_{ip} \geq 0 \qquad \text{for } i = 1,\ldots,n; p = 1,\ldots,k. \tag{6.2d}
$$

6.1.1 Uncoupled Bilinear Program Algorithm

The concept of Uncoupled Bilinear Program Algorithm (UBPA) proposed in [7] can be applied to solve this nonlinear optimization problem. The algorithm contains two major LP formulations. One is for cluster assignment, and the other is for cluster center update.

1. **LP Formulation for Cluster Assignment:**

$$\begin{aligned}
&\min_{T} && \sum_{i=1}^{n}\sum_{p=1}^{k}\left\{\sum_{j=1}^{n} D_{ij}X_{jp}\right\}T_{ip} \\
&\text{s.t.} && \\
& && \sum_{p=1}^{k} T_{ip} = 1, \qquad && i = 1,\dots,n \\
& && T_{ip} \geq 0, \qquad && i = 1,\dots,n; p = 1,\dots,k.
\end{aligned} \tag{6.3}$$

2. **LP Formulation for Cluster Median Update:**

$$\min_{X} \sum_{i=1}^{n}\sum_{p=1}^{k}\left\{\sum_{j=1}^{n} D_{ij}X_{jp}\right\}T_{ip} \tag{6.4a}$$

s.t.

$$\sum_{p=1}^{k} X_{ip} \leq 1 \qquad i = 1,\dots,n \tag{6.4b}$$

$$\sum_{i=1}^{n} X_{ip}T_{ip} = 1 \qquad p = 1,\dots,k \tag{6.4c}$$

$$0 \leq X_{ip} \leq 1, \qquad i = 1,\dots,n; p = 1,\dots,k.$$

Once the two major LP problems are formulated, local optimal solutions to Program (6.2) can be found by starting with a set of initial cluster centers and alternately solving the two linear programs, as shown in Algorithm 6.1.1. Both Program (6.3) and Program (6.4) can produce binary solutions of X_{ip} and T_{ip} for $i = 1,\dots,n$ and $p = 1,\dots,k$.

Algorithm 6.1.1 Bilinear Program Sample-Preserved k-Median Clustering Algorithm

```
Input: $k$ = number of clusters; and $X^0$, an initial decision of $k$ samples as cluster medians.

Set $t = 0$ and solve the two linear programs alternatively.

    1. Cluster Assignment. Given $X^t$, find the solution $T^t$ of Program (6.3).
    2. Cluster Median Update. Given $T^t$ from the optimal solution of Program (6.3), update $X^{t+1}$ by solving Program (6.4).

Stop when $\sum_{p=1}^{k} X_{ip}^{t} = \sum_{p=1}^{k} X_{ip}^{t+1}$ for $i = 1, \ldots, n$. Set $X^* = X^{t+1}$ and $T^* = T^t$. Otherwise, increment $t$ by one and continue to Step 1.

Output: A decision of cluster medians, $X^*$, and cluster assignment, $T^*$.
```

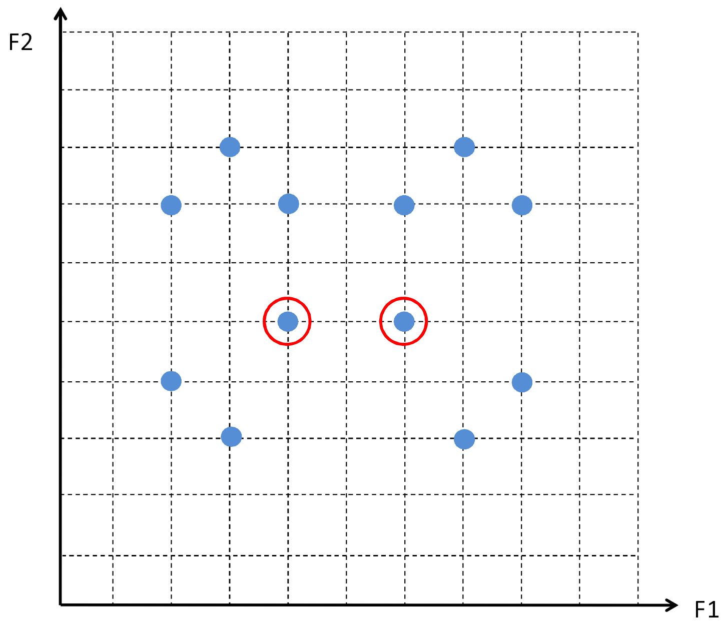


Figure 6.3: An example of a violation of the uniqueness.

Proposition 6.1.1 confirms that Program (6.3) produces binary solutions. Proposition 6.1.2 states a condition for Program (6.4) to produce binary solutions. It is to assume that the sum of distances from sample i to all other samples in the same cluster p is "unique" within that cluster for all $i \in G_p$. This assumption is not difficult to achieve. Only in a very rare case such that all samples in a cluster are symmetric with possible cluster centers, the sum of distances to other samples can be equal for at least two separate samples. An example can be found in Figure 6.3, where two circled samples have the equal sum of the distances to all other samples in the cluster. In such cases, fractional solutions may occur. Even if the assumption is not satisfied, the samples with fractional solutions would indicate that any of them can equally be a cluster median. The fractional solution is due to the equal minimum sum of distances from the samples with the fractional solution to all other samples in the cluster. Hence, any one of such samples can be taken as a solution. With these two propositions, one can obtain local optimal solutions of SPKM without solving the IP problem. Only the two simple linear programs are needed.

Algorithm 6.1.1 converges to a local optimal solution, according to a similar idea of the finite termination results proposed in [14], which is given in Proposition 6.1.3.

Proposition 6.1.1. *There exists an optimal solution of Program (6.3) such that* $T \in \{0,1\}^{n\times k}$.

Proof. The constraint matrix in Program (6.3) is totally unimodular. By Proposition 3.3 in [95], the linear program (6.3) has integral optimal solution. Since T_{ip} is bounded by 0 and 1, for $i = 1, \ldots, n$ and $p = 1, \ldots, k$, Program (6.3) has a binary optimal solution. □

Proposition 6.1.2. *Let* $\Theta_{ip} = \sum_{j=1}^{n} D_{ij} T_{jp}$ *for* $i = 1, \ldots, n$ *and* $p = 1, \ldots, k$. *For each* p, *if every element of* Θ_{ip} *for all* $i \in G_p$ *is unique, then the linear programming problem of Program (6.4) gives binary solutions of* X.

Proof. It can be proved by contradiction. Recall that Θ_{jp} is the sum of distances from sample j to all other samples in the same cluster p. Constraint (6.4b) ensures that each sample can be a cluster center for at most one cluster. Constraint (6.4c) indicates that for a given p, exactly one of the samples that are assigned to cluster p is a cluster center. Objective (6.4a) finds the k minimum values of Θ_{jp} for all $j \in G_p$ where $p = 1, \ldots, k$.

Suppose that for each p, every element of Θ_{jp} for all $j \in G_p$ is unique. Also, suppose that an optimal solution of Problem (6.4) contains X^*_{ip} and $X^*_{i'p}$, $i, i' \in G_p$, that are fractional for a given p such that $X^*_{ip}T_{ip} + X^*_{i'p}T_{i'p} = 1$ and $X^*_{ip} + X^*_{i'p} = 1$ due to Constraint (6.4c). Then the term $X^*_{ip}\Theta_{ip} + X^*_{i'p}\Theta_{i'p}$ in the objective is the minimum among any $\sum\limits_{j \in G_p} X_{jp}\Theta_{jp}$ formed by feasible values of $X'_{jp}s$ corresponding to a given p. Since every element of Θ_{ip} for all $i \in G_p$ is unique for a given p, without loss of generality, assume that $\Theta_{ip} < \Theta_{i'p}$, which implies that there exists a minimum value of Θ_{jp} for all $j \in G_p$.

Then $\Theta_{ip} = X^*_{ip}\Theta_{ip} + X^*_{i'p})\Theta_{ip} < X^*_{ip}\Theta_{ip} + X^*_{i'p}\Theta_{i'p}$, since $X^*_{ip} + X^*_{i'p} = 1$ and $\Theta_{ip} < \Theta_{i'p}$. There exist a solution $X^{**}_{ip} = 1$ such that the term $X^{**}_{ip}\Theta_{ip}$ minimizes the objective function for the given p. This contradicts the fact that $X^*_{ip}\Theta_{ip} + X^*_{i'p}\Theta_{i'p}$ must be the minimum among all feasible solution of $X'_{jp}s$. Hence, the optimal solutions of X_{ip} are binary if for each p, every element of Θ_{ip} for all $i \in G_p$ is unique. □

Proposition 6.1.3. *The Bilinear Program Sample-Preserved k-Median Clustering Algorithm (Algorithm 6.1.1) terminates in a finite number of iterations at a cluster assignment that is locally optimal.*

Proof. It needs to be shown that the objective value of Program (6.2) can not be decreased by either reassignment of a sample to a different cluster, or by deciding a new cluster median for any of the clusters. At each iteration, in the step for cluster assignment, each sample is assigned to a closest median. Thus, the objective value of Program (6.2) cannot be increased. In the step for cluster median update, each cluster median is recomputed in order to minimize the sum of all distances from each sample in the cluster to its cluster median. Hence, the overall objective value can be either strictly decreased or can stay the same as the algorithm terminates.

It can be concluded that the algorithm must terminate at some clustering assignment that is locally optimal, because the algorithm does not repeat assignments by the stopping criterion, the overall objective function is non-increasing and bounded below by zero, and there is a finite number of ways to assign n samples to k clusters. $\square$

6.1.2 Equivalent Sequential Search Algorithm without LP formulations

By introducing two matrices Δ and Θ, an equivalent algorithm to Algorithm 6.1.1 can be formulated without the two linear programs. It is a non-LP sequential search SPKM algorithm shown in Algorithm 6.1.2.

In the step of cluster assignment, a sample is assigned to the closest cluster median. Let $\Delta = D \cdot X$, then Δ_{ip} represents distance from sample i to the cluster median p. Sample i is then assigned to a closest cluster median p, which has the minimum of Δ_{ip} among all $p \in \{1, \dots, k\}$. This solution is the same as the exact solution to Program (6.3).

Similarly, in the step of cluster median update, according to assigned samples in a cluster, a cluster median in that cluster is a sample, which has the smallest distance to all other samples in the cluster. Let $\Theta = D \cdot T$. Then for a given p, Θ_{ip} represents the sum of the distances from sample i to all other samples in the same cluster p where $i \in G_p$. Hence, the cluster median is the sample that has the minimum of Θ_{ip} among all $i \in G_p$. As in Proposition 6.1.2, for each p, if every element of Θ_{ip} for all $i \in G_p$ is unique, then the solution of cluster medians from Algorithm 6.1.2 is the same as the one from Program (6.4). If the uniqueness condition is not satisfied, any sample i having the same minimum Θ_{ip} as other samples in the cluster p can be taken as a solution. In this case, Algorithm 6.1.2 can also pick the same solution as Program (6.4).

Since the same idea is used in Algorithm 6.1.1, Proposition 6.1.3 also applies to Algorithm 6.1.2 that in finite number of steps it will converge to a local optimal solution.

Algorithm 6.1.2 (Non-LP) Sequential Search Sample-Preserved k-Median Clustering Algorithm

Input: k = number of clusters; X^0_{ip}, an initial decision of k samples as cluster medians, for $i = 1, \ldots, n$ and $p = 1, \ldots, k$; and D_{ij}, a distance matrix for $i, j = 1, \ldots, n$.

Set $t = 0$ and solve the following two problems alternatively.

1. **Cluster Assignment.** Compute $\Delta = D \cdot X^t$. Then Δ_{ip} represents distance from sample i to the cluster center p. Find the closest median p^* to each sample i.

$$p^* = \arg \min_{p=1,\ldots,k} \Delta_{ip} \quad \text{for each } i = 1, \ldots, n.$$

   Set $T^t_{ip^*} = 1$ and all other $T^t_{ip} = 0$ for $p = 1, \ldots, k \setminus p^*$, for each $i = 1, \ldots, n$.

2. **Cluster Median Update.** Compute $\Theta = D \cdot T^t$. Then Θ_{ip} represents the sum of distances from sample i to all other samples in the same cluster p. For each cluster p, find the sample i^* which has the smallest sum of distances to all other samples in the same cluster.

$$i^* = \arg \min_{i \in G_p} \Theta_{ip} \quad \text{for each } p = 1, \ldots, k.$$

   Set $X^{t+1}_{i^*p} = 1$ and all other $X^{t+1}_{ip} = 0$ for $i = 1, \ldots, n \setminus i^*$, for each $p = 1, \ldots, k$.

Stop when $\sum_{p=1}^k X^t_{ip} = \sum_{p=1}^k X^{t+1}_{ip}$ for all i. Output final decision $X^* = X^{t+1}$. Otherwise, increment t by one and continue to solve the two problems.

Output: A decision of cluster median, X^*.

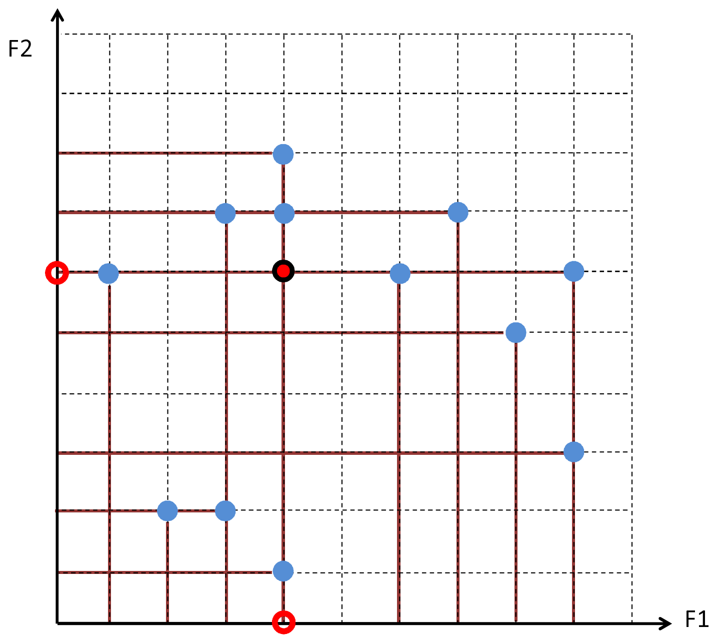


Figure 6.4: An example of the feature space sample-preserved median in a cluster.

6.2 Feature Space Sample-Preserved k-Median (FSSPKM) Clustering

The Bilinear Program Sample-Preserved k-median (BPSPKM) method applies a sample-to-sample distance matrix in a "whole" m-dimensional space. Extending from the BP-SPKM algorithm, the Feature Space Sample-Preserved k-median (FSSPKM) method adapts a sample-to-sample distance matrix at "each" of the m dimensions. The objective of FSSPKM is to find k samples at each of the m feature spaces, and take them as cluster medians so that the sum of all distances from all samples to their closest cluster medians is minimized. Figure 6.4 displays an example of the feature space sample-preserved median in a cluster. All sample values are projected onto each of the feature spaces. Then a sample is chosen as a median of each feature space. The resulting feature space sample-preserved median is a combination of those median values from each of the feature spaces. It is similar to the original k-median clustering as shown in [15], which chooses a medium value at each of the m feature spaces. However, those medians obtained from the original k-median clustering may not be any of the existing samples, which are not sample-preserved. The feature space sample-preserved medians can avoid the problem of having values that are not valid in some application domain. It guarantees that the chosen values do exist in each feature space. Figure 6.5 displays

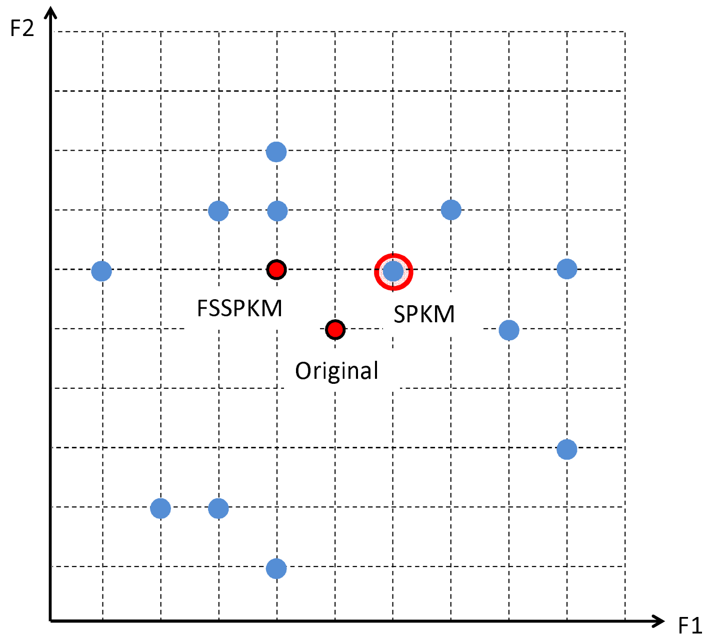


Figure 6.5: The feature space sample-preserved median, the sample-preserved median and the original median in a cluster.

the three medians in a cluster obtained according to the SPKM, FSSPKM and the original k-median. It can be observed that the sample-preserved medians tend to get closer to the majority of the samples in this example. It may imply that those medians are most representative of the samples in the cluster.

To solve FSSPKM, a bilinear program FSSPKM clustering algorithm is proposed, applying similar ideas used in [15]. It contains two linear programming problems adapting a pre-calculated distance matrix and hence forms a sample-preserved method. Using this method, the FSSPKM algorithm has two major benefits, the efficiency due to the linear programming formulation, and the flexibility due to the use of a distance matrix. It is especially beneficial when measuring similarities of time series, where variant distance measures are usually considered. Also the obtained k medians are sample-preserved in every single feature space, which means that all cluster centers are existing samples. As a result, the "shape" of data is retained in those cluster centers. Further classification may be more meaningful.

Let D denote a three-dimensional distance matrix whose element D_{ijs} represents the distance from sample i to sample j at a feature space s for $i, j = 1, \ldots, n$ and

$s = 1, \ldots, m$. By having this distance matrix in a clustering algorithm, variant distance measures can be applied. A decision variable $\bar{X} \in \{0,1\}^{n \times m \times k}$ is used to determine the optimal decision of choosing cluster centers. If $\bar{X}_{isp} = 1$, sample i is determined to be a cluster median for cluster p at feature space s, for $i = 1, \ldots, n$, $s = 1, \ldots, m$ and $p = 1, \ldots, k$; otherwise, $\bar{X}_{isp} = 0$. Let X be the relaxed variable of $\bar{X}$. It will be shown that the proposed algorithm gives binary solutions by solving the relaxation of the integer program under a certain condition.

Program (6.5) is reformulated as a dual minimization problem. In the optimal solution to the inner minimization, the minimum distances from each sample to each of the k cluster centers are found, which can be viewed as cluster assignment. This optimal objective is to find the closest cluster for each sample. The overall objective function tries to find the optimal cluster medians such that the sum of distances from each sample to its nearest cluster median is minimized. The overall solution to Program (6.5) provides the optimal samples that represent the k cluster medians.

$$\begin{aligned}
\min_{X} \quad & \sum_{i=1}^{n} \min_{p=1,\ldots,k} \{ \sum_{j=1}^{n} \sum_{s=1}^{m} D_{ijs} X_{jsp} \} && \\
\text{s.t.} \quad & && \\
& \sum_{i=1}^{n} X_{isp} = 1 && \forall s, p \\
& \sum_{p=1}^{k} X_{isp} \le 1 && \forall i, s \\
& 0 \le X_{isp} \le 1 && \forall i, s, p.
\end{aligned} \tag{6.5}$$

To solve the dual minimization problem in Program (6.5), the objective function is reduced into a single minimization problem. The idea of Lemma 2.1 in [15] (Lemma 2.4.1) is applied and another equivalent optimization problem is formed with a single nonlinear objective function with linear inequalities as shown in Program (6.6).

Lemma 2.4.1 indicates that solving Program (6.5) is equivalent to solving Program (6.6). The objective function (6.6a) contains the element-by-element multiplication of two decision variables X_{isp} and T_{ip} for all i, s and p. Therefore, to satisfy the equivalent condition stated in Lemma 2.4.1, Constraints (6.6c), (6.6d) and (6.6e) are included in the formulation.

$$\min_{X,T} \quad \sum_{i=1}^{n}\sum_{p=1}^{k}\left\{\sum_{j=1}^{n}\sum_{s=1}^{m} D_{ijs}X_{jsp}\right\}T_{ip} \tag{6.6a}$$

s.t.

$$\sum_{p=1}^{k} X_{isp} \le 1 \qquad \forall i, s \tag{6.6b}$$

$$0 \le X_{isp} \le 1 \qquad \forall i, s, p$$

$$\sum_{p=1}^{k} T_{ip} = 1 \qquad \forall i \tag{6.6c}$$

$$\sum_{i=1}^{n} X_{isp}T_{ip} = 1 \qquad \forall s, p \tag{6.6d}$$

$$T_{ip} \ge 0 \qquad \forall i, p. \tag{6.6e}$$

6.2.1 Uncoupled Bilinear Program Algorithm

The concept of Uncoupled Bilinear Program Algorithm (UBPA) proposed in [7, 15] can be applied on solving this nonlinear optimization problem. The algorithm contains two major linear programming (LP) formulations. One is for cluster assignment, and the other is for cluster median update.

1. **LP Formulation for Cluster Assignment:**

$$\begin{aligned} \min_{T} \quad & \sum_{i=1}^{n}\sum_{p=1}^{k}\left\{\sum_{j=1}^{n}\sum_{s=1}^{m} D_{ijs}X_{jsp}\right\}T_{ip} \\ \text{s.t.} \quad & \\ & \sum_{p=1}^{k} T_{ip} = 1 \qquad \forall i \\ & T_{ip} \ge 0 \qquad \forall i, p. \end{aligned} \tag{6.7}$$

2. **LP Formulation for Cluster Median Update:**

$$\min_X \sum_{i=1}^{n}\sum_{p=1}^{k}\left\{\sum_{j=1}^{n}\sum_{s=1}^{m} D_{ijs}X_{jsp}\right\} T_{ip}$$

s.t.

$$\sum_{p=1}^{k} X_{isp} \le 1 \qquad \forall i, s \qquad (6.8a)$$

$$\sum_{i=1}^{n} X_{isp}T_{ip} = 1 \qquad \forall s, p \qquad (6.8b)$$

$$0 \le X_{isp} \le 1 \qquad \forall i, s, p.$$

Algorithm 6.2.1 Bilinear Program Feature Space Sample-Preserved k-Median (FSSPKM) Clustering Algorithm

Input: k = number of clusters and X^0, an initial decision of k samples as cluster medians.

Set $t = 0$ and solve the two linear programs alternatively.

1. **Cluster Assignment.** Given X^t, find the solution T^t of Program (6.7).
2. **Cluster Median Update.** Given T^t from the optimal solution of Program (6.7), update X^{t+1} by solving Program (6.8).

Stop when $\sum_{p=1}^{k} X_{isp}^{t} = \sum_{p=1}^{k} X_{isp}^{t+1}$ for all i, s. Set $X^* = X^{t+1}$ and $T^* = T^t$. Otherwise, increment t by one and continue to solve Program (6.7) and Program (6.8).

Output: A decision of cluster median, X^*, and cluster assignment, T^*.

Once the two major LP problems are formulated, local optimal solutions to Program (6.6) can be found by starting with a set of initial cluster centers and alternately solving the two linear programs, as shown in Algorithm 6.2.1. Both Program (6.7) and Program (6.8) can produce binary solutions of X_{isp} and T_{ip} for all i, s and p. Proposition 6.2.1 confirms that Program (6.7) gives binary solutions. Proposition 6.2.2 states a condition for Program (6.8) to produce binary solutions. It is to assume that the sum of the distances from sample i to all other samples in the same cluster p is "unique" within that cluster for all $i \in G_p$. This assumption is not difficult to achieve. Only in a very rare case such that all samples in a cluster are symmetric with possible cluster medians, the sum of the distances to other samples can be equal for at least two separate samples. In such cases, fractional solutions may occur. Even if the assumption is not satisfied, the samples with fractional solutions would indicate that any of them can equally be a cluster median. The fractional solution is due to the equal minimum sum of the distances from the samples with the fractional solution to all other samples in the cluster. Hence, any one of the samples can be taken as a solution. With these two propositions, one can approximate the solution of FSSPKM efficiently. Only the two simple linear programs are needed. Algorithm 6.2.1 converges to a local optimal solution, according to a similar idea of the finite termination results proposed in [7, 15], which is given in Proposition 6.2.3.

Proposition 6.2.1. *There exists an optimal solution of Program (6.7) such that* $T \in \{0,1\}^{n\times k}$.

Proof. The constraint matrix in Program (6.7) is totally unimodular. By Proposition 3.3 in [95], the linear program (6.7) has integral optimal solutions. Since T_{ip} is bounded by 0 and 1, for $i = 1,\ldots,n$ and $p = 1,\ldots,k$, Program (6.7) has a binary optimal solution. □

Proposition 6.2.2. *Let* $\Theta_{jsp} = \sum_{i=1}^{n} D_{ijs}T_{ip}$ *for* $j = 1,\ldots,n;\ s = 1,\ldots,m$ *and* $p = 1,\ldots,k$. *Given* s, *for each* p, *if every element of* Θ_{jsp} *for all* j *is unique, then the linear programming relaxation of Program (6.8) gives binary solutions of* X.

Proof. It can be proved by contradiction. Consider a feature space s. Recall that Θ_{jsp} is the sum of distances from sample j to all other samples in the same cluster p. Constraint (6.8a) ensures that each sample can be a cluster center for at most one cluster. Constraint (6.8b) indicates that for a given p, exactly one of the samples that assigned to cluster p is a cluster center. Objective (6.8) finds the k minimum values of Θ_{jsp} for all $j \in G_p$ for each p.

Suppose that for each p in a given s, every element of Θ_{jsp} for all $j \in G_p$ is unique. Also, suppose that an optimal solution of Program (6.8) contains X^*_{isp} and $X^*_{i'sp}$ that are fractional for a given p and s such that $X^*_{isp}T_{ip} + X^*_{i'sp}T_{i'p} = 1$ and $X^*_{isp} + X^*_{i'sp} = 1$ due to Constraint (6.8b). Then the term $X^*_{isp}\Theta_{ip} + X^*_{i'sp}\Theta_{i'p}$ in the objective is the minimum among any $\sum_{j \in G_p} X_{jsp}\Theta_{jp}$ formed by all feasible values of $X'_{jsp}s$ corresponding to a given p. Since every element of Θ_{jsp} for all $j \in G_p$ is unique, without loss of generality, assume that $\Theta_{isp} < \Theta_{i'sp}$, which implies that there exists a minimum value of Θ_{jsp} for all $j \in G_p$.

Then $\Theta_{isp} = X^*_{isp}\Theta_{isp} + X^*_{i'sp}\Theta_{isp} < X^*_{isp}\Theta_{isp} + X^*_{i'sp}\Theta_{i'p}$ since $\Theta_{isp} < \Theta_{i'sp}$ and $X^*_{isp} + X^*_{i'sp} = 1$. There exist a solution $X^{**}_{isp} = 1$ such that the term $X^{**}_{isp}\Theta_{ip}$ minimizes the objective function for the given p. This contradicts the fact that $X^*_{isp}\Theta_{isp} + X^*_{i'sp}\Theta i'sp$ must be the minimum among all feasible solution of $X'_{jsp}s$. Hence, the optimal solutions of X_{isp} are binary if for each p every element of Θ_{jsp} for all $j \in G_p$ is unique in a given s. □

Proposition 6.2.3. *The Bilinear Program FSSPKM Clustering Algorithm (Algorithm 6.2.1) terminates in a finite number of iterations at a cluster assignment that is locally optimal.*

Proof. It needs to shown that the objective value of Program (6.6) can not be decreased by either reassignment of a sample to a different cluster, or by deciding new cluster median for any of the clusters. At each iteration, in the step for cluster assignment, each sample is assigned to a closest median. Thus, the objective value of program (6.6) can not be increased. In the step for cluster median update, each cluster median is recomputed in order to minimize the sum of all distances from each sample in the cluster to its cluster median. Hence, the overall objective value can be either strictly decreased or can stay the same as the algorithm terminates.

It can be concluded that the algorithm must terminate at some clustering assignment that is locally optimal, because the algorithm does not repeat assignments by the stopping criterion, the overall objective function is non-increasing and bounded below by zero, and there is a finite number of ways to assign n samples to k clusters. □

6.2.2 Equivalent Algorithm without Solving the LP Formulations

By introducing two matrices Δ and Θ, an equivalent algorithm to Algorithm 6.2.1 can be formulated without the two linear programs. It is a non-LP sequential search

Algorithm 6.2.2 (Non-LP) Sequential Search Feature Space Sample-Preserved k-Median Clustering Algorithm

Input: $k =$ number of clusters; X^0_{isp}, an initial decision of k samples as cluster medians, for all i, s and p; and D_{ijs}, a distance matrix for $i, j = 1, \ldots, n$ and $s = 1, \ldots, m$.

Set $t = 0$ and solve the following two problems alternatively.

1. **Cluster Assignment.** Compute $\Delta_{ip} = \sum_{j=1}^{n} \sum_{s=1}^{m} D_{ijs} X^t_{jsp}$ for $i = 1, \ldots, n$ and $p = 1, \ldots, k$. Then Δ_{ip} represents distance from sample i to the cluster center p. Find the closest median p^* to each sample i.

$$p^* = \arg \min_{p=1,\ldots,k} \Delta_{ip} \quad \text{for each } i.$$

   Set $T^t_{ip^*} = 1$ and $T^t_{ip} = 0$ for $i = 1, \ldots, n$ and $p = 1, \ldots, k \setminus p^*$.

2. **Cluster Median Update.** Compute $\Theta_{isp} = \sum_{j=1}^{n} D_{ijs} T^t_{jp}$. Then Θ_{isp} represents the sum of distances from sample i to all other samples in the same cluster p at the dimension of feature s. For each feature s and each cluster p, find the sample i^* which has the smallest sum of distances to all other samples in the same cluster.

$$i^* = \arg \min_{i \in G_p} \Theta_{isp} \quad \text{for each } s, p.$$

   Set $X^{t+1}_{i^*sp} = 1$ and $X^{t+1}_{isp} = 0$ for $i = 1, \ldots, n \setminus i^*$.

Stop when $\sum_{p=1}^{k} X^t_{isp} = \sum_{p=1}^{k} X^{t+1}_{isp}$ for all i. Set $X^* = X^{t+1}$ and $T^* = T^t$. Otherwise, increment t by one and continue to solve the two problems.

Output: A decision of cluster median, X^*, and cluster assignment, T^*.

FSSPKM algorithm as shown in Algorithm 6.2.2.

In the step of cluster assignment, a sample is assigned to the closest cluster median. Let

$$\Delta_{ip} = \sum_{j=1}^{n} \sum_{s=1}^{m} D_{ijs} X_{jsp} \quad \text{for all} \quad i, p.$$

Then Δ_{ip} represents the distance from sample i to the cluster median p. Sample i is then assigned to a closest cluster median p with the distance obtained by the minimum of Δ_{ip} for $p = 1, \ldots, k$. This solution is the same as the exact solution to Program (6.7).

Similarly, in the step of cluster median update, according to assigned samples in a cluster, a cluster median of the cluster is a sample, which has the smallest distance to all other samples in the cluster. Let

$$\Theta_{isp} = \sum_{j=1}^{n} D_{ijs} T_{jp} \quad \text{for all} \quad i, s, p.$$

Then given p and s, Θ_{isp} represents the sum of distances from sample i to all other samples in the same cluster p at a feature space s for all $i \in G_p$. Hence, the cluster median is the sample that has the minimum of Θ_{isp} among all $i \in G_p$. As in Proposition 6.2.2, for each p, if every element of Θ_{isp} for all $i \in G_p$ is unique, then the solution of cluster medians from Algorithm 6.2.2 is the same as the one from Program (6.8). If the uniqueness condition is not satisfied, any sample i having the same minimum Θ_{isp} as other samples in a cluster p can be taken as a solution. In this case, Algorithm 6.2.2 can also pick the same solution as Program (6.8).

Since the same idea is used in Algorithm 6.2.1, Proposition 6.2.3 also applies to Algorithm 6.2.2 that in finite number of steps it converges to a local optimal solution.

6.3 Feature Selection in FSSPKM Clustering

Feature selection for unsupervised learning is difficult since no class labels are used when clusters are generated and therefore there is no clear guide for choosing significant features [26, 34, 59, 80]. The proposed feature selection methods utilize the sample-to-sample distance matrix at each feature space. Distances between cluster medians and their densities are considered. The selection algorithm is based on a mild modification of the two well known wrapper approaches: forward subset selection (FSS) and backward subset selection (BSS) [30]. The initial subset of FSS contains no features. At each step, it adds the feature that improves the classification the most to the subset. This process continues until no improvement is possible. However, the difference to the proposed FSS in FSSPKM clustering is that the process continues until no features are left. The best combination of subset is the one that gives the highest classification accuracy. BSS starts with an initial subset with all features included, and the worst feature is eliminated from the subset at each step until no improvement is possible. Similarly, the proposed BSS in FSSPKM clustering is to have the process continued until only one feature is left. Then the optimal number of features can be determined.

A possible feature selection in FSSPKM clustering can be made at the second major step of the FSSPKM algorithm (Algorithm 6.2.1), where new cluster medians are determined. Algorithm 6.3.1 displays a modified formulation. Given μ, which denotes the number of desired features, the optimization program of the second step can be reformulated into an integer program by adding constraints to force the optimization model to consider only μ features. Program (6.10) is the optimization model that automatically decides which of the optimal μ features to keep and which others to drop from the data. Constraint (6.10a) forces only μ features to be considered in each cluster. Constraints (6.10b) and (6.10c) together ensures that the model keeps same features at all clusters. The binary variable $y \in \{0,1\}^m$ is a decision variable used to decide whether to keep a feature or not. The original decision variable X has to be binary since there is no sufficient condition for the relaxation problem to provide integral solution due to the added constraints. This formulation then contains $n \times m \times k + m$ binary variables. It is an integer programming problem that can not be solved efficiently.

In addition, there is a problem of selecting incorrect features because the objective function of FSSPKM is to minimize the sum of the distances from every sample to its closest cluster's median. Consider an unwanted feature space such that clusters are

Algorithm 6.3.1 Possible Feature Selection Forced in FSSPKM

Input: μ = desired number of features; k = number of clusters; X^0_{isp}, an initial decision of k samples as cluster medians, for all i, s and p; and D_{ijs}, a distance matrix for i, j and s.

Set $t = 0$ and solve the following two problems alternatively.

1. **Cluster Assignment.** Given X^t_{isp}, for all i, s and p, find the solution T^t to the linear program problem:

$$\begin{aligned} \min_{T^t} \quad & \sum_{i=1}^{n}\sum_{p=1}^{k}\left\{\sum_{j=1}^{n}\sum_{s=1}^{m} D_{ijs}X^t_{jsp}\right\} T_{ip} \\ \text{s.t.} \quad & \\ & \sum_{p=1}^{k} T_{ip} = 1 && \forall i \\ & T_{ip} \geq 0 && \forall i, p. \end{aligned} \tag{6.9}$$

2. **Cluster Median Update.** Given T^t from the optimal solution of program (6.9), update X^{t+1} by solving the integer programming problem:

$$\min_{X^{t+1}} \sum_{i=1}^{n}\sum_{p=1}^{k}\left\{\sum_{j=1}^{n}\sum_{s=1}^{m} D_{ijs}X_{jsp}\right\} T^t_{ip}$$

s.t.

$$\sum_{p=1}^{k} X_{isp} \leq 1 \qquad \forall i, s$$

$$\sum_{i=1}^{n}\sum_{s=1}^{m} X_{isp}T^t_{ip} = \mu \qquad \forall p \tag{6.10a}$$

$$\sum_{i=1}^{n}\sum_{p=1}^{k} X_{isp}T^t_{ip} = ky_s \qquad \forall s \tag{6.10b}$$

$$\sum_{s=1}^{m} y_s = \mu \tag{6.10c}$$

$$X_{isp} \in \{0, 1\} \qquad \forall i, s, p$$

$$y_s \in \{0, 1\} \qquad \forall s \in \{1, \ldots, m\}.$$

Stop when $\sum_{p=1}^{k} X^t_{isp} = \sum_{p=1}^{k} X^{t+1}_{isp}$ for all i. Set $X^* = X^{t+1}$ and $T^* = T^t$. Otherwise, increment t by one and continue to solve the two problems.

Output: A decision of cluster median, X^*, and cluster assignment, T^*.

not very well defined and samples are closely mixed with each other. In such a case, the k cluster centers are very close to each other as well. The reformulated model will keep that unwanted feature since it has the smallest sum of distances. Therefore, better feature selection criteria is needed to set up for selecting features in the FSSPKM clustering.

Here, a good feature space is defined as having its samples densely grouped in each cluster and having its centers as far from each other as possible. The sample-to-sample distance matrix D is utilized to measure these properties to include best features for FSS and to drop worst features for BSS. Four feature selection criteria are proposed based on this definition.

6.3.1 Feature Selection Algorithm

In this section, four feature selection criteria are introduced. They can be applied on both FSS and BSS. Observe that to obtain a complete subset selection, a clustering algorithm needs to be tested $m(m+1)/2$ times for FSS while only m times for BSS. Thus, BSS needs less iterations than FSS. Due to this efficiency the feature selection criteria designed for BSS are first described. Then they can be easily reversed for FSS.

6.3.1.1 Backward Subset Selection Algorithm

The objective function of the FSSPKM Clustering (Problem 6.5) is to find k samples as centers of each feature space for the k clusters such that the sum of the distances of all samples to their closest cluster medians are minimized. Let $l_p^s \in G_p$ represent the index of the sample that is the cluster median in cluster p of feature space s. The objective of FSSPKM is the solution of the minimum sum of the distances from all samples to its closest cluster medians. That is:

$$\min_{G_p, l_p^s} \quad \sum_{i \in G_p} \sum_{s=1}^{m} D_{i, l_p^s, s}, \quad p = 1, \ldots, k.$$

Since this minimization problem is independent among features, each feature can be treated separately. To minimize to-the-center distances of each feature space, the problem is divided into m subproblems for a fixed feature space $s \in \{1, \dots, m\}$ as follows:

$$\min_{G_p, l_p^s} \sum_{i \in G_p} D_{i, l_p^s, s}, \quad p = 1, \dots, k. \tag{6.11}$$

The BSS starts with an initial subset with all features included, and the worst feature is eliminated from the subset at each step until one feature is left. The goal is to find a criterion for dropping the worst feature at each iteration. Since the objective of the FSSPKM clustering is to minimize the function (6.11), the clustering algorithm would favor the features that have the smaller sum of the distances from each sample to its cluster's median. The clustering algorithm would *not* favor the features that have a large sum (or small sample density).

Given a solution of cluster medians and cluster assignment, the worst feature may contain a cluster p that has the largest sum of the distances from all samples in the cluster to the median. That is a cluster who has the largest to-the-center sample distance and whose samples are less densely grouped. In a fixed feature space s, the largest sum of to-the-center sample distances among all clusters is denoted as:

$$\max_{p=1,\dots,k} \sum_{i \in G_p} D_{i, l_p^s, s}. \tag{6.12}$$

This is not enough. A "bad" feature space may have its clusters close to each other. Such a feature may need to be eliminated as well. The closest center-to-center distance among all features is defined as follows:

$$\min_{s=1,\dots,m} \sum_{p=1}^{k-1} \sum_{q=p+1}^{k} D_{l_p^s, l_q^s, s}. \tag{6.13}$$

Combining the above two measurements (6.12) and (6.13) gives a BSS Minimum Maximum (MM) Feature Selection FSSPKM Algorithm (Algorithm 6.3.2). Let V denote the set of all the current optimally selected features. For BSS, the initial set of V contains all features. In the maximization, the largest cluster $p_s \in \{1, \dots, k\}$ of each feature space $s \in \{1, \dots, m\}$ is chosen, which has the largest sum of the distances from

Algorithm 6.3.2 BSS Minimum Maximum (MM) Feature Selection FSSPKM Algorithm

1. Let $V = \{1, \ldots, m\}$.
2. Apply the FSSPKM algorithm to cluster a data set into k disjoint groups considering features in V. That is to obtain cluster assignments G_p for $p = 1, \ldots, k$ and cluster center indices l_p^s for $p = 1, \ldots, k$ and $s \in V$.
3. For each feature space $s \in V$ and for each cluster $p \in \{1, \ldots, k\}$ compute
$$p_s^* = \arg \max_{p=1,\ldots,k} \sum_{i \in G_p} D_{i,l_p^s,s}.$$
4. Delete feature s^* from V, where
$$s^* = \arg \min_{s \in V} \sum_{q \neq p_s^*, q=1}^{k} D_{l_{p_s^*}^s, l_q^s, s}.$$
5. Stop if all features are deleted, else let $D = \bar{D} \in \mathbb{R}^{n \times n \times \bar{m}}$, where $\bar{m} = m - 1$ and $\bar{D}$ is the matrix with feature s^* deleted.
6. Go to Step 2.

the cluster median to all other samples in that cluster. After obtaining the largest cluster at each feature, the sum of the distances from the determined median to all other medians is calculated. Finally, the feature that has the smallest sum of the distances from the largest cluster's median to all other cluster medians is deleted from the set V. The algorithm is then continued by using the FSSPKM to cluster the data again without the deleted feature. This is repeated until only one feature is left in V.

In addition to the MM feature selection algorithm, three ratios are proposed, α, β and γ. Algorithm 6.3.3 displays the use of α-ratio, which can be replaced by β and γ for different feature selections. All three ratios are trying to find the feature that has a small sum of center-to-center distances. The α-ratio in Equation (6.14) determines the worst feature by minimizing the sum of center-to-center distances while maximizing the *sum* of to-the-center sample distances of all clusters in a feature space. The β-ratio in Equation (6.15) finds the feature that has the minimum center-to-center distance per *average* sum of to-the-center sample distances of all clusters.

Algorithm 6.3.3 BSS α-Ratio Feature Selection FSSPKM Algorithm

1. Let $V = \{1, \ldots, m\}$.
2. Apply the FSSPKM algorithm to cluster a data set into k disjoint groups considering features in V.
3. For each feature space $s \in V$, compute
$$\alpha_s = \frac{\sum_{p=1}^{k-1} \sum_{q=p+1}^{k} D_{l_p^s, l_q^s, s}}{\sum_{p=1}^{k} \sum_{i \in G_p} D_{i, l_p^s, s}}. \tag{6.14}$$
4. Delete feature s^* from V, where
$$s^* = \arg \min_{s \in V} \alpha_s.$$
5. Stop if all features are deleted, else let $D = \bar{D} \in \mathbb{R}^{n \times n \times \bar{m}}$, where $\bar{m} = m - 1$ and $\bar{D}$ is the matrix with feature s^* deleted.
6. Go to Step 2.

$$\beta_s = \frac{\sum_{p=1}^{k-1} \sum_{q=p+1}^{k} D_{l_p^s, l_q^s, s}}{\sum_{p=1}^{k} \frac{1}{|G_p|} \sum_{i \in G_p} D_{i, l_p^s, s}}. \tag{6.15}$$

The γ-ratio in Equation (6.16) considers only one sample at each cluster that has the largest distance to its cluster center. The largest distance can be seen as a radius that forms a neighborhood which covers all samples in the cluster. It finds the worst feature by minimizing center-to-center distance per sum of the *largest* to-the-center sample distance over all clusters.

$$\gamma_s = \frac{\sum_{p=1}^{k-1} \sum_{q=p+1}^{k} D_{l_p^s, l_q^s, s}}{\sum_{p=1}^{k} \max_{i \in G_p} D_{i, l_p^s, s}}. \tag{6.16}$$

6.3.1.2 Forward Subset Selection Algorithm

Similar to BSS, the four proposed feature selection criteria can be applied on FSS. They are simply the negation of those in BSS. Let $\bar{m}$ be the current number of optimally selected features. The initial subset of FSS contains no features, so FSS starts with $\bar{m} = 0$ and the set of all the current optimally selected features $V = \{\emptyset\}$. At each iteration, it adds the best feature to the subset V. Let $\bar{V}_s = V \cup \{s\}$ denote a set of selected features with one unselected feature s for each $s \in \{1, \dots, m\} \setminus V$, where the set $\{1, \dots, m\} \setminus V$ contains those features that are not currently selected. The FSSPKM algorithm is then tested on the data set with features in $\bar{V}_s$ for each unselected features $s \in \{1, \dots, m\} \setminus V$ in order to determine the next optimal feature that should be selected. The algorithm stops when $\bar{m} = m$, which is when $V = \{1, \dots, m\}$.

Algorithm 6.3.4 displays the Maximum Minimum Feature Selection FSSPKM Algorithm for FSS. The distance matrix $\tilde{D}$ represents the sample-to-sample distance matrices of currently selected features. At the first iteration, each feature is treated separately and the FSSPKM clustering algorithm is operated m times to obtain optimal clusters and cluster medians at each feature space. Then the best feature is kept in V according the MM criterion. In the second iteration, FSSPKM cluster algorithm is operated $m-1$ times. At each time two features are considered. One is from the unselected features and the other is the selected feature in V. Then the next best feature is selected. The algorithm continues until all features are selected.

Instead of finding the minimum α ratio in BSS in order to drop the worst feature, the FSS tries to find the feature with the maximum α ratio that should be included in the subset. The FSS α-Ratio Feature Selection FSSPKM Algorithm is displayed in Algorithm 6.3.5. The α-ratio (6.17) in Algorithm 6.3.5 can then be replaced by β-ratio (6.15) or γ-ratio (6.16) with the distance matrix replaced by $\tilde{D}^s$. Then different feature selection criteria can be applied.

Algorithm 6.3.4 FSS Maximum Minimum (MM) Feature Selection FSSPKM Algorithm

Let $t = 1, V = \{\emptyset\}$ and $\tilde{D} \in \mathbb{R}^{n\times n\times \bar{m}}$, where $\bar{m} = |V| = 0$.

Repeat

1. Let $\bar{V}_s = V \cup \{s\}$ with feature s appended at the end of the set V, for each $s \in \{1, \ldots, m\} \setminus V$. For each feature space $s \in \{1, \ldots, m\} \setminus V$, compute $\tilde{D}^s$ considering only those features in $\bar{V}_s$. That is a three-dimensional distance matrix having the $n \times n$ distance matrix of feature space s appended at the end of the (t-1)th (also the last) feature of $\tilde{D}$. The feature s is then the tth feature of $\tilde{D}^s$, where $\tilde{D}^s \in \mathbb{R}^{n\times n\times t}$.

2. Apply the FSSPKM algorithm $(m-t+1)$ times. Each time is to cluster the data set into k disjoint groups considering features in $\bar{V}_s$. At each time the tth (also the last) element of the set $\bar{V}_s$ refers to feature s for each $s \in \{1, \ldots, m\} \setminus V$. This is to obtain cluster assignments $G_p^{\tilde{s}}$ and cluster center indices $l_p^{\tilde{s}}$ for $p = 1, \ldots, k$ and for all $\tilde{s} \in \bar{V}_s$.

3. For each feature space $s \in \{1, \ldots, m\} \setminus V$ and for each cluster $p \in \{1, \ldots, k\}$ compute
$$p_s^* = \arg \min_{p=1,\ldots,k} \sum_{i\in G_p^t} \tilde{D}^s_{i,l_p^t,t}.$$

4. Add feature s^* to the data set, where
$$s^* = \arg \max_{s\in\{1,\ldots,m\}\setminus V} \sum_{q\neq p_s^*,q=1}^{k} \tilde{D}^s_{l^t_{p_s^*},l^t_q,t}.$$

5. Let $\tilde{D} = \tilde{D}^{s^*} \in \mathbb{R}^{n\times n\times t}$, include s^* in V.

6. Increment t by 1.

Until $(t > m)$.

Algorithm 6.3.5 FSS α-Ratio Feature Selection FSSPKM Algorithm

Let $t = 1, V = \{\emptyset\}$ and $\tilde{D} \in \mathbb{R}^{n \times n \times \bar{m}}$, where $\bar{m} = |V| = 0$.

Repeat

1. Let $\bar{V}_s = V \cup \{s\}$ with feature s appended at the end of the set V, for each $s \in \{1, \ldots, m\} \setminus V$. For each feature space $s \in \{1, \ldots, m\} \setminus V$, compute $\tilde{D}^s$ considering only those features in $\bar{V}_s$. That is a three-dimensional distance matrix having the $n \times n$ distance matrix of feature space s appended at the end of the (t-1)th (also the last) feature of $\tilde{D}$. The feature s is then the tth feature of $\tilde{D}^s$, where $\tilde{D}^s \in \mathbb{R}^{n \times n \times t}$.

2. Apply the FSSPKM algorithm $(m - t + 1)$ times. Each time is to cluster the data set into k disjoint groups considering features in $\bar{V}_s$. At each time the tth (also the last) element of the set $\bar{V}_s$ refers to feature s for each $s \in \{1, \ldots, m\} \setminus V$. This is to obtain cluster assignments $G_p^{\tilde{s}}$ and cluster center indices $l_p^{\tilde{s}}$ for $p = 1, \ldots, k$ and for all $\tilde{s} \in \bar{V}_s$.

3. For each feature space $s \in \{1, \ldots, m\} \setminus V$, compute
$$\alpha_s = \frac{\sum_{p=1}^{k-1} \sum_{q=p+1}^{k} \tilde{D}^s_{l_p^t, l_q^t, t}}{\sum_{p=1}^{k} \sum_{i \in G_p^t} \tilde{D}^s_{i, l_p^t, t}}. \tag{6.17}$$

4. Add feature s^* to the data set, where
$$s^* = \arg \max_{s \in \{1, \ldots, m\} \setminus V} \alpha_s.$$

5. Let $\tilde{D} = \tilde{D}^{s^*} \in \mathbb{R}^{n \times n \times t}$, include s^* in V.

6. Increment t by 1.

Until $(t > m)$.

Chapter 7

APPLICATIONS OF DEVELOPED OPTIMIZATION MODELS FOR CLUSTERING

The developed clustering techniques, which are sample-preserved k-median (SPKM) clustering, feature space sample-preserved k-median (FSSPKM) clustering and feature selection algorithms tailor made for the FSSPKM clustering, are applied on real world data sets. Characteristics of the data sets, performance evaluation techniques and experimental results are discussed in this Chapter.

7.1 Experimental Data Sets

The clustering algorithms are tested on three types of data sets in order to examine their clustering performance. These data sets are attribute (non-time series), single (or univariate) time series and multivariate time series data sets. The term, non-time series, is referred to static data and is used to distinguish them from time series data. Non-time series data are those whose samples can be represented as real vectors. For example, a non-time series data set may contain attributes (or features): age, weight and height, which are real-valued. An instance may be represented by a vector that contains these three elements. Time series data have values of a fixed attribute but vary with time. For example, stock prices vary with time and form a time series. Multivariate time series data have both properties, having multiple attributes and varying with time. For example, sensor data may consist of multiple series of observations from recordings of multiple sensors over a period of time. Each sensor counts an attribute (or a feature) of a time series. A combination of these recordings forms a multivariate time series.

7.1.1 Attribute Data Sets

Attribute data sets are numerical data where each sample can be represented as a vector. Each element in a vector corresponds to an attribute. Table 7.1 shows numbers of classes and features for each of the seven publicly available data sets from UCI database [3].

Each of the first four data sets has two classes. Features in the Wisconsin Diagnostic Breast Cancer (WDBC) data set are characteristics of the cell nuclei present in a digitized image of a fine needle aspirate of a breast mass. The class distribution is 357 benign and 212 malignant cases. The features in BUPA liver disorders (BLD) data set contains results of blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. There are 345 instances, 145 are negative and 200 positive. In the Cleveland Heart Disease (HD) data set, 13 attributes are used for the presence of heart disease in 303 patients, 164 are negative and 139 are positive. Pima Indian Diabetes (PID) data set contains eight attributes that may show signs of diabetes of patients. There are 500 negative and 268 positive instances in the data set.

Both the Wine Recognition (Wine) and Iris Plants (Iris) data sets are three-group. The Wine data are the results of a chemical analysis of wines grown in the same region from three different cultivars in Italy. Numbers of instances of each class are 59, 71 and 48. The Iris data set contains lengths and widths of sepal and petal of three types of iris. Three classes are equally distributed. Each has 50 instances. The Image Segmentation (Image) data set has 19 continuous attributes of instances of seven outdoor images. Each of the seven classes has 330 instances.

Table 7.1: Parameters of UCI Data Sets [3]

Data Set	n	m	k
Wisconsin Diagnostic Breast Cancer (WDBC)	569	30	2
Bupa Liver Disorders (BLD)	345	6	2
Heart Disease (HD)	303	13	2
Pima Indian Diabetes (PID)	768	8	2
Wine Recognition (Wine)	178	13	3
Iris Plants (Iris)	150	4	3
Image Segmentation (Image)	2310	19	7

Table 7.2: University of California-Riverside Time Series Data Sets [55].

Data Set	Number of Classes (k)	Size of Training Set	Size of Testing Set	Length
Synthetic Control	6	300	300	60
Gun-Point	2	50	150	150
CBF	3	30	900	128
Face (all)	14	560	1690	131
OSU Leaf	6	200	242	427
Swedish Leaf	15	500	625	128
50Words	50	450	455	270
Trace	4	100	100	275
Two Patterns	4	1000	4000	128
Wafer	2	1000	6174	152
Face (four)	4	24	88	350
Lightning-2	2	60	61	637
Lightning-7	7	70	73	319
ECG	2	100	100	96
Adiac	37	390	391	176
Yoga	2	300	3000	426
Fish	7	175	175	463
Beef	5	30	30	470
Coffee	2	28	28	286
OliveOil	4	30	30	570

7.1.2 Single Time Series Data Sets

A single time series is a series of observations made over a period of time. It can be treated as a vector, and each element in a vector is related to a specific time. Such time series data set is called a *univariate* time series data set because it has only one feature (or attribute).

The University of California-Riverside (UCR) time series data sets [55] have been used to measure performances of a number of clustering and classification methods. Each data set contains training and testing subsets. The collected data sets consists of 20 real world and synthetic time series data sets and includes various time series data properties. Information of these data sets can be found in Table 7.2, including number of classes, number of training and testing samples and the length of time series.

7.1.3 Multivariate Time Series Data Sets

When a series of observations over a period of time contains more than one feature, it is called a *multivariate* (multidimensional or multi-attribute) time series (MTS). Each individual sample is a matrix with information from two characteristics: features (spatial property) and time (temporal property).

Table 7.3 displays characteristics of a collection of EEG data sets used in [20]. Each

Table 7.3: EEG Data Set Characteristics [20].

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
n	42	42	132	114	102	102	54	138	120	72

$m = 26$, $k = 2$ and each segment contains 30 time points.

of the data sets has two groups, positive and negative, that are equally distributed. These data sets are multivariate since each time segment is from a simultaneous recording of 26 channels. Each channel is viewed as one feature.

7.2 Experimental Results Using BPSPKM and FSSPKM Clustering Methods

The original k-median clustering, the SPKM clustering in IP formulation, together with the proposed BPSPKM and FSSPKM clustering algorithms are tested on real world data sets that are publicly available. Three types of data sets are tested: attribute data from UCI database collected by [3], single time series from UCR database created by [55] and multivariate time series from multichannel EEG recordings used in [20]. Since both BPSPKM and FSSPKM clustering are the same for univariate data sets, FSSPKM clustering is tested only on the other two multivariate data sets: attribute data and multivariate time series data.

7.2.1 Clustering Performance

To measure the performance of the k-median clustering methods, they are tested on variant data sets. Each of the data sets is divided into two disjoint subsets, the training set and the testing set, so that the clusters can be trained and tested accordingly as shown in [15] and [88]. Each clustering method is then used to cluster the training set. For the k-median clustering, the majority of a class in each cluster defines the cluster label. The obtained k cluster medians are then used to classify samples in the testing set. Each sample is assigned to the cluster whose cluster median is the closest to that sample. The true class labels are then used to determine the clustering accuracy. A method that gives the more correctly classified samples has the better performance.

In the training process, class labels of the training samples are assumed to be unknown. After clusters are constructed by a clustering method, the class labels are used to measure the training accuracy. Since the cluster label is determined by the majority of the samples in that cluster, this majority is considered to be correctly clustered samples and the rest are taken as incorrectly clustered. The training accuracy is then the percentage of correctly clustered samples in all clusters.

After clusters are trained, the cluster centers are used to measure the testing set accuracy. Distances from each sample in the testing set to all of the cluster centers are calculated. The label of the closest cluster is assigned to the sample and compared with the real label of the testing sample. A sample is correctly clustered if both labels are identical. The testing accuracy is the proportion of correctly clustered testing samples.

7.2.2 Results on Attribute Data Sets

Table 7.4 shows training and testing results of the four k-median clustering techniques applied on the UCI data sets [3]. Since the data sets are not separated into training set and testing set when they are given, ten replications of 5-fold cross validation are used to divide each data set into training and testing subsets. In the training phase, the three SPKM methods outperform the original k-median on six out of the seven data sets. Similarly, in the testing phase the SPKM methods get higher accuracy than that of the original k-median on most of the data sets. The SPKM has lower accuracy than the original k-median on only one data set, the PID. In addition, both training and testing results from the BPSPKM (denoted as LP in the table) algorithm is very close to the results from the IP formulation of SPKM algorithm. This means that the proposed bilinear program technique approximates the optimal solution of the IP problem very well. Observe also that the Image data set contains seven groups, which is very difficult to cluster. All k-median methods have less than 50% accuracy on the Image data set in both training and testing. FSSPKM gives the best performance on the Wine data set in both training and testing phase. Overall results indicate that the SPKM methods give higher classification accuracy. The sample-preserved medians represent the characteristics of the samples in the clusters very well.

Table 7.4: UCI training and testing accuracy (%) obtained from solving the original k-median, SPKM (IP formulation), BPSPKM (LP formulations) and FSSPKM clustering.

	Training				Testing			
Data Set	Orig	IP	LP	FSSPKM	Orig	IP	LP	FSSPKM
WDBC	62.1	93.3	93.3	94.0	93.3	93.3	93.3	93.1
BLD	57.7	58.0	58.0	58.0	55.0	58.0	58.0	58.0
HD	63.6	78.1	70.6	79.6	46.6	77.6	69.2	74.9
PID	93.4	66.1	65.5	65.4	95.3	65.5	64.7	64.6
Wine	88.5	93.2	90.6	96.2	87.8	93.1	91.8	95.0
Iris	76.9	90.3	89.6	89.1	74.9	88.3	90.1	89.3
Image	35.4	46.2	46.1	47.4	34.8	45.8	45.6	45.3

Orig: original k-median.

7.2.3 Results on Single Time Series Data Sets

Training and testing accuracies obtained from the three k-median clustering techniques on single time series data sets are summarized in Table 7.5. Three different similarity measures are used in the two SPKM methods. In the training phase, the dynamic time warping (DTW) distance strongly dominates the Euclidean (EU) and T-statistics (TS) similarity measures. The BPSPKM method again very well approximates the solution of the IP formulation of SPKM. The original k-median fails in most of the data sets. It achieves less than 50% accuracy on fourteen out of twenty data sets, and less than 5% accuracy on two of them. It is obvious that the original k-median method fails to cluster single time series data. It may be concluded that the SPKM methods are more adaptable than the original k-median on time series.

In the testing phase, the performance of the SPKM clustering with DTW distance is dramatically reduced, creating not as high accuracy as it has in the training phase. It especially has less than 10% accuracy on the "50Words" data set. Among the three similarity measures of the SPKM methods, Euclidean distance produces best accuracies among them all. T-statistics only dominate on two data sets, "Gun-Point" and "Trace". Five of the data sets have same testing accuracies from all clustering methods. This may indicate that the all samples are very well separated into groups so that all methods produce similar clusters. In addition, the BPSPKM again very well approximates the solution of the IP formulation of SPKM. Similar to the training results, the original k-median fails in most of data sets. Only on the "Synthetic Control" data set, the original k-median gives better accuracy (62.3%) than the SPKM methods (with the

Table 7.5: UCR training and testing accuracy (%) obtained from solving the original k-median, SPKM (IP formulation) and BPSPKM (LP formulations) clustering, with three distance measures for the two SPKM clustering methods.

Data Set	Training							Testing						
	Orig	IP EU	LP EU	IP DTW	LP DTW	IP TS	LP TS	Orig	IP EU	LP EU	IP DTW	LP DTW	IP TS	LP TS
Synthetic Control	64.3	50.0	57.7	99.0	99.0	50.7	40.3	62.3	50.0	54.7	46.3	46.3	49.3	33.0
Gun-Point	48.0	56.0	56.0	52.0	52.0	54.0	54.0	48.0	48.0	48.0	49.3	50.7	60.0	60.0
CBF	56.4	66.7	60.0	96.7	66.7	73.3	63.3	56.4	67.1	57.7	44.6	35.2	62.4	51.8
Face (all)	4.2	46.3	41.1	68.0	70.5	34.8	32.7	4.3	34.0	36.0	15.5	10.6	22.4	24.5
OSU Leaf	13.2	40.0	41.0	50.5	46.0	34.5	33.0	13.2	33.5	26.9	26.0	26.0	24.4	23.1
Swedish Leaf	32.2	50.4	41.8	46.8	47.8	27.6	20.4	32.2	50.6	40.3	19.7	22.2	19.7	15.4
50Words	12.5	59.8	56.4	61.3	57.1	39.3	34.9	12.5	50.8	47.0	9.2	8.1	29.0	26.2
Trace	43.0	58.0	58.0	79.0	79.0	58.0	58.0	43.0	43.0	43.0	42.0	42.0	51.0	51.0
Two Patterns	25.9	34.1	32.6	97.4	97.4	30.3	31.6	25.9	33.3	32.6	24.5	24.5	27.1	29.1
Wafer	89.2	90.3	90.3	90.3	90.3	90.3	90.3	89.2	89.2	89.2	89.2	89.2	89.2	89.2
Face (four)	38.6	66.7	75.0	58.3	54.2	79.2	70.8	38.6	56.8	34.1	29.6	31.8	46.6	51.1
Lightning-2	54.1	66.7	66.7	66.7	66.7	66.7	66.7	54.1	54.1	54.1	54.1	54.1	54.1	54.1
Lightning-7	42.5	54.3	51.4	64.3	52.9	45.7	42.9	42.5	43.8	42.5	27.4	13.7	32.9	26.0
ECG	64.0	69.0	69.0	69.0	69.0	69.0	69.0	64.0	64.0	64.0	64.0	64.0	64.0	64.0
Adiac	1.8	44.4	45.4	45.9	44.9	31.3	28.5	1.8	38.1	31.0	10.5	9.5	20.2	18.4
Yoga	53.6	54.3	54.3	54.3	54.3	54.3	54.3	53.6	53.6	53.6	53.6	53.6	53.6	53.6
Fish	21.1	46.9	44.0	52.6	57.1	36.0	35.4	21.1	40.0	36.0	26.9	29.7	24.0	34.9
Beef	40.0	56.7	60.0	56.7	63.3	53.3	56.7	40.0	43.3	43.3	43.3	40.0	30.0	33.3
Coffee	46.4	50.0	50.0	50.0	50.0	50.0	50.0	46.4	46.4	46.4	46.4	46.4	46.4	46.4
OliveOil	40.0	80.0	76.7	73.3	76.7	53.3	60.0	40.0	70.0	70.0	26.7	33.3	50.0	43.3

highest accuracy 54.7%). The overall results further strengthen the implication that the SPKM methods with Euclidean distances are more adaptable than the original k-median method on clustering time series.

7.2.4 Results on Multivariate Time Series Data Sets

Training and testing performances of the four k-median methods on multivariate time series data sets are shown in Table 7.6. Again, since the data sets are not separated into training set and testing set when they are given, ten replications of 5-fold cross validation are used to divide each data set into training and testing subsets. In the training phase, Euclidean distance achieves the highest accuracy among the three distance measures used in the SPKM methods, though not significantly higher than the others. The BPSPKM method also very well approximates the solution of the IP formulation of the SPKM. The original k-median achieves 50% accuracy on eight out of the ten data sets with the highest result being 65.3% of the training accuracy. Since both positive and negative samples are equally distributed, 50% accuracy may imply that the original k-median method tends to cluster all samples in one group, which points to the conclusion that the original k-median fails at clustering multivariate time series. The best training accuracy of the proposed BPSPKM method with Euclidean is 77.4% on the P2 data set.

In the the testing phase, the performances of the two SPKM methods using Euclidean distance are very close to their performances in the training phase. Among the three similarity measures, Euclidean distance again achieves higher accuracies than the other two on six out of the ten data sets. The second best distance measure is the DTW distance, which does not perform well on single time series data sets, but produces the highest accuracies among the three distance measures on four data sets. Data sets P4 and P5 are difficult data sets, but T-statistics achieves the highest accuracies on them. Similar to the training phase, the original k-median method gives 50% testing accuracy on most data sets, which means it fails to cluster multivariate time series. Again, the proposed BPSPKM method with Euclidean distance is suggested. The best performance in the testing is BPSPKM with Euclidean on the P2 data set, which has

Table 7.6: EEG training and testing accuracy (%) obtained from solving the original k-median, SPKM (IP formulation), BPSPKM (LP formulations) and FSSPKM clustering, with three distance measures for the two SPKM clustering methods.

	Training									
Data Set	Orig	IP EU	LP EU	FSSPKM EU	IP DTW	LP DTW	FSSPKM DTW	IP TS	LP TS	FSSPKM TS
P1	50.0	77.4	73.0	76.6	70.2	71.8	78.2	55.2	59.5	63.9
P2	50.0	76.6	77.4	75.8	75.4	74.2	73.8	73.1	76.2	72.2
P3	50.0	51.7	54.2	51.8	51.1	51.3	53.6	51.2	52.3	52.1
P4	50.0	56.1	55.0	53.8	52.9	53.1	52.4	54.0	53.7	53.7
P5	50.0	52.1	51.9	51.7	54.4	53.0	52.3	53.1	55.2	56.0
P6	50.0	67.1	56.5	61.3	70.1	70.1	70.0	65.7	62.3	56.0
P7	50.0	62.2	56.5	54.2	62.2	58.0	58.7	52.1	51.8	51.6
P8	50.0	54.9	53.7	53.5	53.2	52.5	54.5	51.9	51.9	53.6
P9	53.8	54.5	53.5	52.3	53.6	51.1	51.4	55.9	52.2	52.1
P10	65.3	66.0	69.1	65.2	66.4	64.0	67.6	55.7	52.5	52.7
	Testing									
Data Set	Orig	IP EU	LP EU	FSSPKM EU	IP DTW	LP DTW	FSSPKM DTW	IP TS	LP TS	FSSPKM TS
P1	50.0	73.8	76.2	73.8	45.2	66.7	69.0	50.0	57.1	50.0
P2	50.0	76.2	81.0	73.8	69.0	76.2	71.4	76.2	71.4	59.5
P3	50.0	46.2	56.8	43.2	37.9	40.2	51.5	31.8	31.1	43.9
P4	50.0	44.1	47.1	54.9	54.9	55.9	50.0	60.8	53.9	48.0
P5	50.0	40.2	46.1	32.4	44.1	31.4	55.9	53.9	51.0	52.0
P6	50.0	66.7	59.3	61.1	66.7	66.7	64.8	64.8	63.0	53.7
P7	50.0	61.6	55.8	54.3	61.6	61.6	59.4	50.7	52.2	39.1
P8	50.0	57.0	61.4	51.8	63.2	57.0	50.9	46.5	46.5	53.5
P9	54.2	63.3	51.7	45.8	50.8	42.5	47.5	48.3	46.7	47.5
P10	62.5	65.3	61.1	59.7	68.1	48.6	58.3	54.2	50.0	50.0

81% accuracy.

7.2.5 Computational Time

Table 7.7, Table 7.8 and Table 7.9 display the average clustering computation time of the three SPKM algorithms (IP, LP and non-LP) on the three different types of data sets (attribute, single time series and multivariate time series). All programs are written in MATLAB. All optimization models in the IP and LP formulations are solved by GAMS from the MATLAB and GAMS interface [37].

It can be observed that it takes time to pass parameters through the MATLAB and GAMS interface for solving the optimization problems. The IP formulation of the SPKM algorithm only has one optimization problem, while the BPSPKM algorithm needs to solve two LP problems alternately until convergence. Theoretically, the LP problems are easier to solve than the IP problems. However, since the approximation algorithm needs to solve the bilinear programs alternately, it requires parameters passing through MATLAB and GAMS alternately as well. This results in a long time for

Table 7.7: Average Clustering Time (in seconds) on UCI Data Sets

Data Set	IP	LP	Non-LP
WDBC	109.4	305.1	6.4
BLD	42.1	66.0	2.2
HD	32.8	50.7	1.6
PID	255.4	377.4	12.1
WR	8.1	22.3	0.6
IP	5.7	19.6	0.4
IS	5493.1	5286.4	68.4

Table 7.8: Average Clustering Time (in seconds) on UCR Data Sets

	EU			DTW			TS		
Data Set	IP	LP	Non-LP	IP	LP	Non-LP	IP	LP	Non-LP
Syn. Control	30.7	108.0	0.008	31.0	124.6	0.011	217.4	25.0	0.004
Gun-Point	0.9	3.1	0.001	0.9	3.7	0.002	0.9	2.9	0.001
CBF	0.4	1.8	0.001	0.4	2.0	0.001	0.4	1.1	0.001
Face (all)	438.1	485.3	0.026	584.7	455.5	0.024	585.6	190.5	0.020
OSU Leaf	34.7	33.9	0.004	128.8	44.9	0.006	187.8	23.0	0.004
Swedish Leaf	90.9	311.0	0.019	486.7	510.3	0.032	252.2	148.3	0.018
50Words	235.4	407.9	0.118	314.7	390.2	0.324	347.1	154.0	0.048
Trace	3.2	15.2	0.003	3.2	12.0	0.002	3.4	15.1	0.005
Two Patterns	3910.0	1462.6	0.053	653.6	1686.3	0.066	60333.9	601.4	0.029
Wafer	914.5	1719.2	0.075	818.7	1666.1	0.072	1054.2	1440.1	0.075
Face (four)	0.3	1.3	0.001	0.3	1.3	0.002	0.4	0.9	0.001
Lightning-2	1.2	5.4	0.001	1.2	4.9	0.002	1.3	3.8	0.002
Lightning-7	1.6	10.4	0.003	1.6	8.5	0.004	1.6	7.1	0.002
ECG	3.3	9.9	0.001	3.4	8.8	0.002	3.3	3.3	0.001
Adiac	173.0	438.8	0.042	393.7	263.7	0.025	111.1	232.5	0.024
Yoga	41.9	96.8	0.008	35.7	168.0	0.013	37.8	77.2	0.008
Fish	16.3	46.7	0.006	32.4	26.4	0.004	21.2	18.2	0.003
Beef	0.4	1.9	0.001	0.4	1.6	0.002	0.4	1.7	0.001
Coffee	0.4	1.6	0.001	0.4	1.4	0.002	0.4	1.5	0.001
OliveOil	0.4	2.3	0.001	0.4	2.1	0.001	0.5	1.1	0.001

LP algorithm to converge to a local optimal solution. Hence, the clustering time of the LP algorithm is longer than that of the IP algorithm in most cases.

All three tables also show that the equivalent non-LP sequential search SPKM algorithm takes dramatically less computational time than the other two algorithms. It is because it does not have any optimization formulation that needs to be solved by the MATLAB and GAMS interface, and hence no parameters need to be passed between MATLAB and GAMS. It gives the same solution as the LP algorithm in a more efficient way.

7.2.6 Conclusion on BPSPKM and FSSPKM Clustering Methods

A new approximation algorithm for the sample-preserved k-median (SPKM) clustering is proposed, called Bilinear Program Sample-Preserved k-Median (BPSPKM) clustering algorithm. The cluster medians obtained by SPKM clustering are from existing samples

Table 7.9: Average Clustering Time (in seconds) on EEG Data Sets

	EU			DTW			TS		
Data Set	IP	LP	Non-LP	IP	LP	Non-LP	IP	LP	Non-LP
1	0.5	2.7	0.096	0.6	2.4	0.044	0.6	2.9	0.052
2	0.5	3.3	0.001	0.6	3.3	0.001	0.6	2.1	0.001
3	5.4	25.0	0.005	6.3	20.5	0.003	5.6	17.2	0.003
4	3.0	12.1	0.003	3.1	9.9	0.002	3.0	11.3	0.002
5	3.0	14.2	0.004	3.0	13.0	0.003	3.2	8.8	0.002
6	0.8	3.4	0.002	0.8	4.4	0.002	1.0	3.0	0.001
7	6.0	22.1	0.004	6.4	21.2	0.003	6.1	16.1	0.002
8	3.8	13.7	0.003	3.8	11.3	0.002	4.1	11.1	0.002
9	4.5	15.9	0.004	4.4	13.1	0.002	4.7	14.7	0.002
10	1.5	7.5	0.003	1.5	7.9	0.002	1.5	5.1	0.001

whereas the cluster medians obtained by the original k-median clustering methods may not be existing samples. The original k-median clustering is not a sample-preserved method and is limited to 1-norm distance measure. Because the SPKM clustering method incorporates a pre-calculated sample-to-sample distance matrix, it provides the flexibility of choosing a distance measure. Hence, it can be easily applied on time series data, which need special type of similarity measures for a proper data analysis. However, the SPKM clustering problem is formulated as an integer programming problem and is hard to solve. Although there exist several approximation algorithms for solving the IP formulation of SPKM method, they are not as simple and easy as the bilinear program algorithm that is used to solve the original k-median problem. The proposed BPSPKM clustering algorithm combines the two major benefits. It is sample-preserved and contains a pre-calculated distance matrix. It can also be solved by a similar bilinear program algorithm and provides binary solutions. An equivalent non-LP sequential search SPKM clustering algorithm is also proposed, which provides the same solution as the BPSPKM and can be solved even more efficiently.

An extension of the BPSPKM clustering is the FSSPKM clustering method. Instead of using a sample-to-sample distance matrix in the whole m-dimensional space, the FSSPKM uses sample-to-sample distance matrices from each of the m feature spaces. It can also be solved by a similar bilinear program algorithm, and comes with an equivalent non-LP sequential search algorithm as well.

The computational time of the proposed equivalent non-LP SPKM clustering algorithm is far lower than those of the other two SPKM methods, which need to solve IP and LP formulations. Moreover, experimental results show that the original k-median only works well on attribute data sets, and fails to cluster time series data. Both the SPKM and BPSPKM achieved very good performance on attribute, single time series and multivariate time series data sets. Results on multivariate data sets also show that the performance of FSSPKM clustering is comparable to the BPSPKM clustering.

7.3 Experimental Results Using FSSPKM with Feature Selection

The proposed feature selection approaches for the feature space sample-preserved k-median (FSSPKM) clustering are tested on four publicly available attribute data sets from UCI database [3] and on ten Electroencephalography (EEG) multivariate time series data sets that has been used in [19, 20]. The sample size (n), number of features (m) and number of classes (k) of each data set are shown in Table 7.1 for attribute data sets and in Table 7.3 for EEG multivariate time series data sets.

7.3.1 Feature Selection Error Rates

Two error curves, a classification error curve and a clustering error curve, are generated for performance evaluation. These error curves are also used in [71, 26]. The entire data set is used for evaluating both the classification error and clustering error. The clustering algorithm is repeated 50 times with a random initialization. The classification error is the percentage of incorrectly classified samples. The majority of a true class label in a cluster determines the estimated label of that cluster. The estimated labels and the true labels are then compared to give correctness. The true class labels are used only in obtaining the classification error and not in generating the clusters. The clustering error is calculated in the same way, except that instead of the true class labels, the gold labels obtained from the clustering results when using all features are used. By this definition, the clustering error on the data set containing all features is zero.

The lowest classification error rates and their corresponding number of features obtained from the four proposed feature selection criteria in FSSPKM are shown in Table 7.10 and Table 7.11. Both FSS and BSS feature selection algortihms are used, and are compared with the FSSPKM clustering without any feature selection. Table 7.10 contains results on the on the UCI data sets and Table 7.11 EEG multivariate time series data sets.

Table 7.10: Classification error rates and lowest error rates with corresponding optimal number of features obtained from experiments on UCI data sets. Methods used are the FSSPKM clustering and both FSS and BSS feature selection algorithms with all four feature selection criteria. Error rates are averages of 50 random runs.

	FSSPKM only		FSSPKM with Feature Selection							
			MM		α		β		γ	
	Error (%)	# of features	Error (%)	# of features	Error (%)	# of features	Error (%)	# of features	Error (%)	# of features
FSS										
WDBC	6.3	30	5.0	15	6.0	25	5.3	23	5.3	22
BLD	42.0	6	42.0	Any	42.0	Any	42.0	Any	42.0	Any
HD	21.7	13	21.3	9	21.6	12	21.3	9	21.1	10
PID	34.7	8	34.2	3	28.1	1	34.5	2	33.7	2
Wine	3.9	13	3.9	13	3.9	13	3.9	13	3.9	13
Iris	13.6	4	6.4	1	9.5	3	9.2	1	6.3	1
BSS										
WDBC	6.3	30	4.9	15	5.5	22	5.3	23	5.3	16
BLD	42.0	6	42.0	Any	42.0	Any	42.0	Any	42.0	Any
HD	21.7	13	21.3	10	21.3	10	21.3	10	21.3	10
PID	34.7	8	34.6	3	34.7	8	34.7	8	33.8	3
Wine	3.9	13	3.8	7	3.9	13	3.9	13	3.9	13
Iris	13.6	4	7.0	1	10.1	1	10.1	1	7.5	1

Any: Any number of features: $\{1, 2, \ldots, m\}$.

Overall, feature selection with α-ratio gives very close results to those given by β-ratio on almost all data sets. In addition, the MM and γ-ratio give similar error rates. The best result for WDBC data set is using the BSS MM feature selection, which reduces the error rate from 6.3% to 4.9% with half of the features used. Feature selection does not improve the classification performance in BLD and HD data sets. Specifically, in BLD data set the error rates are all the same no matter what numbers of features are used. The error rate for HD data set can only be decreased from 21.7% to 21.1% the most by FSS γ-ratio feature selection. The only improvement for Wine data set is made by BSS MM feature selection, which eliminate six features but only decrease the error rate from 3.9% to 3.8%. However, feature selection for HD and Wine data sets is not totally valueless. It indicates that with a smaller number of features, the HD and Wine data set can be classified with a similar error.

Very different results are from the performances in the PID data set, which is known to be difficult to classify [11]. The only best performance is made by the FSS α-ratio feature selection, which decrease the error rate from 33.7% to 28.1% with only one feature used. All feature selection methods in FSSPKM significantly improves the classification on Iris data set. The best result is by using FSS γ-ratio, which reduces the error rate from 13.6% to 6.3% with only one feature used.

Table 7.11: Classification error rates and lowest error rates with corresponding optimal number of features obtained from experiments on EEG data sets. Methods used are the FSSPKM clustering and both FSS and BSS feature selection algorithms with all four feature selection criteria. Error rates are averages of 50 random runs.

	FSSPKM only		FSSPKM with Feature Selection							
			MM		α		β		γ	
	Error (%)	# of features	Error (%)	# of features	Error (%)	# of features	Error (%)	# of features	Error (%)	# of features
FSS										
P1	23.9	26	20.8	20	20.8	23	21.2	32	20.9	22
P2	24.4	26	17.8	5	20.0	6	19.9	6	20.2	12
P3	48.6	26	45.0	1	44.0	1	44.2	1	45.4	1
P4	47.1	26	43.3	18	43.3	20	43.5	21	43.3	17
P5	48.0	26	44.8	3	43.2	1	45.3	7	45.5	1
P6	33.7	26	32.6	20	32.9	22	32.7	23	33.6	24
P7	47.0	26	38.4	1	41.9	1	42.3	15	42.1	1
P8	45.6	26	38.1	3	41.3	1	41.2	1	41.1	2
P9	46.9	26	46.3	5	44.7	1	44.7	1	46.3	6
P10	34.2	26	34.2	20	34.1	23	34.2	21	34.1	23
BSS										
P1	23.9	26	20.8	23	20.8	23	20.8	23	20.8	23
P2	24.4	26	10.2	1	10.2	1	10.2	1	9.5	3
P3	48.6	26	39.3	3	43.7	1	37.8	1	43.3	1
P4	47.1	26	43.5	17	43.1	14	43.1	14	43.1	19
P5	48.0	26	45.5	7	45.7	7	45.7	7	47.4	5
P6	33.7	26	33.7	26	33.4	25	33.4	24	33.7	26
P7	47.0	26	36.3	1	36.3	1	35.4	2	41.1	6
P8	45.6	26	37.6	4	41.7	1	37.9	4	38.8	4
P9	46.9	26	46.9	26	46.9	26	46.9	26	46.3	2
P10	34.2	26	34.2	26	34.1	22	34.1	20	34.0	14

Results show that the EEG data sets are very difficult to cluster. The proposed feature selection methods work very well only on the P2 EEG data set. The best results for the P2 EEG data set is BSS γ-ratio feature selection. Its error rate decreases from 24.4% to 9.5% with only three features used. Observe that both FSS and BSS produce comparable results, except that on the P2 EEG data set, the BSS methods dominant the FSS methods. It is also worth noting that with feature selection, the results made by FSSPKM clustering are improved on almost all data sets. Only on P6, P9 and P10 EEG data sets no significant improvements occurred with feature selection.

7.3.2 Classification and Clustering Error Curves

The classification error and the clustering error in relation to the number of remaining features are shown in the error curves, as well as one sample standard deviation above and below each value. Note that the real use of the FSSPKM with feature selection is to unlabeled data, therefore only a clustering error curve can be generated and then a tolerable magnitude of error can be chosen by deciding the number of features to keep [71]. However, by having the classification error curve compared with the clustering error curve, possible improvement made by feature selection can be observed. According to the definition of the classification error and clustering error, if both classification error curve and clustering error curve are parallel to each other at some numbers of features used, it means that the gold labels (obtained from clustering using all features) and the real labels in the clusters are quite consistent at the chosen numbers of features. In this case, one may choose a number of features that gives an acceptable error rate. However, if the classification error curve is not parallel with the clustering error curve, it means that some of the features contain a lot of noise which makes the resulting gold label not able to represent the data well. In such a case, feature selection can actually improve the classification. One can see that the classification error significantly decreases while the number of features decreases.

Figure 7.1 displays the error curves from the four proposed FSS feature selection methods on the Wine data set, and Figure 7.2 displays the error curves from the BSS feature selection methods. The FSS methods start with very high error rates when only few features are used. The error rates keep reducing while the number of features is increasing. The error rates generated by the FSS α-ratio behave very differently from those by the other three.

Similarly, in BSS both classification and clustering error rates increase dramatically when less than two or three features are used. However, Steady error rates are occurred in all four BSS feature selection methods when 8 to 13 features are used. This consistency ensure that one may choose only eight features rather than 13 features to have almost the same clustering performance. The results obtained from α-ratio are similar to that from the β-ratio, while the results obtained from MM are similar to that from the γ-ratio.

Error curves obtained from the experiments using the four proposed FSS feature selection methods on the PID data set are displayed in Figure 7.3, and error curves

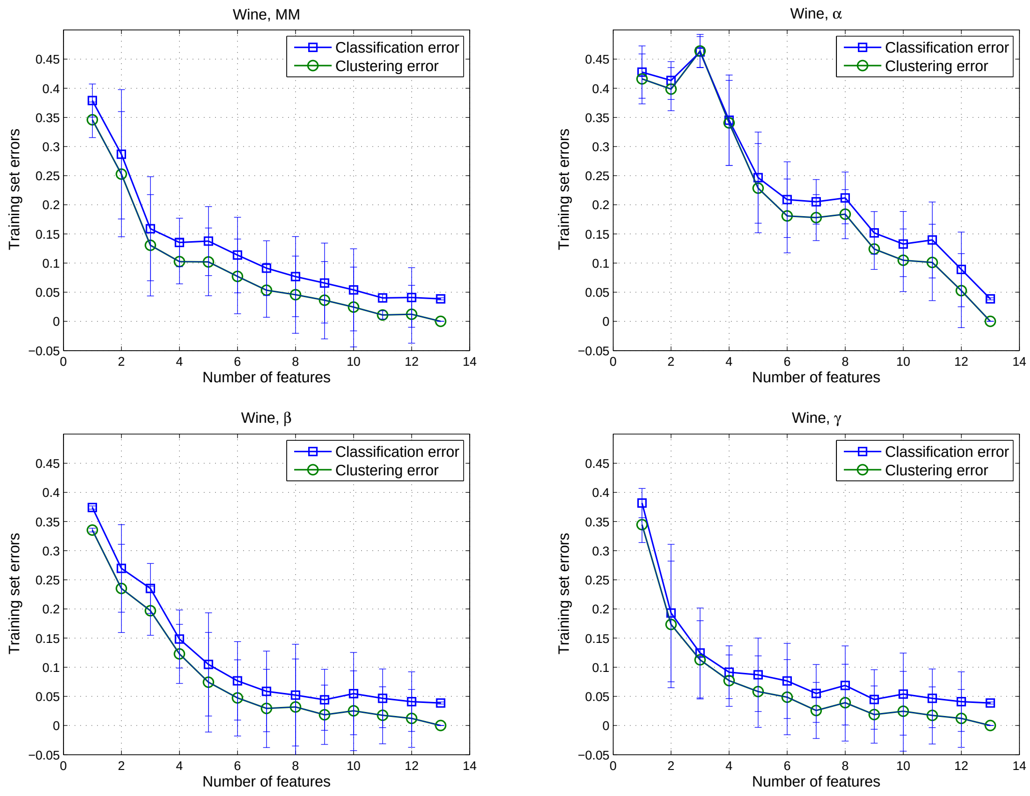


Figure 7.1: Error curves generated by FSS feature selection methods on the Wine data set

obtained from the BSS feature selection methods are displayed in Figure 7.4. The classification error rates of the FSS MM, β-ratio and γ-ratio are at around 35% no matter how many features are used. Only the FSS α-ratio gives the classification error at a very law 28.1% when using one feature. It may imply that the FSS α-ratio captures a critical feature which can help in classification. The high clustering error rates when less than six features are used indicate that the gold labels obtained when using all features do not match with the real labels of the samples. There exist some features that need to be eliminated in order to improve classification. In Figure 7.4, the BSS algorithms of all four feature selection criteria have almost the same classification errors around 35% no matter how many features are used.

Figure 7.5 shows the error curves of the FSS tests on the P1 EEG data set. It is obvious that the classification error curves are parallel with the clustering error curves

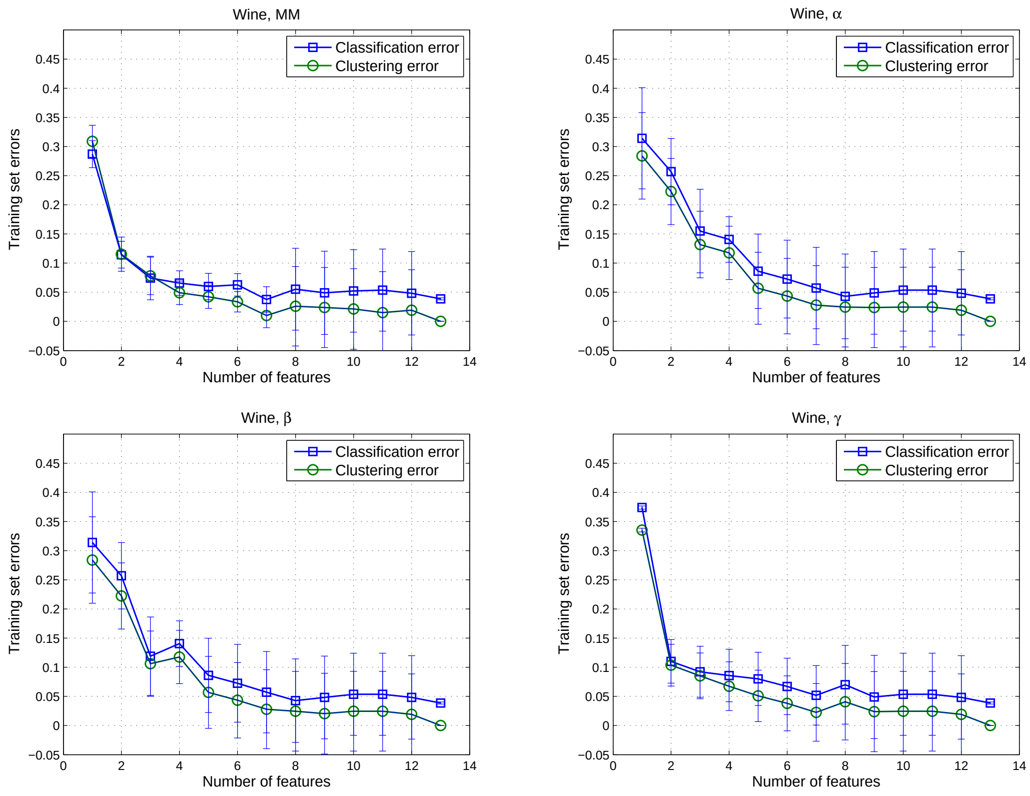


Figure 7.2: Error curves generated by BSS feature selection methods on the Wine data set

almost at any number of features. The standard deviations of both the classification and clustering errors are high. When more than 12-15 features are used, the error rates become more steady.

The results of the BSS tests on the P1 EEG data set are shown in Figure 7.6. Again, the classification error curves are parallel with the clustering error curves almost at all numbers of features. The standard deviations of both the classification and clustering errors are not as high as those generated by the FSS methods. In general, the average errors are not much different when about more than 10 features are used. These consistent results indicate that there exists no significant improvement in classification or clustering by selecting certain features. However, using fewer numbers of features may provide similar classification or clustering results.

Figure 7.7 shows the error curves from the four FSS feature selection methods on the

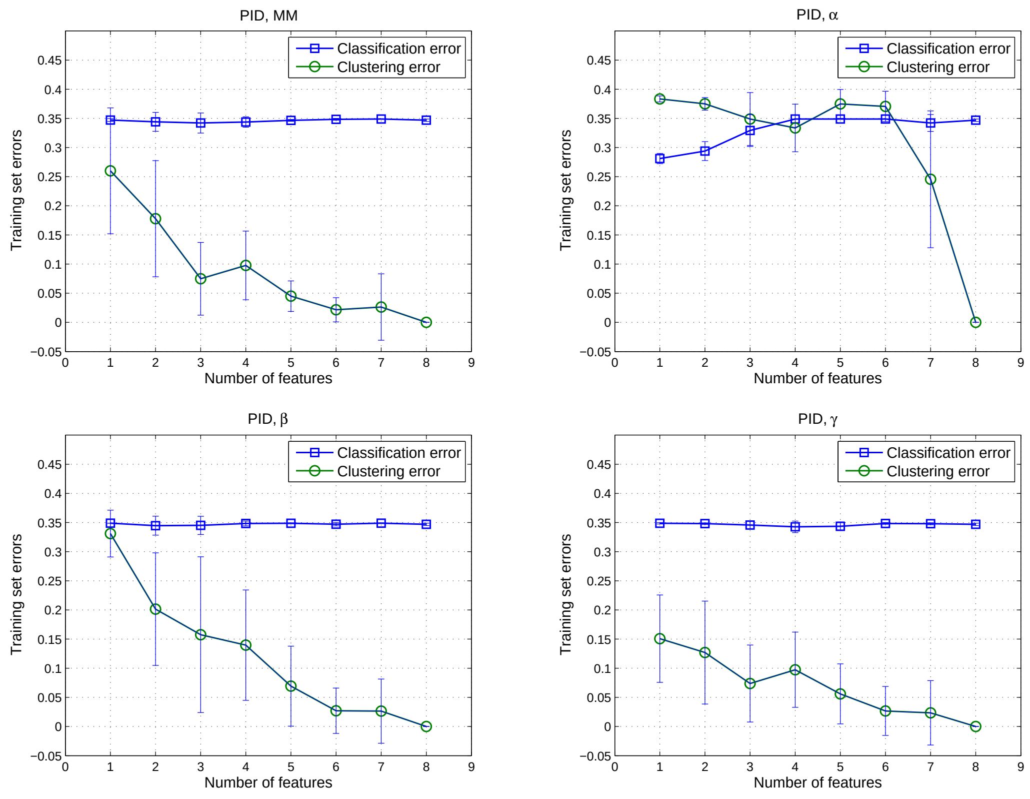


Figure 7.3: Error curves generated by FSS feature selection methods on the PID data set

P2 EEG data set. The classification error is decreasing when using one to five features, and is increasing when using six features. The classification error curves start to be constant and parallel with the clustering error curves when more than 10-15 features are used. This indicates that the gold labels and the real labels are significantly different and using fewer numbers of features can improve the classification.

The error curves generated by the BSS feature selection methods are shown in Figure 7.8. The classification and clustering error curves have very different behaviors from all previous curves. The MM method has a big gap between classification and clustering error curve when using from 15 to 26 features. Then the classification error decreases and the clustering error increases while the number of features decreases. Note that the classification error largely decreases and reaches its lowest when using only one feature. The α-ratio and β-ratio methods behave similarly and also reach their

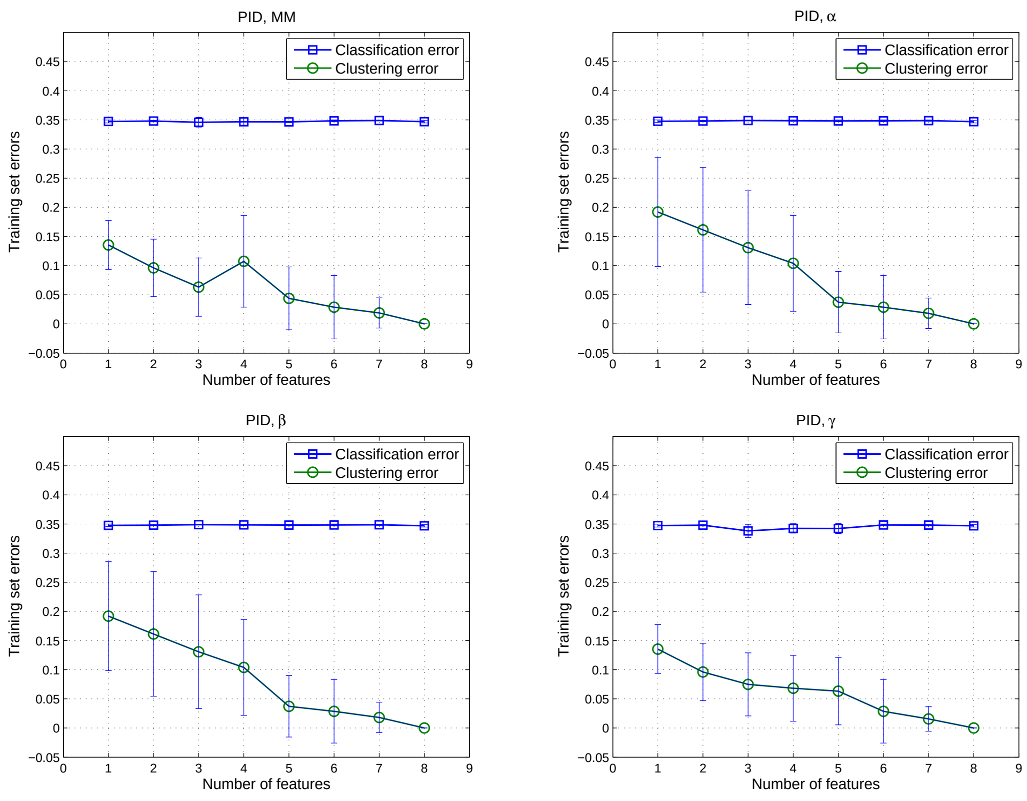


Figure 7.4: Error curves generated by BSS feature selection methods on the PID data set

lowest classification error when using one feature. Both α-ratio and β-ratio methods have their classification error decrease and clustering error increase while using less than 10 features. The γ-ratio reaches its lowest classification error when using three features. These results indicate that feature selection for FSSPKM clustering does improve the classification on the P2 EEG data set.

Overall results show that both FSS and BSS methods produce similar best performances. It may be suggested to prefer using BSS since fewer iterations are needed. It is also suggested to apply all four feature selection methods since every data set has different structure. No specific criterion can work for all data sets. If all methods give consistent results, one can be more confident on the conclusions. A complete collection of error curves for all data sets that are tested can be found in Appendix A for attribute data and Appendix B for EEG MTS data.

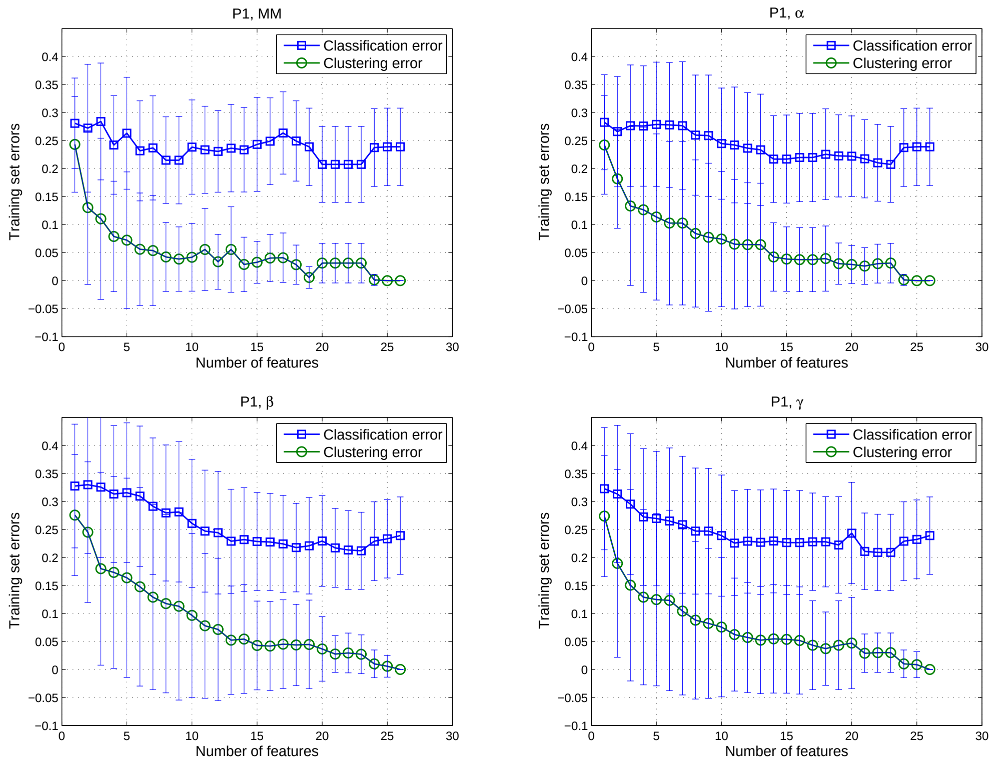


Figure 7.5: Error curves generated by FSS feature selection methods on the P1 EEG data set

7.3.3 Comparison with Other Methods

Table 7.12 shows the comparison of classification errors of the four BSS feature selection method with other feature selection clustering algorithms. In the LFJ method [59], both Wine and WDBC data sets are used, which give classification errors at 9.35% and 6.61%, respectively. Both errors are higher than all other methods. However, its method not only performs feature selection but also determines the best number of clusters needed. The M&W method [71] gives higher classification errors than the the proposed BSS MM method. Its clustering technique is different from the proposed FSSPKM. The M&W method uses the original k-median clustering without the sample-preserved property, which our FSSPKM clustering has.

The CAC method [26] gives higher error rates than the BSS MM method on the WDBC data set. Its lowest error is 6.2% using 16 features, while the BSS MM method's lowest error rate is 4.9% using 15 features. On the HD data set, the CAC method of

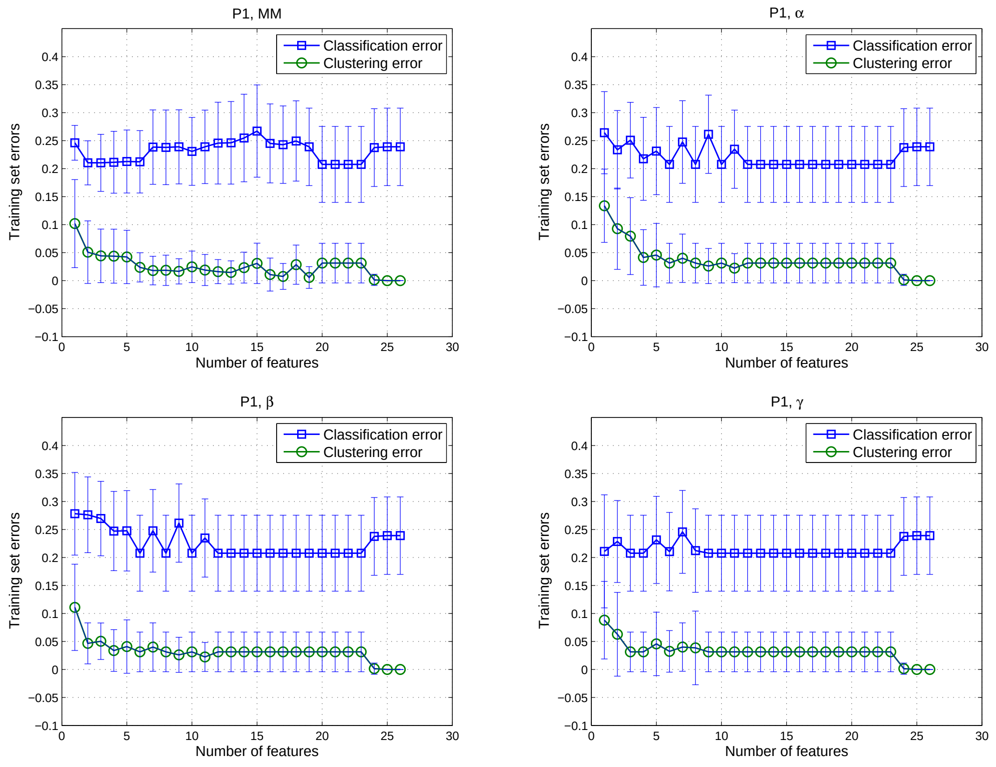


Figure 7.6: Error curves generated by BSS feature selection methods on the P1 EEG data set

gives an error rate of 30.5%, lower than 32% from the BSS MM method when only two features are used. However, the proposed BSS MM method can reduce the error rate to 21.3% if 10 features are used. On the Wine data set, even though the CAC method gives a lower error rate of 5.8% than 6.6% from the BSS MM method when four features are used. Both methods can give a lowest error of 3.8% if 7 features are used.

It is important to notice that all other three methods (from [26, 59, 71]) use the k-median clustering without the sample-preserved property. Our proposed FSSPKM algorithm forces the cluster medians to be one of the existing samples, which can best represent the samples in each cluster. Hence, when using the cluster medians to classify other samples, the FSSPKM with feature selection gives the best results. In summary, the proposed feature selection methods tend to need higher numbers of features when

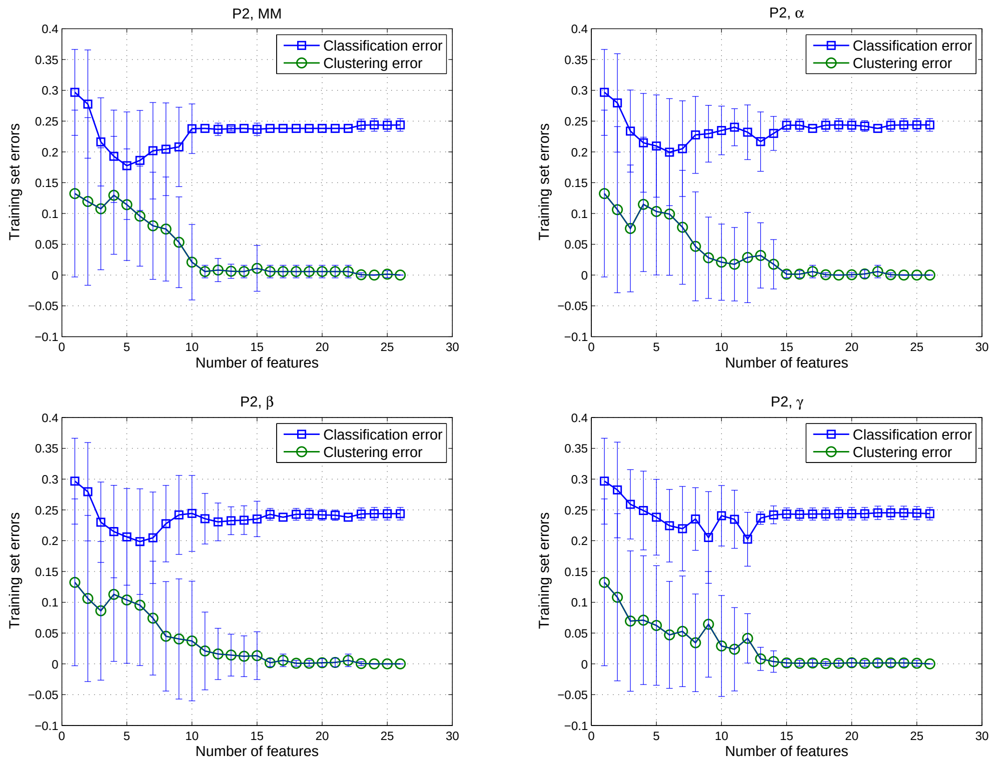


Figure 7.7: Error curves generated by FSS feature selection methods on the P2 EEG data set

reaching the lowest error rates in some cases. This may be the results of the sample-preserved property of our clustering algorithm.

7.3.4 Conclusion on Feature Selection in FSSPKM Clustering

A feature space sample-preserved k-median (FSSPKM) clustering algorithm along with its feature selection methods are proposed. The FSSPKM clustering is comparable to the original k-median except that the FSSPKM uses a sample-to-sample distance matrix at each feature space and each cluster median is a combination of the existing sample values in each feature space. This may be critical because not all values are valid in some application domains. The sample-preserved property ensures the existence of the sample values. Due to this property, the FSSPKM can also be easily applied on multivariate time-series data. Four feature selection criteria utilizing such distance matrices are tested on real world attribute data sets and multivariate time series data

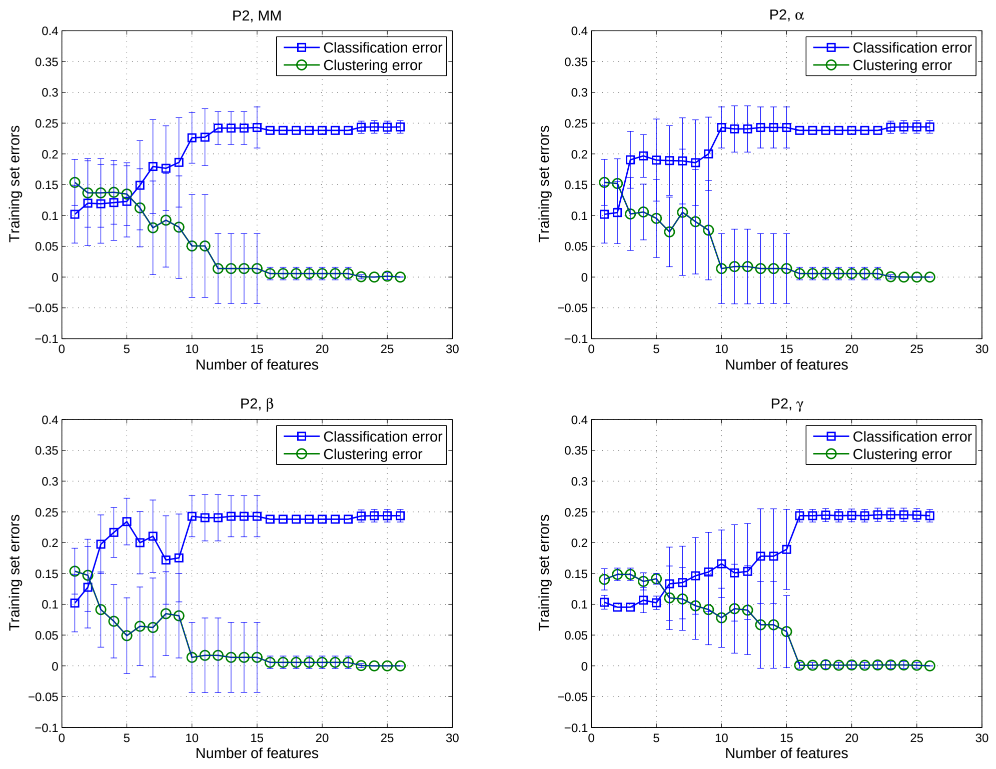


Figure 7.8: Error curves generated by BSS feature selection methods on the P2 EEG data set

sets. They are based on forward subset selection (FSS) and backward subset selection (BSS) approaches. Experimental results show that both methods produce similar best performances. BSS is suggested since it needs fewer iterations than what FSS requires. The behavior of both the classification error and the clustering error curves in relation to the number of remaining features indicates the possible classification improvements using the proposed feature selection criteria. For example, the BSS γ-ratio feature selection technique for the P2 multivariate time series data set gives classification error rate of 9.5% with three features compared with 24.4% with the full 26 features.

The proposed FSSPKM clustering algorithm and its feature selection methods have two major strengths. One is that even though the sample-preserved clustering property may tend to require more number of features in some cases in order to give smaller classification error rates, such property provides the lowest errors compared with other feature selection clustering methods which do not have the sample-preserved property.

Table 7.12: Comparison of the best performance classification errors with different approaches.

	LFJ[59]	M&W [71]		CAC[26]		BSS MM		α		β		γ	
	Error (%)	Error (%)	s	Error (%)	s	Error (%)	s	Error (%)	s	Error (%)	s	Error (%)	s
WDBC	9.35	-	-	8.8	3	7.9	3	11.6	3	11.6	3	10.4	3
	-	9.0	7	-	-	7.0	7	7.3	7	10.2	7	9.3	7
	-	-	-	8.8	10	7.9	10	9.1	10	7.9	10	8.6	10
	-	-	-	6.2	16	4.9	15	7.0	15	5.4	22	5.3	16
	-	-	-	6.4	22	5.3	22	5.5	22	5.3	23	5.3	22
HD	-	-	-	30.5	2	32.0	2	30.4	2	30.4	2	33.1	2
	-	28.0	8	-	-	22.2	8	22.2	8	22.2	8	22.2	8
	-	-	-	-	-	21.3	10	21.3	10	21.3	10	21.3	10
Wine	6.61	4.0	4	5.8	4	6.6	4	14.1	4	14.1	4	8.6	4
	-	-	-	3.8	7	3.8	7	5.7	7	5.7	7	5.2	7
	-	-	-	-	-	3.9	13	3.9	13	3.9	13	3.9	13

The second is that only the FSSPKM and its feature selection methods can be applied directly on multivariate time series data with flexibility of choosing distance measures. The proposed feature selection methods produce improved performance with FSSPKM, and outperform other feature selection techniques used in the original k-median clustering.

Chapter 8
FUTURE RESEARCH

Pattern recognition refers to the identification of individual characters according to a set of training patterns with class labels. Many opportunities can be found for developing original research in the pattern recognition area. These opportunities can be aimed at improving sample-preserved classification (or clustering). They can be focused on: extending the proposed techniques into stronger pattern-based methods, or constructing novel classification (or clustering) models aiming at different analytical purposes.

8.1 Identifying Sample-Preserved Patterns

Consider the Logical Analysis of Data (LAD) (described in Section 2.1.1) used to find positive and negative patterns in an attribute data set. Positive patterns contain values of selected features guaranteed to be found only in positive samples. Negative patterns can be found only in negative samples. Once found, a new sample can be classified into a positive or negative group more confidently. However, LAD can not handle multivariate time series (MTS) directly.

The proposed techniques may be extended into stronger pattern-based methods, such as identifying specific sample-preserved patterns in order to represent a specific group of data. These patterns may be a region in a subspace of the original sample space, and may be obtained by extending the feature space sample-preserved clustering. It may then be used for time series data and possibly improve the classification of MTS.

8.2 Regression Analysis

Sample-preserved methods consider relative distances (similarities) among samples to train a classifier, or to form clusters. One may develop a new model aiming at different analytical purposes, such as considering the distribution of variables in a multivariate data set.

Regression analysis is a technique focused on analyzing the relationship between a dependent variable and one or more independent variables. It constructs a regression function as a function of independent variables. There may exist a probability distribution that describes the variation of the dependent variable around the regression function. There may also exist a conditional distribution of the dependent variable given the independent variables. For example, the Likelihood Basis Pursuit (LBP) model (described in Section 2.1.3.1) determines the probabilities of binary outcomes given vectors of independent variables, while automatically selecting and prioritizing important features. Once a regression function is constructed, it can then be used for prediction and forecasting. Besides regression functions, there are autoregressive moving average (ARMA) models that have been used for analyzing stationary time series, and autoregressive integrated moving average (ARIMA) models for nonstationary time series.

MTS data contain both spacial and temporal properties. Since many regression functions have been designed for attribute data, and ARMA or ARIMA models can be built for time series data, the ideas of both techniques may be combined for the analysis of MTS. Certain distributions of variables may exist in a MTS data set. One may further investigate those distributions and construct regression analysis models for MTS sample prediction and forecasting.

8.3 Hierarchical Clustering for Multivariate Time Series

Unsupervised learning can be subdivided into hierarchical clustering and partitional clustering by the type of structure imposed on the data [47]. Partitional clustering is a single partition, whereas hierarchical clustering is a nested sequence of partitions.

Hierarchical clustering can be obtained from a sequence of partitional clusterings.

A clustering is a partition. Suppose a data set contains n samples. Let G_p denote the set of indices of samples that are assigned to cluster p for $p = 1, \ldots, k$. That is $G_p \subset \{1, 2, \ldots, n\}$ and $G_p \cap G_{p'} = \emptyset$ for joint clustered subsets $p \neq p'$ for p and $p' \in \{1, \ldots, k\}$. $G_1 \cup G_2 \cup \cdots \cup G_k = \{1, 2, \ldots, n\}$. The components of the the partition are called clusters. Partition E is nested into partition G if every component of E is a subset of a component of G. Consider an example of the following clusterings.

$$\begin{aligned} G &= \{(1,2),(3,5,7,9),(4,6,8,10)\}. \\ E &= \{(1,2),(3,5),(7,9),(4,6,8),(10)\}. \\ F &= \{(1,2),(3,4,5,6),(7,8,9,10)\}. \end{aligned}$$

In the example above, the clustering G and F have three clusters, while the clustering E has five clusters. The clustering E is nested into the clustering G. G is formed by combining components of E. Observe also that neither G nor E is nested into F, and F is not nested into G or E.

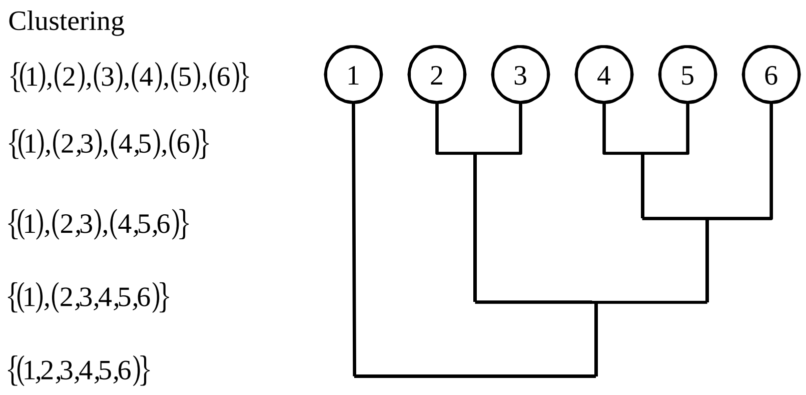


Figure 8.1: An example of a dendrogram.

A *hierarchical clustering* is a sequence of partitions where each partition is nested into the next partition in the sequence. A *dendrogram* is a tree diagram frequently used to provide a convenient view of a hierarchical clustering. Figure 8.1 shows an example of a dendrogram. Lines are used to connect nodes whose components are merged to create components of the next partition. Connected nodes represent clusters that are nested into the other. Graph theory has been largely applied for hierarchical clustering algorithms.

Clustering techniques used in this dissertation are partitional clustering techniques. No hierarchical relationships of clusters have constructed. There exist opportunities to explore a different area of clustering methodologies. One possible technique may be a hierarchical clustering algorithm for multivariate time series.

Appendix A

FSSPKM Feature Selection Error Curves Part 1: UCI Data Sets

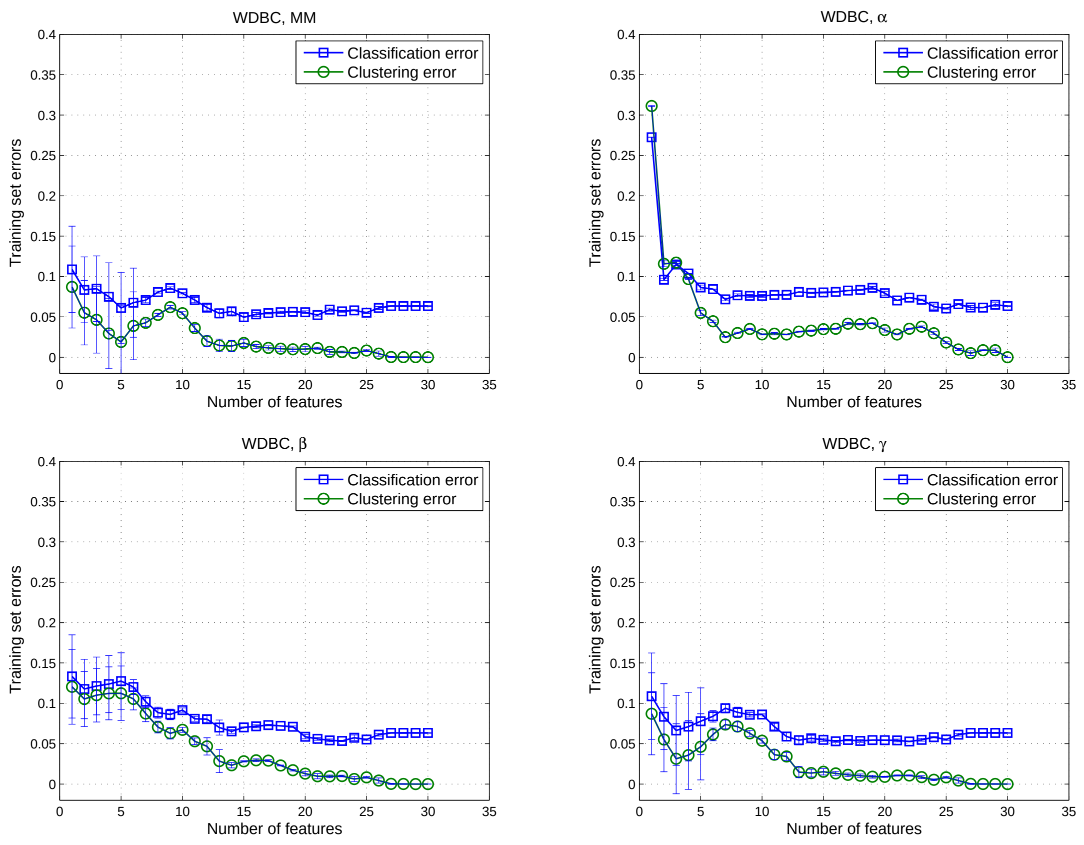


Figure A.1: Error curves generated by FSS feature selection methods on the WDBC data set

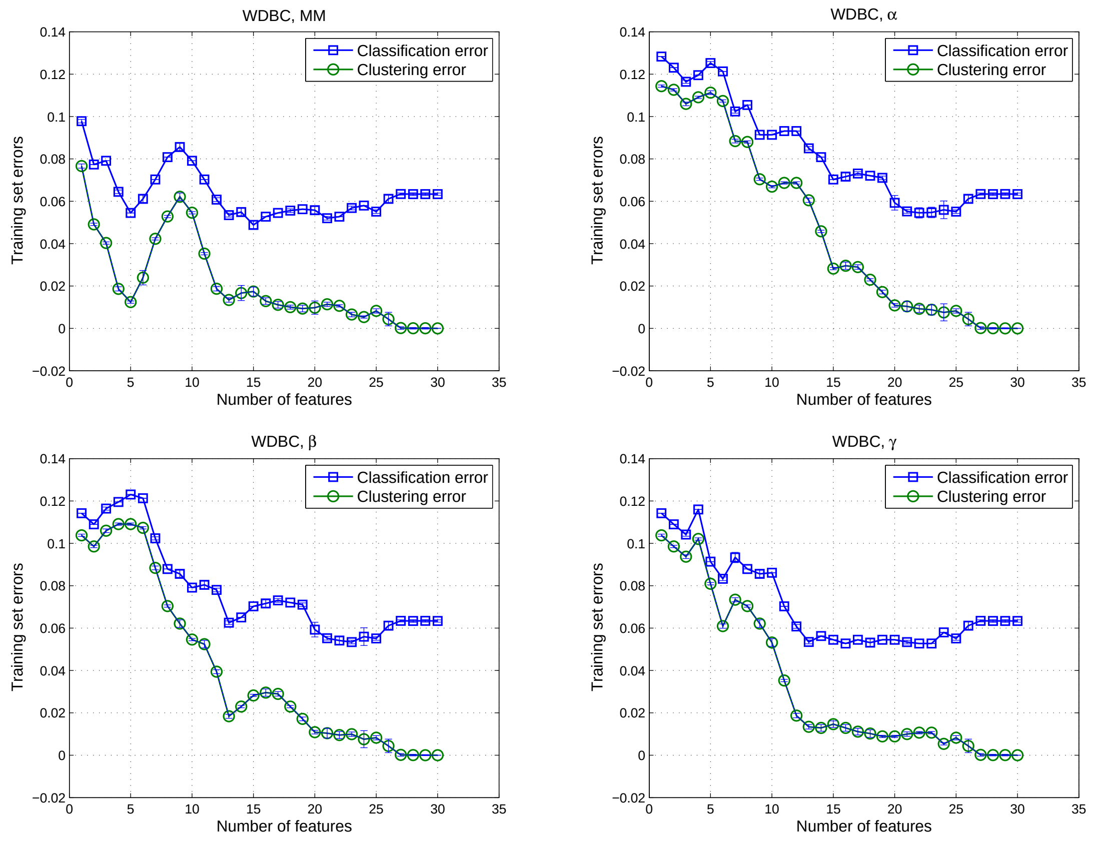


Figure A.2: Error curves generated by BSS feature selection methods on the WDBC data set

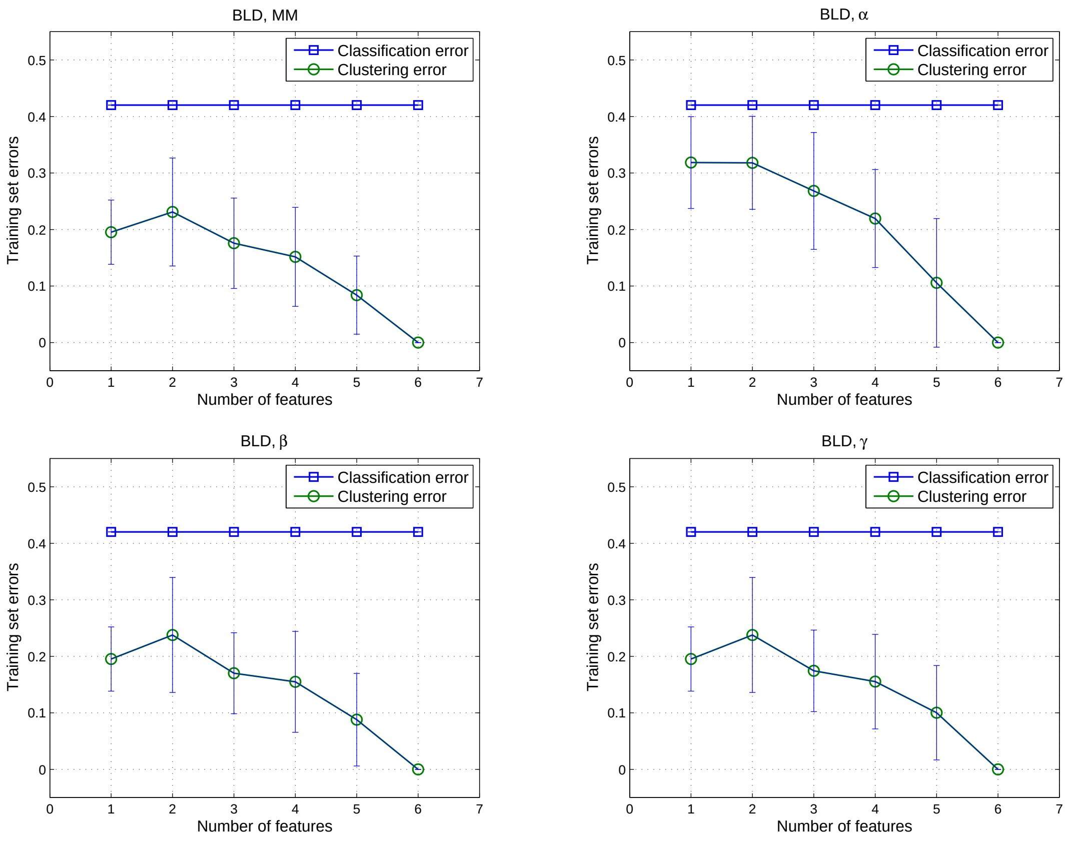


Figure A.3: Error curves generated by FSS feature selection methods on the BLD data set

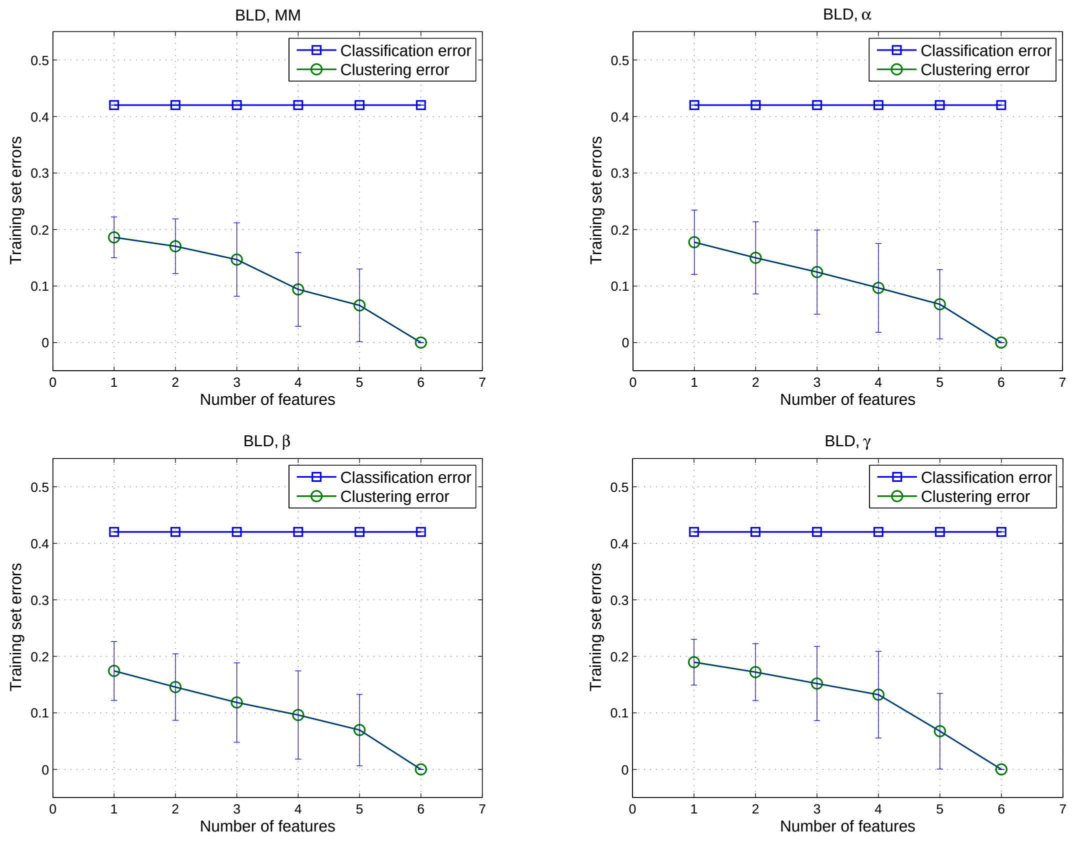


Figure A.4: Error curves generated by BSS feature selection methods on the BLD data set

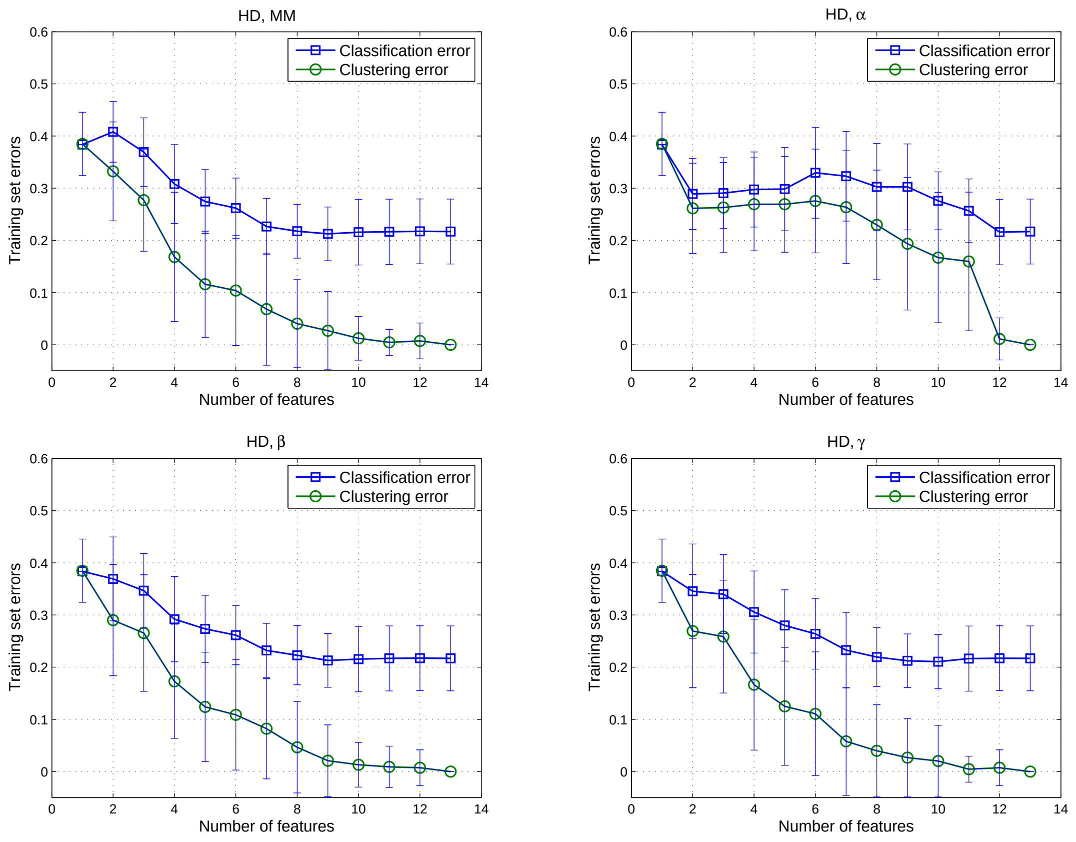


Figure A.5: Error curves generated by FSS feature selection methods on the HD data set

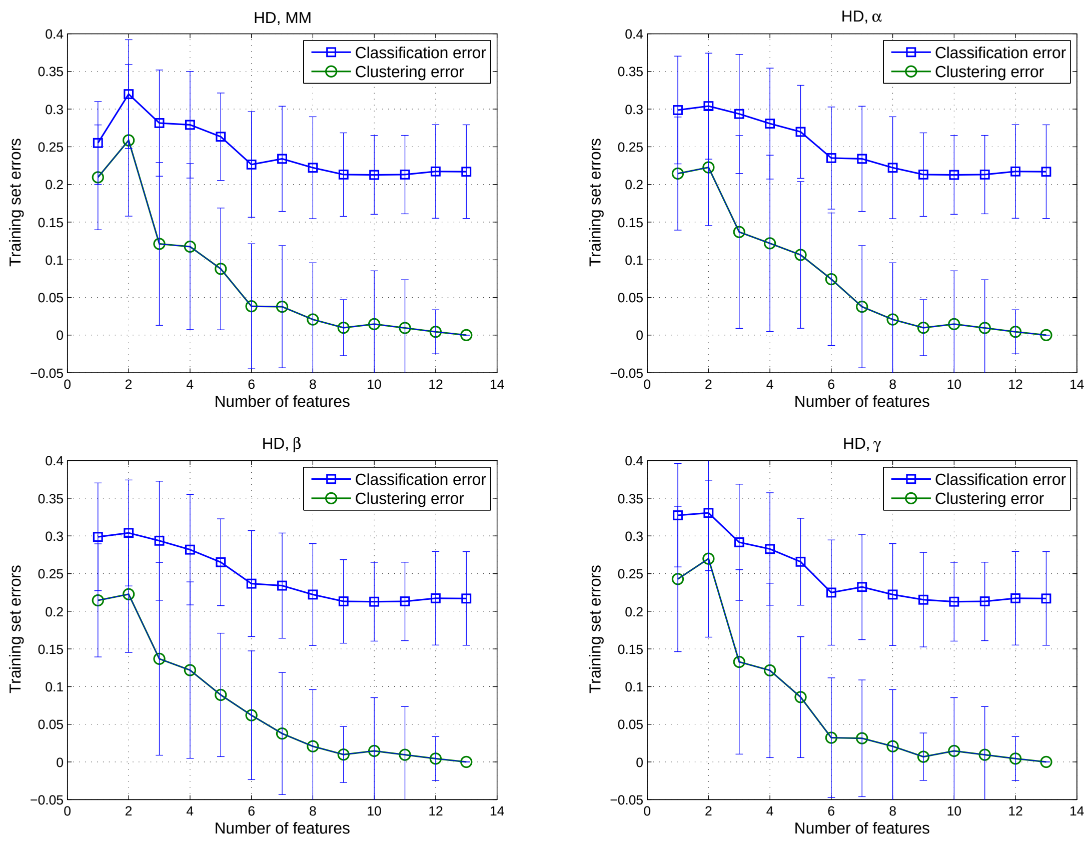


Figure A.6: Error curves generated by BSS feature selection methods on the HD data set

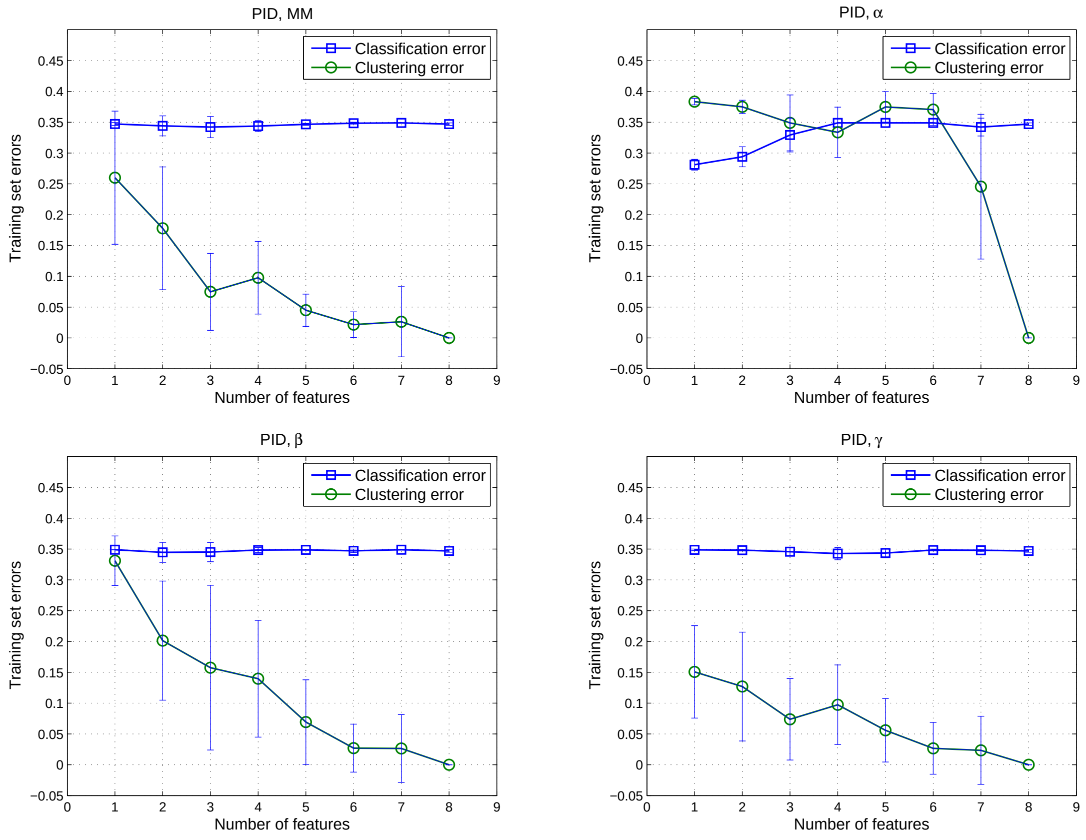


Figure A.7: Error curves generated by FSS feature selection methods on the PID data set

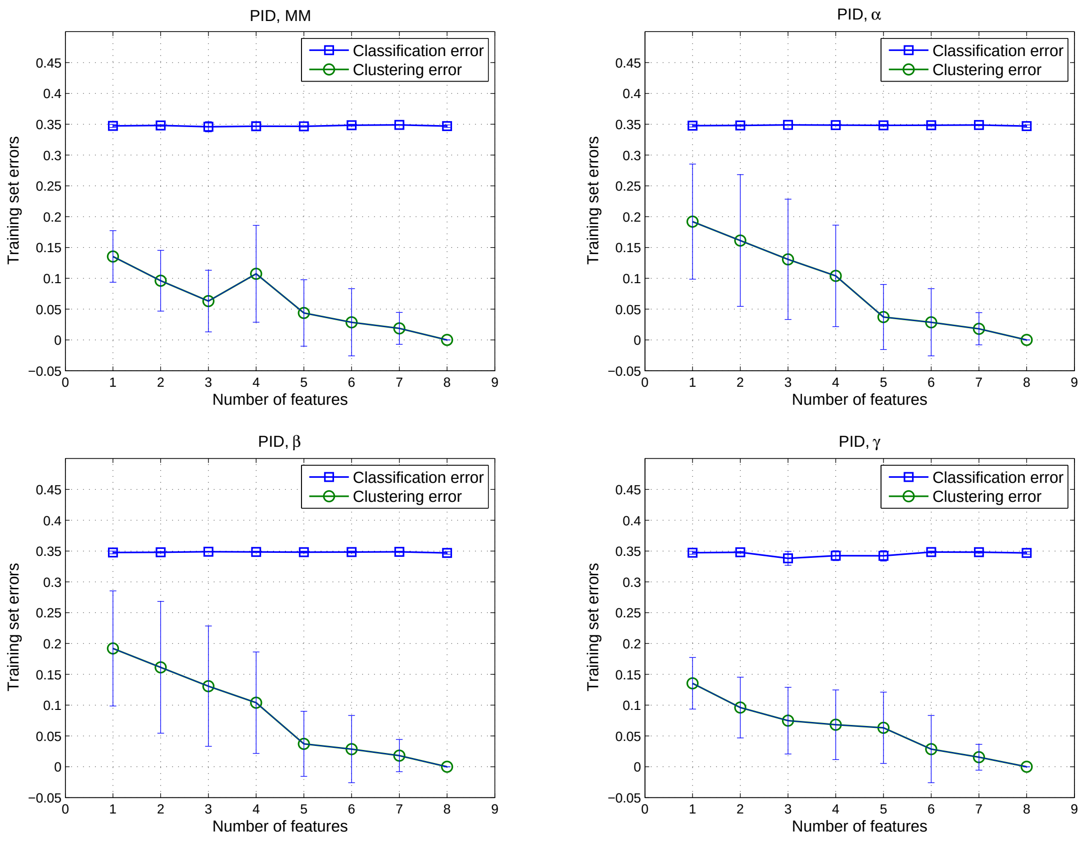


Figure A.8: Error curves generated by BSS feature selection methods on the PID data set

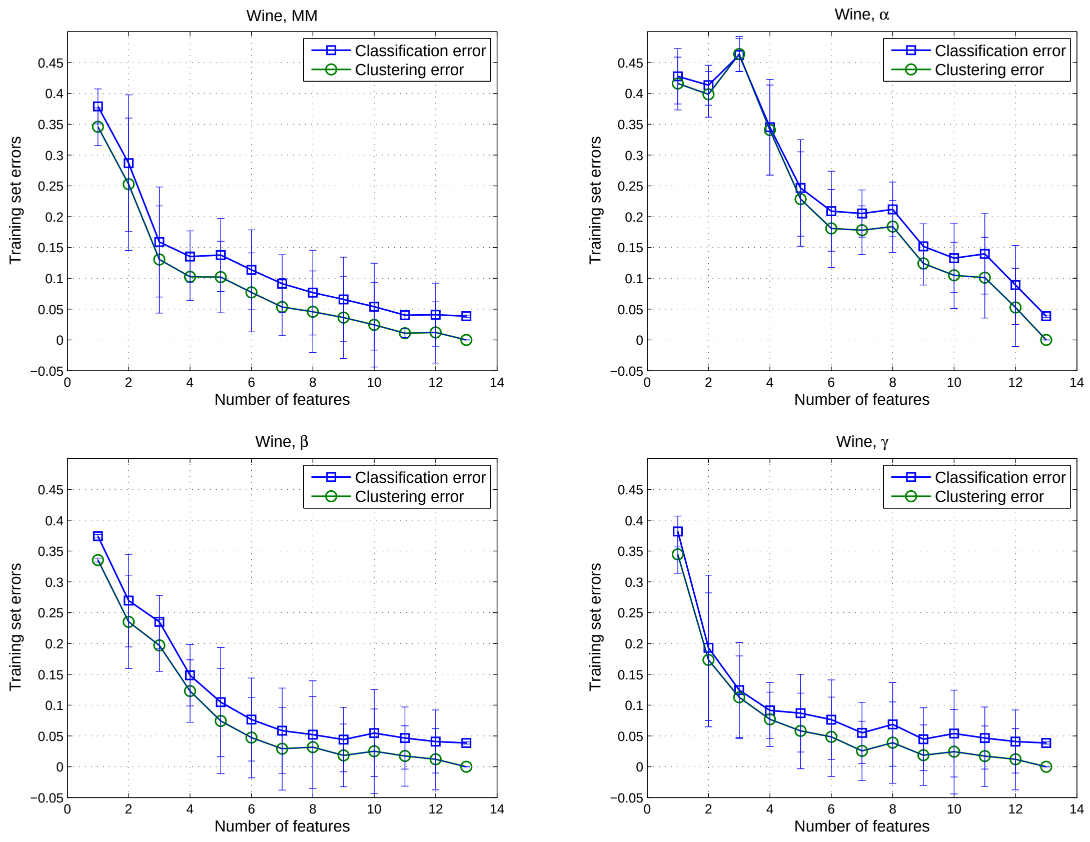


Figure A.9: Error curves generated by FSS feature selection methods on the Wine data set

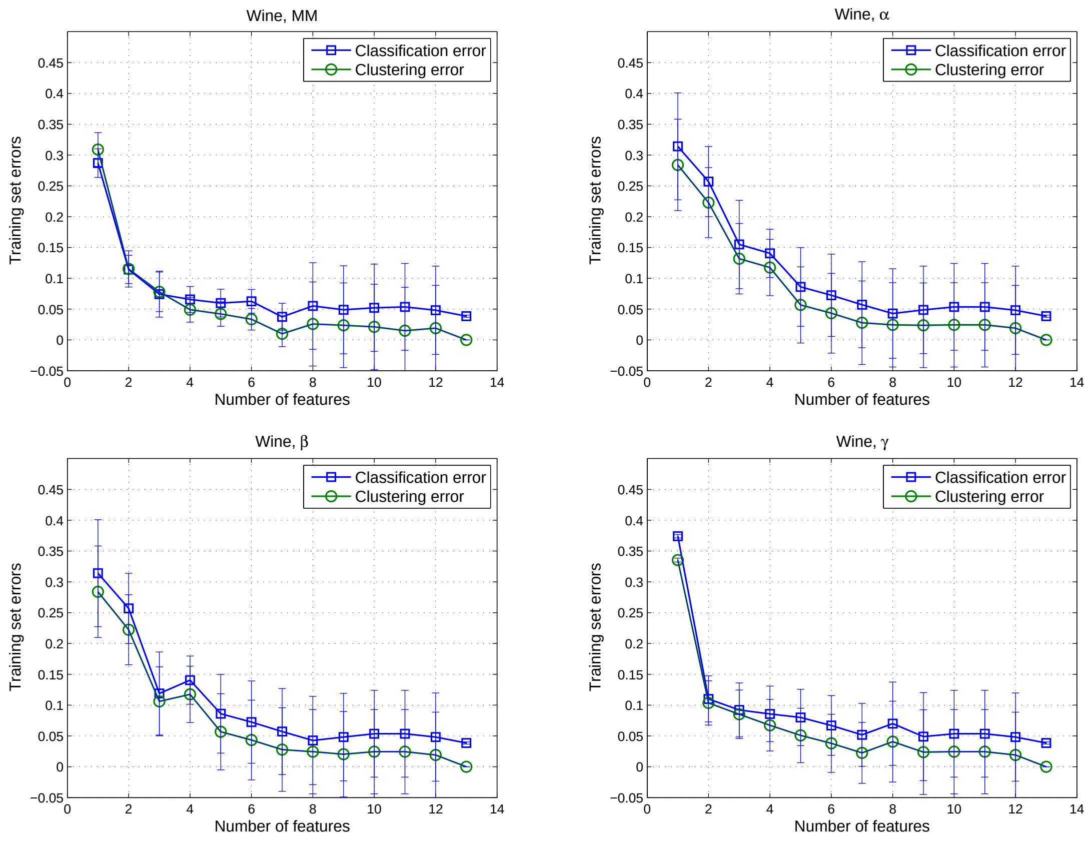
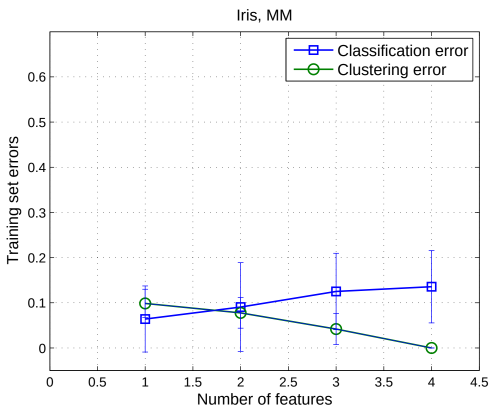
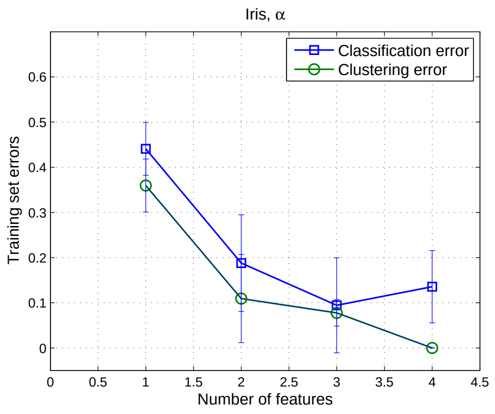
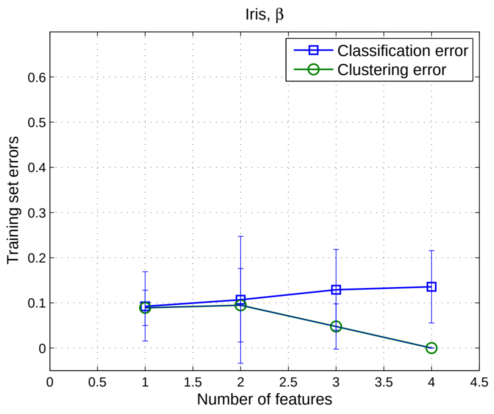


Figure A.10: Error curves generated by BSS feature selection methods on the Wine data set







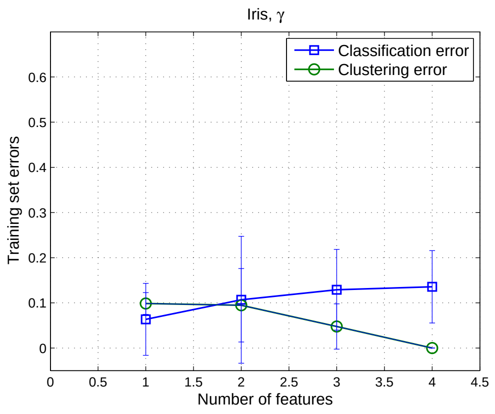


Figure A.11: Error curves generated by FSS feature selection methods on the Iris data set

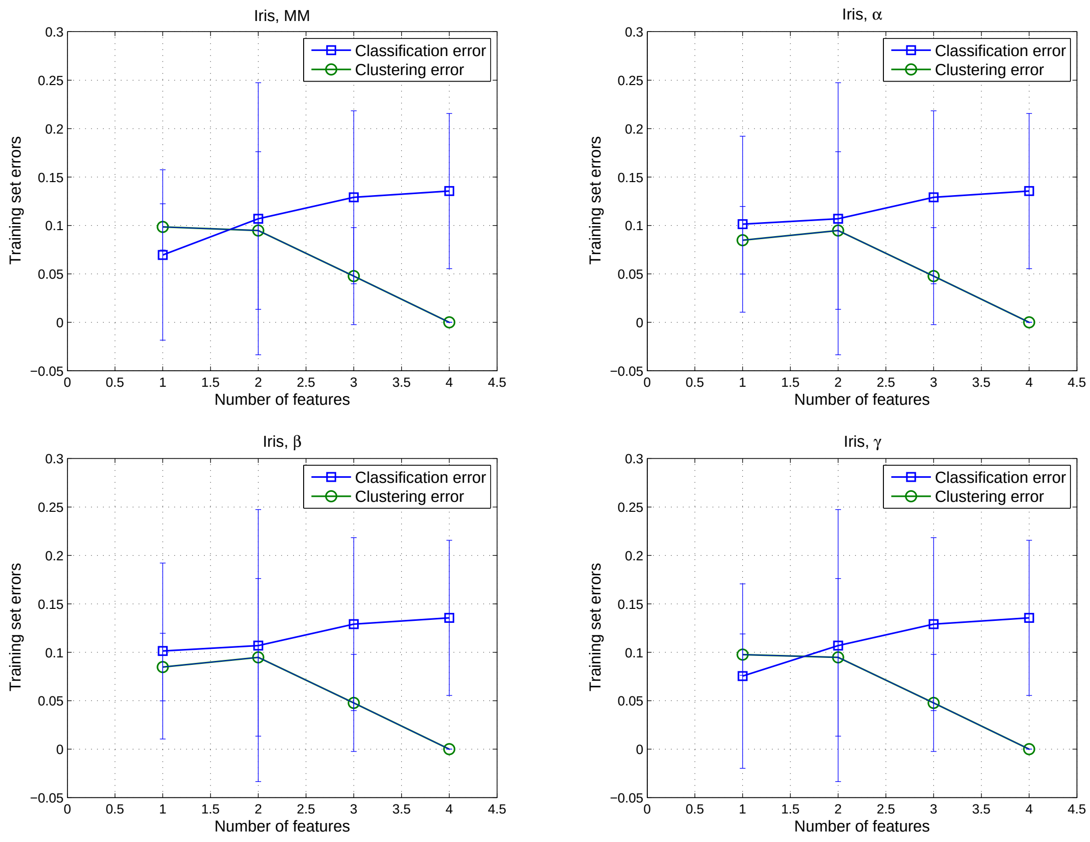


Figure A.12: Error curves generated by BSS feature selection methods on the Iris data set

Appendix B

FSSPKM Feature Selection Error Curves Part 2: MTS Data Sets

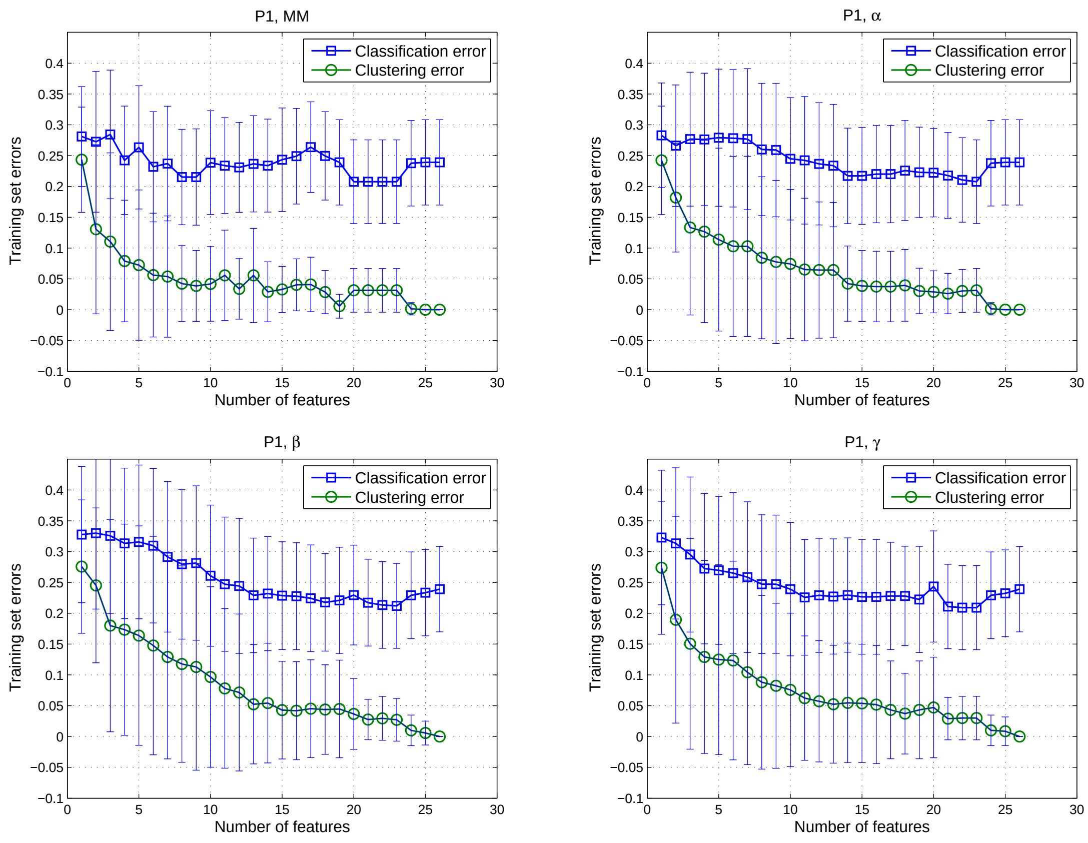


Figure B.1: Error curves generated by FSS feature selection methods on the P1 data set

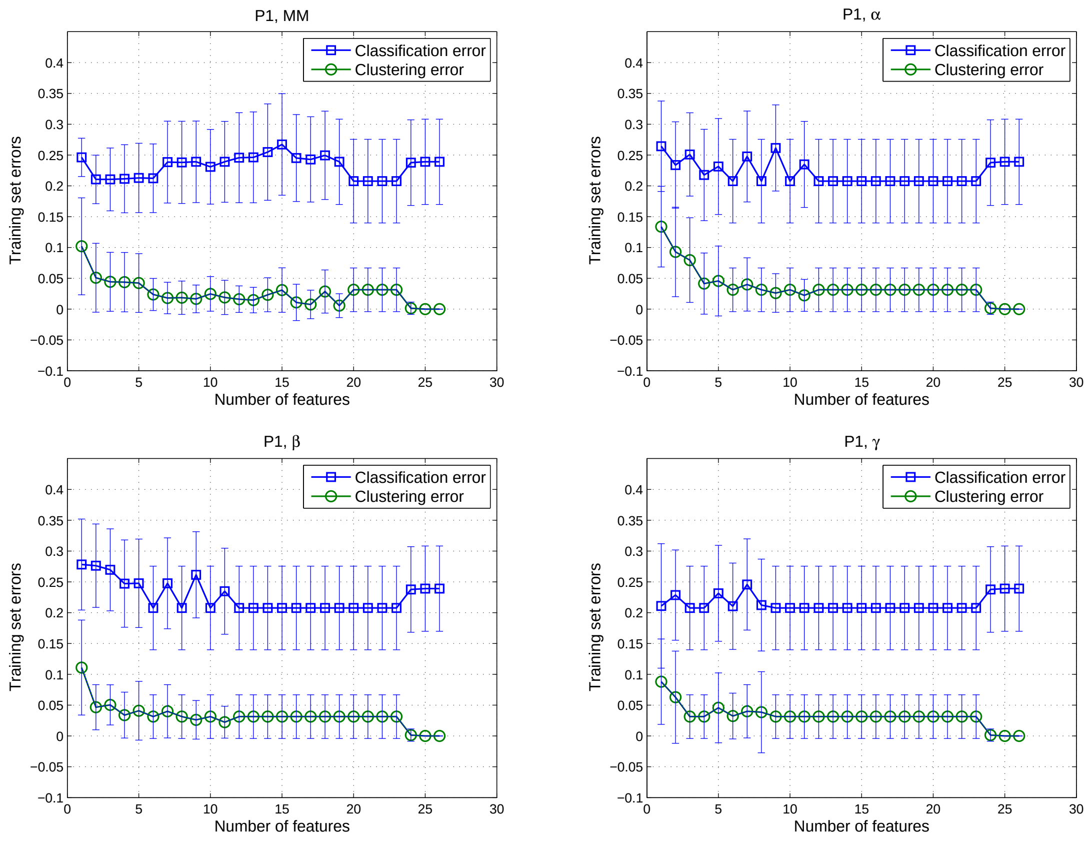
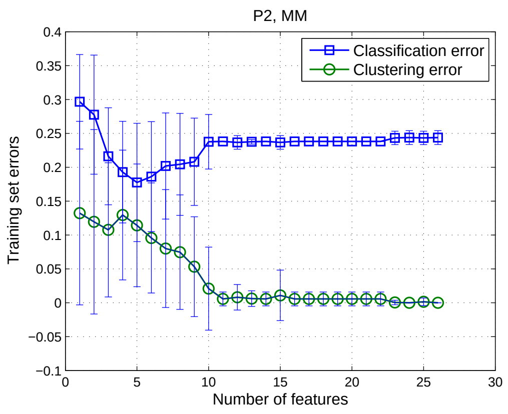
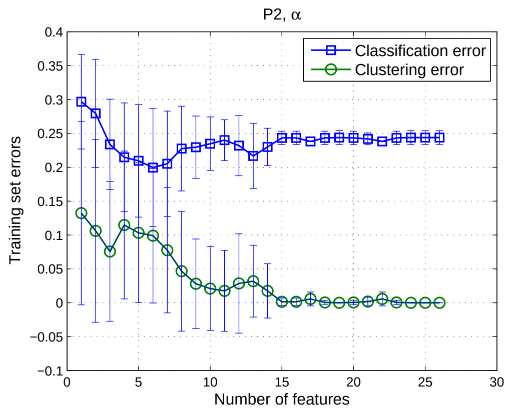
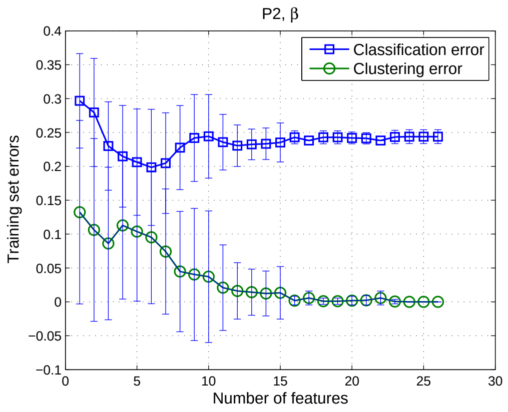


Figure B.2: Error curves generated by BSS feature selection methods on the P1 data set







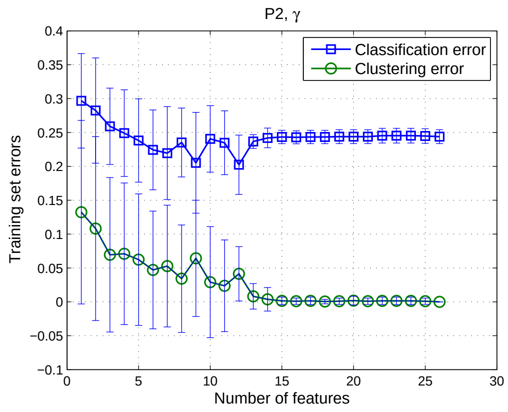


Figure B.3: Error curves generated by FSS feature selection methods on the P2 data set

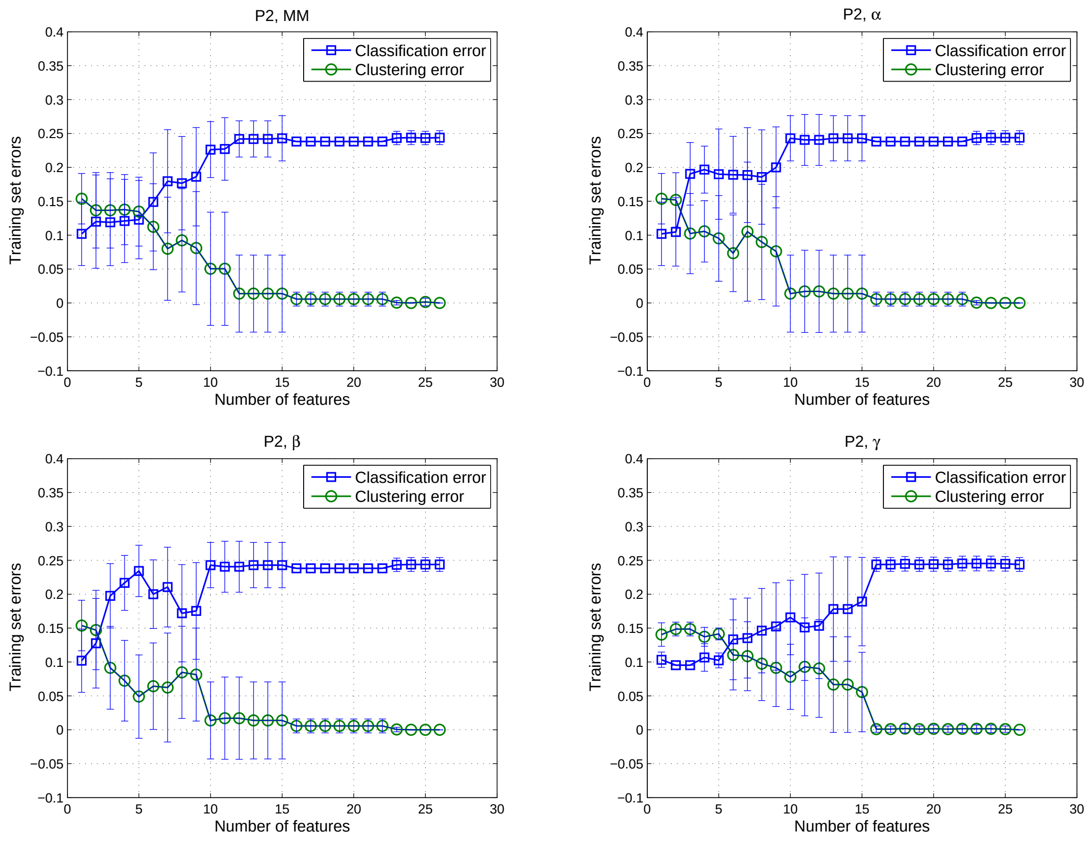


Figure B.4: Error curves generated by BSS feature selection methods on the P2 data set

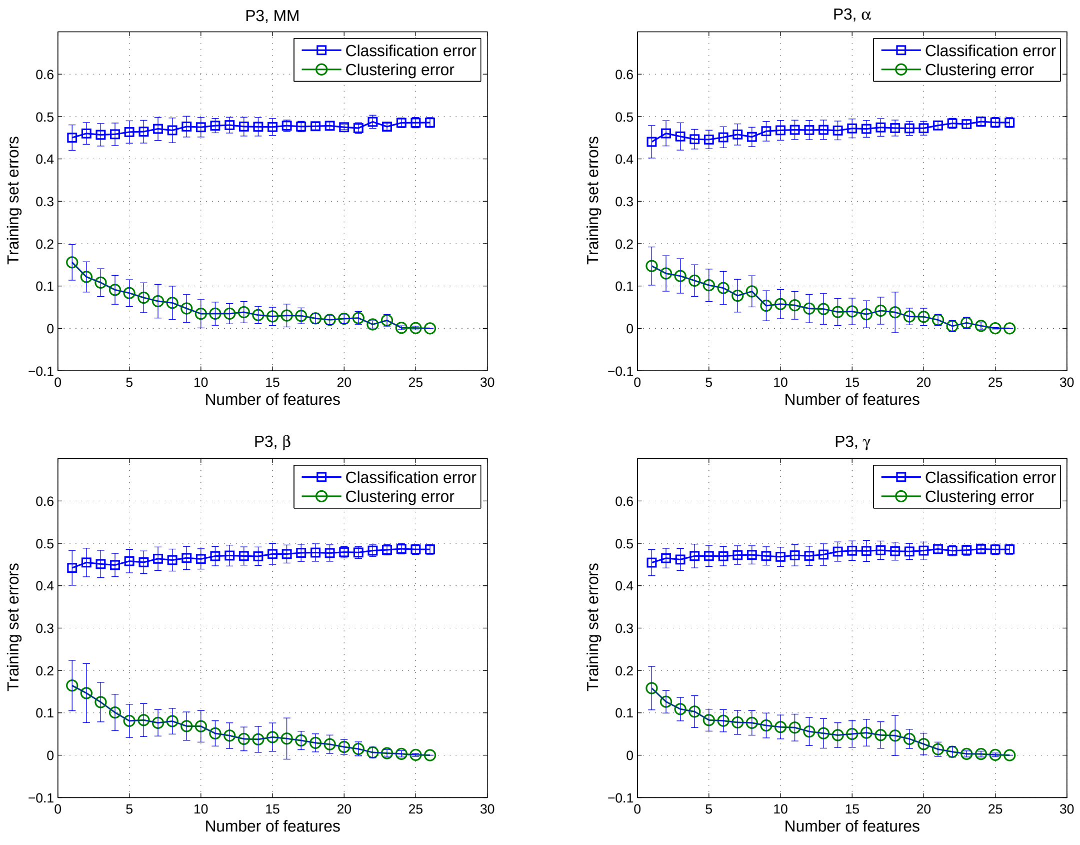
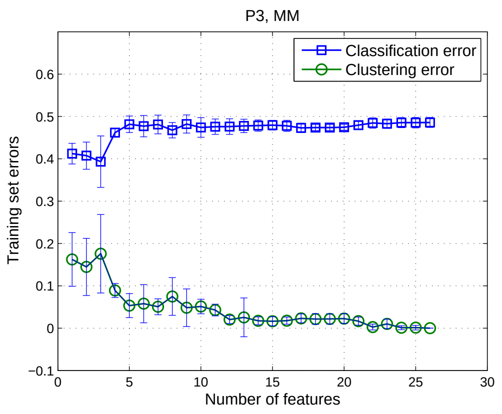
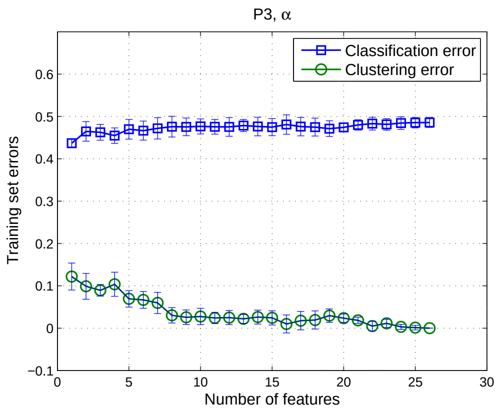
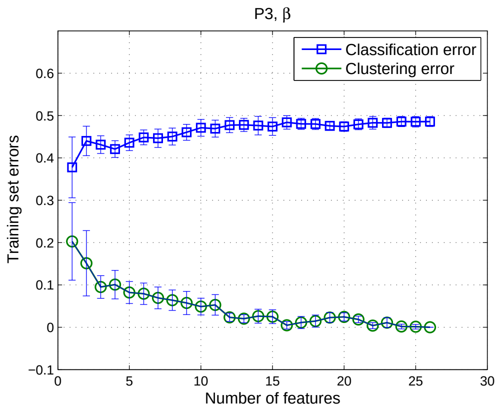


Figure B.5: Error curves generated by FSS feature selection methods on the P3 data set







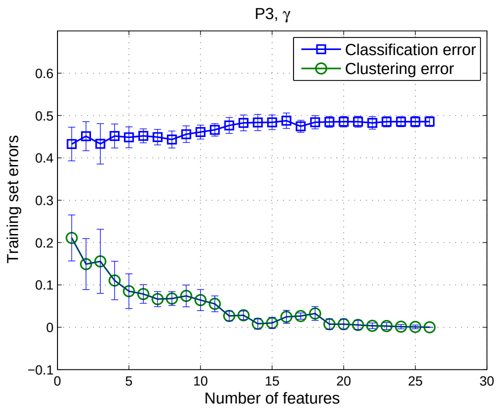


Figure B.6: Error curves generated by BSS feature selection methods on the P3 data set

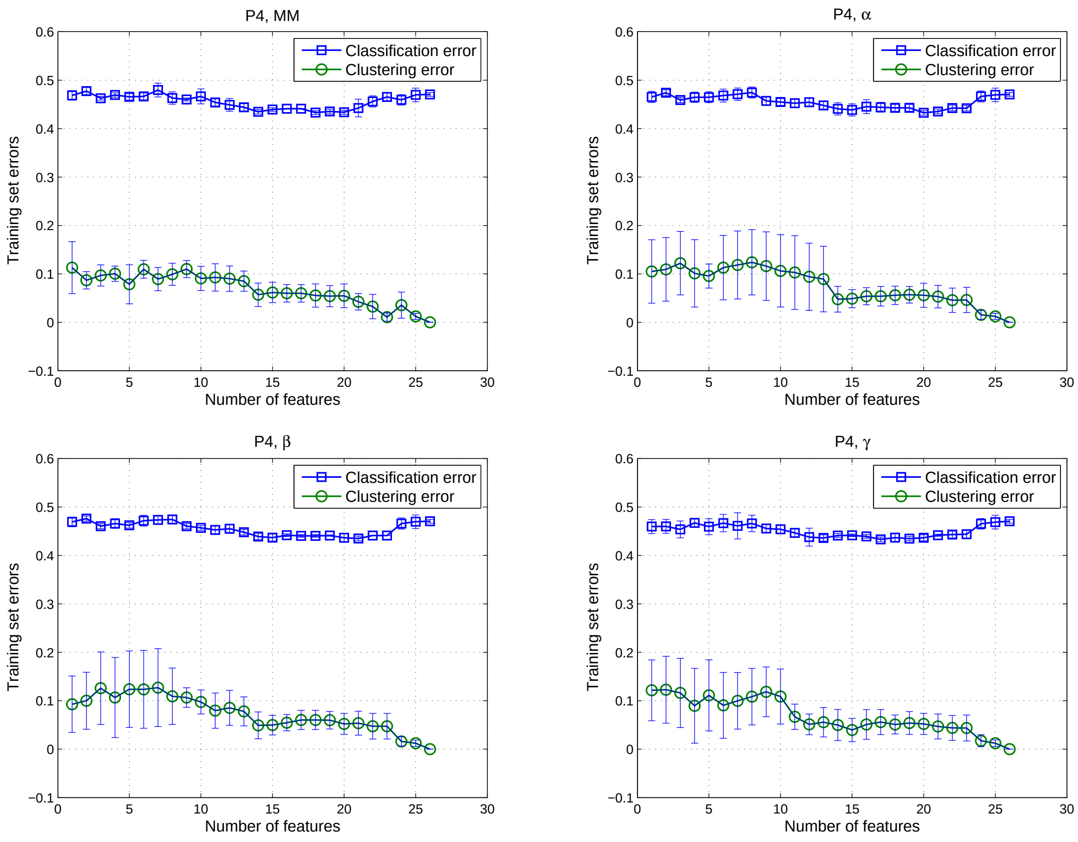


Figure B.7: Error curves generated by FSS feature selection methods on the P4 data set

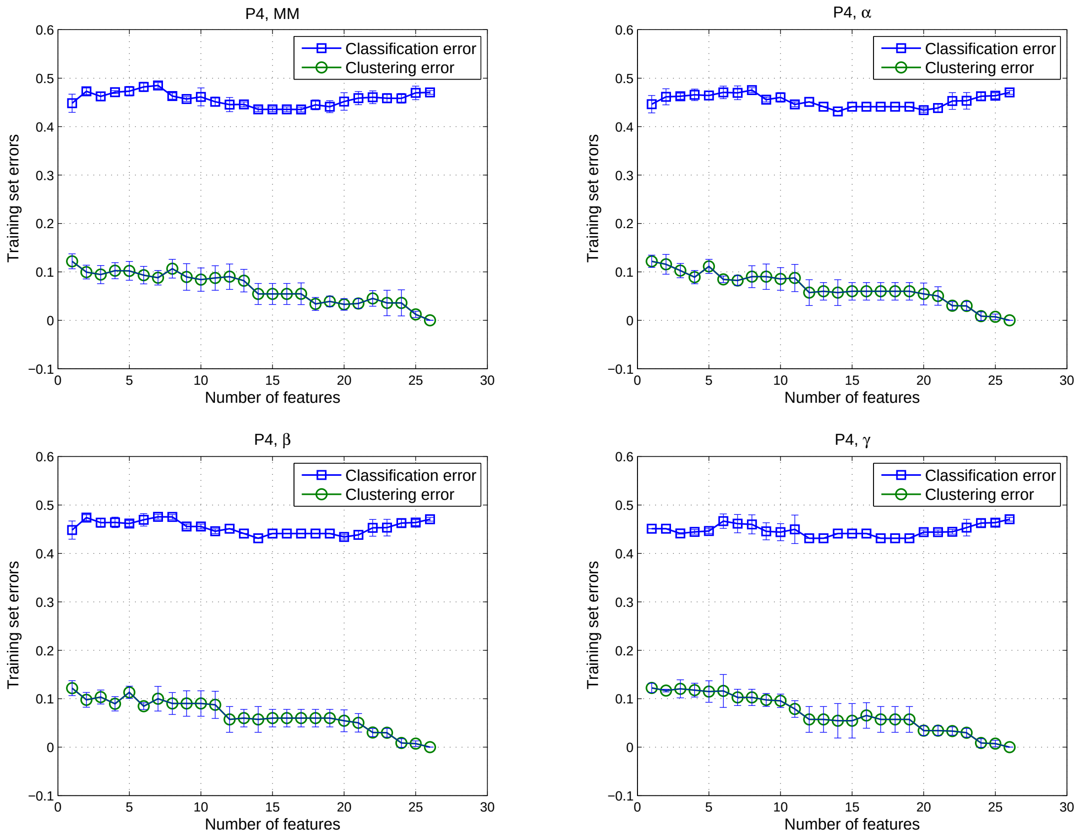


Figure B.8: Error curves generated by BSS feature selection methods on the P4 data set

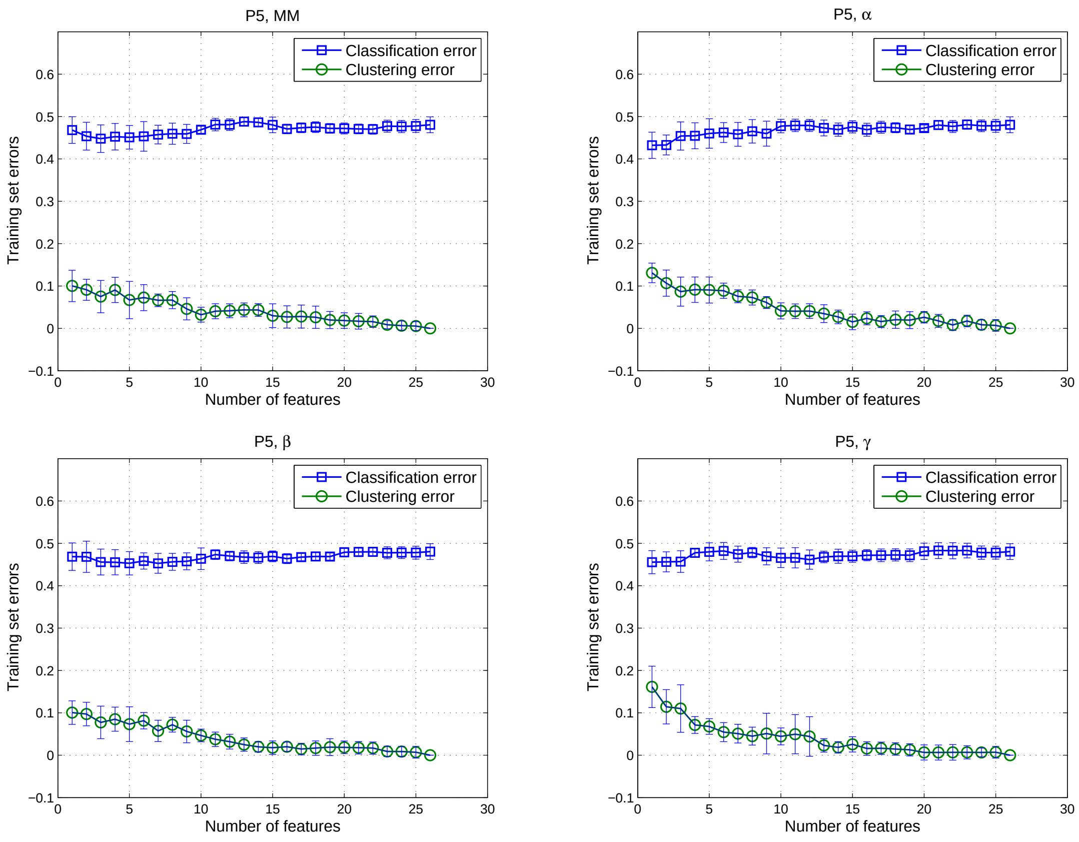


Figure B.9: Error curves generated by FSS feature selection methods on the P5 data set

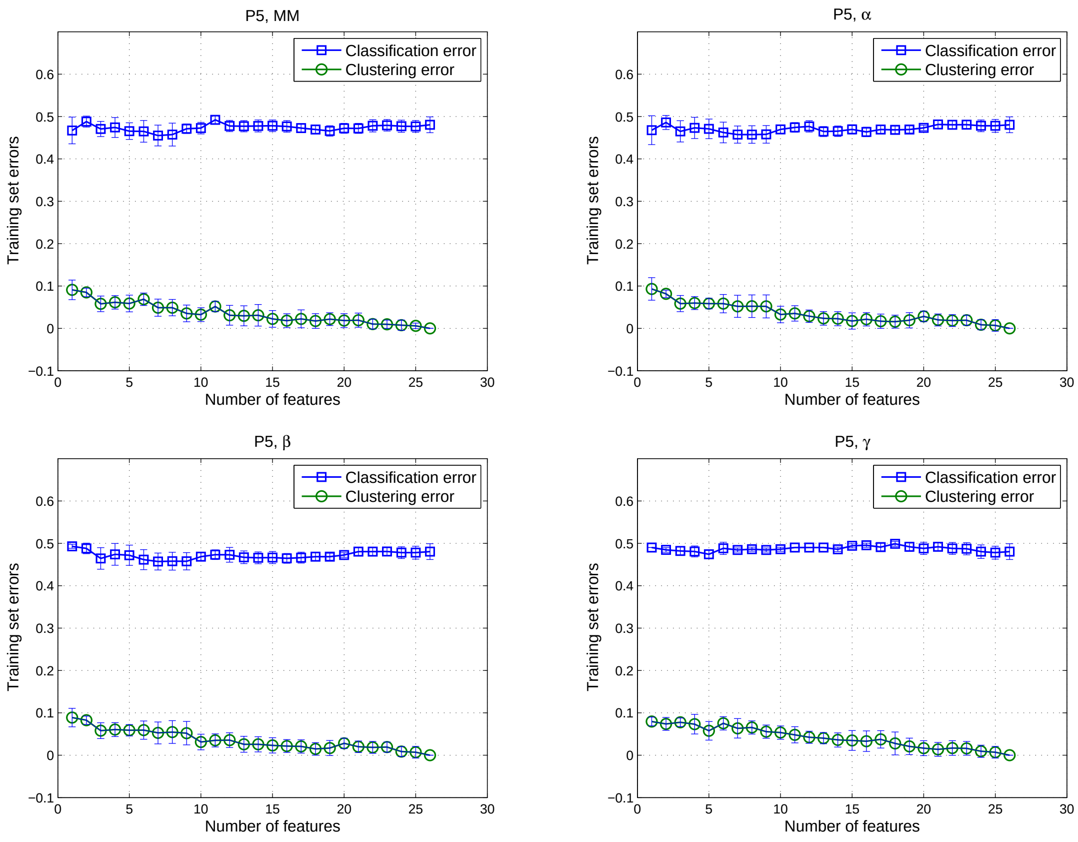


Figure B.10: Error curves generated by BSS feature selection methods on the P5 data set

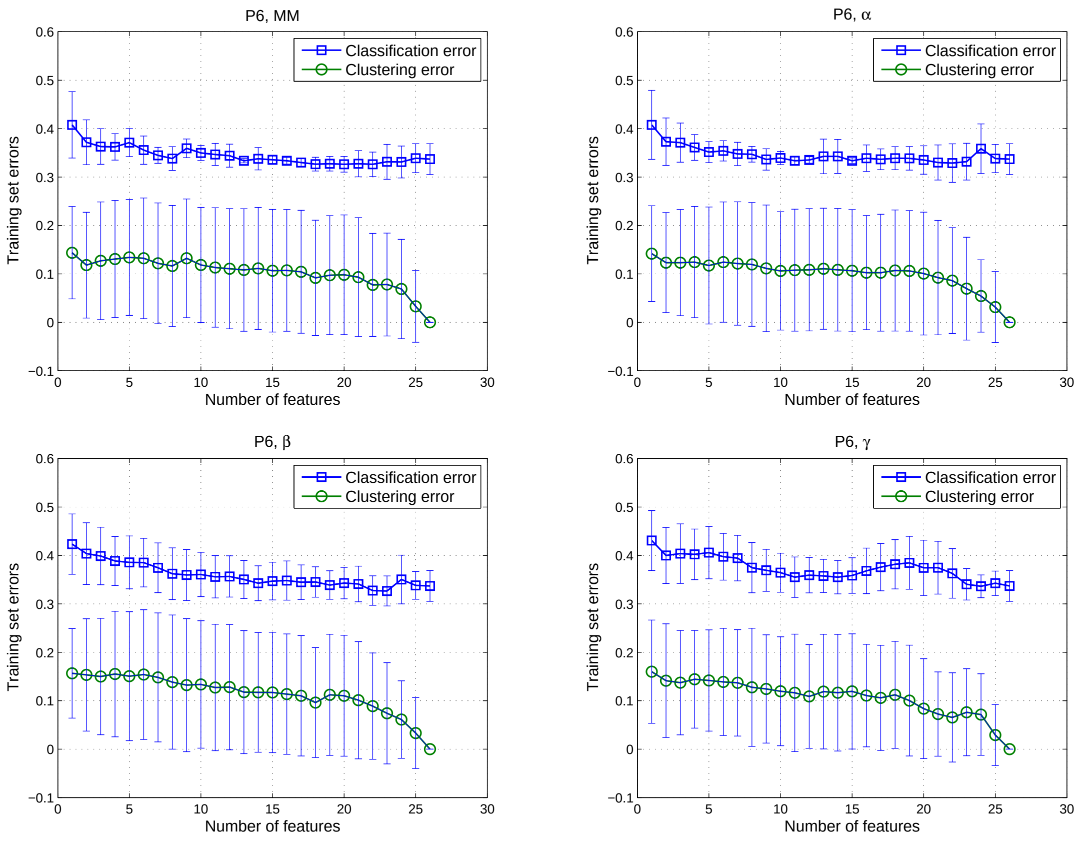


Figure B.11: Error curves generated by FSS feature selection methods on the P6 data set

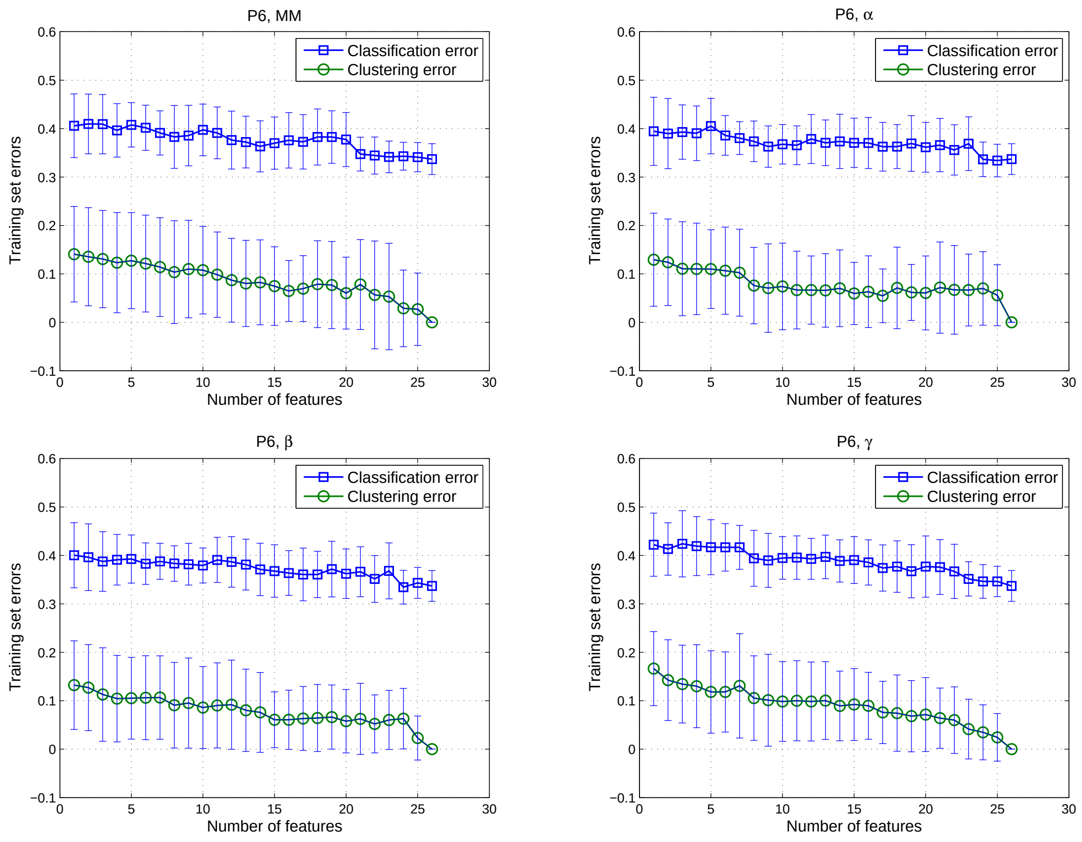


Figure B.12: Error curves generated by BSS feature selection methods on the P6 data set

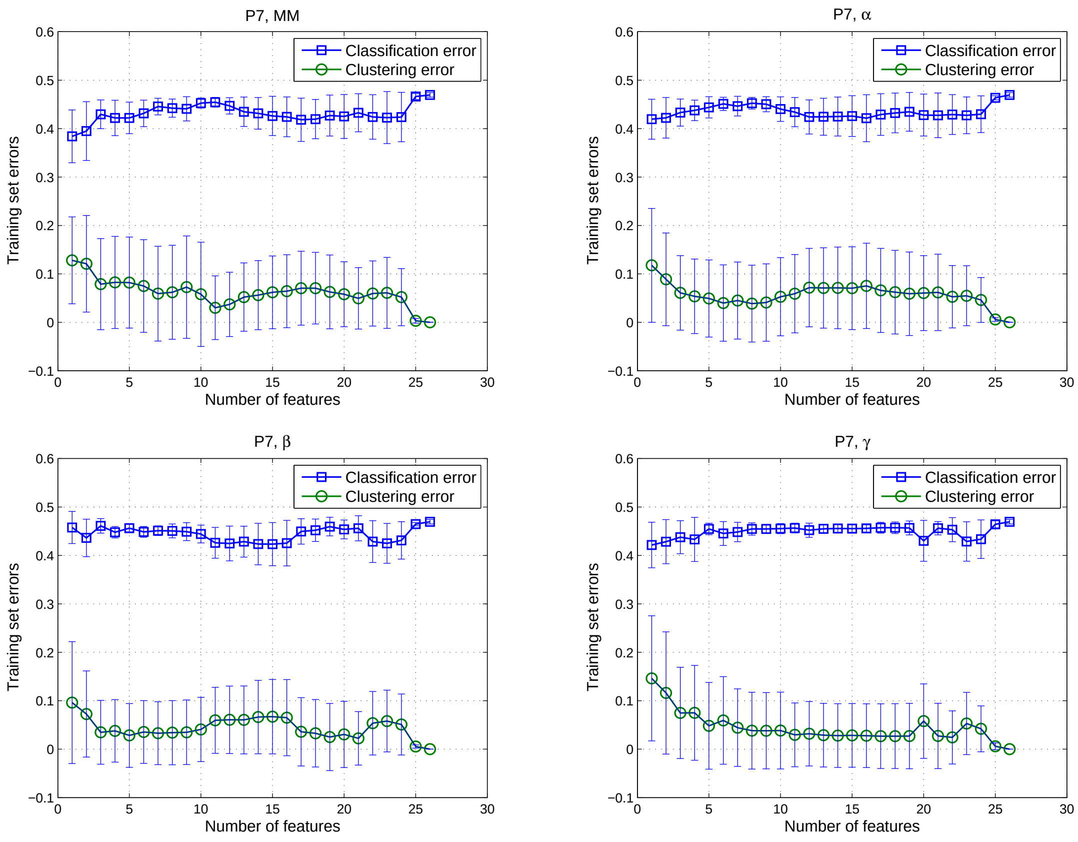


Figure B.13: Error curves generated by FSS feature selection methods on the P7 data set

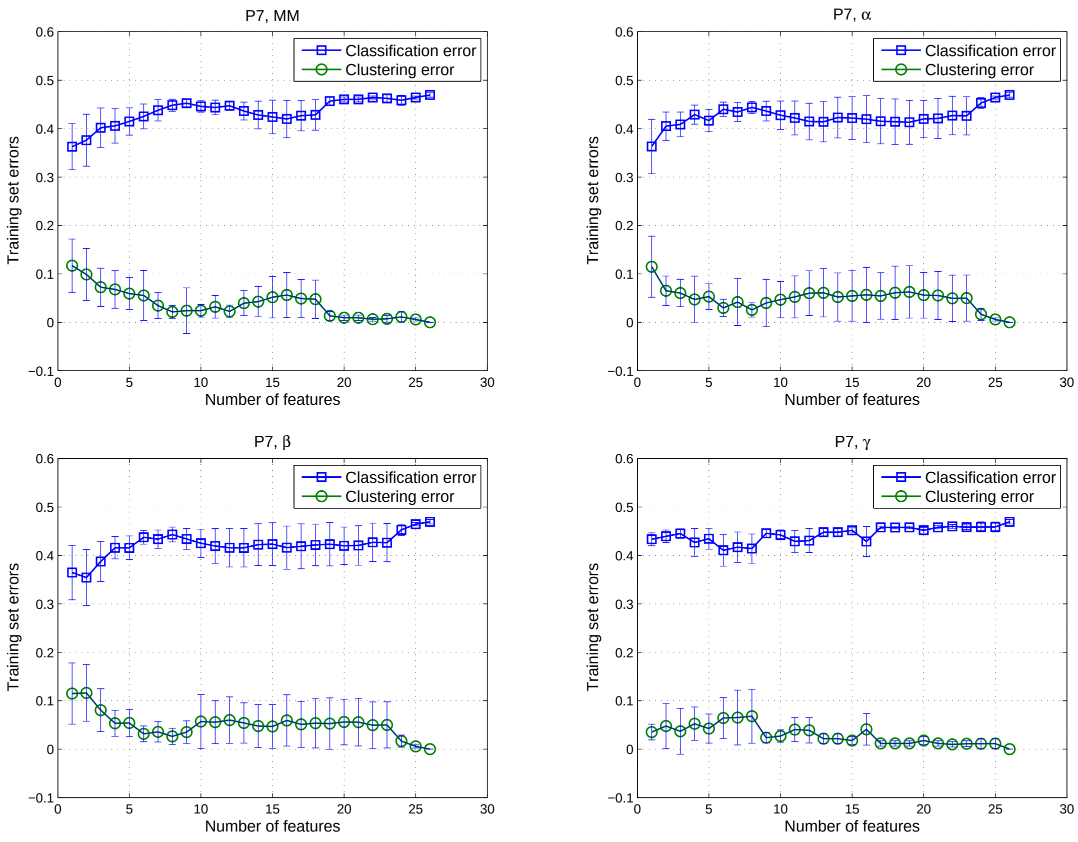


Figure B.14: Error curves generated by BSS feature selection methods on the P7 data set

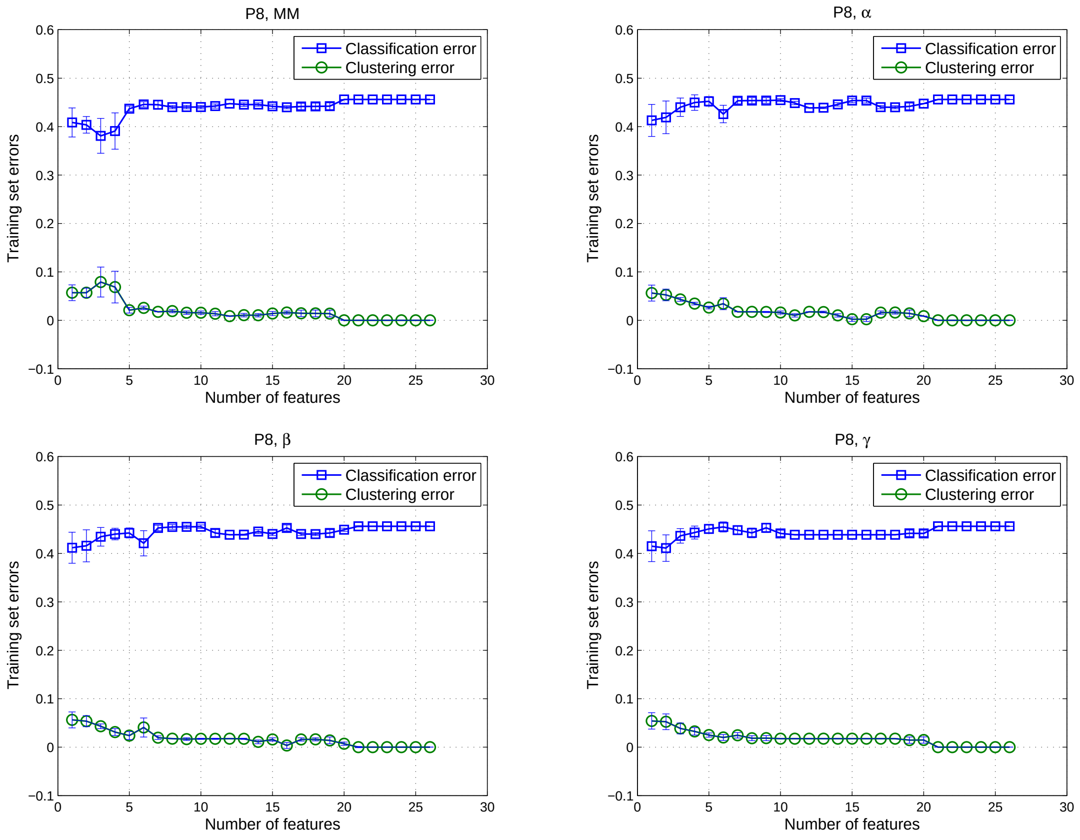


Figure B.15: Error curves generated by FSS feature selection methods on the P8 data set

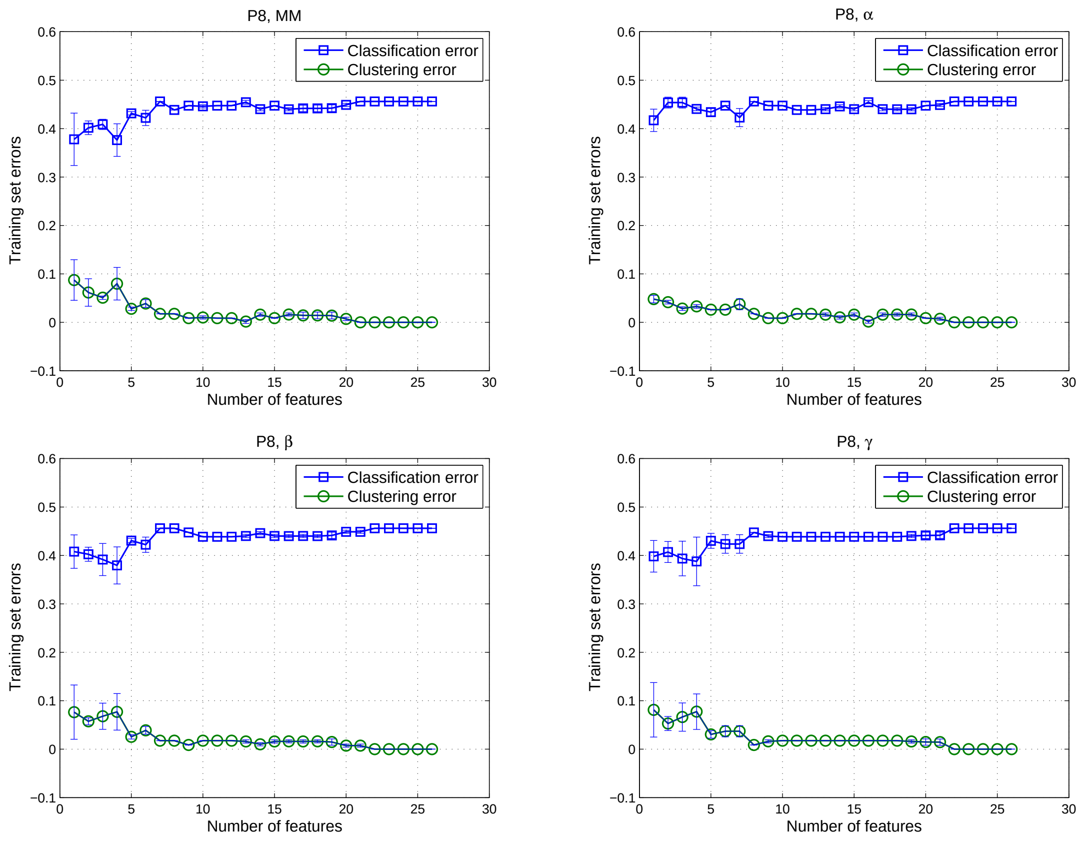


Figure B.16: Error curves generated by BSS feature selection methods on the P8 data set

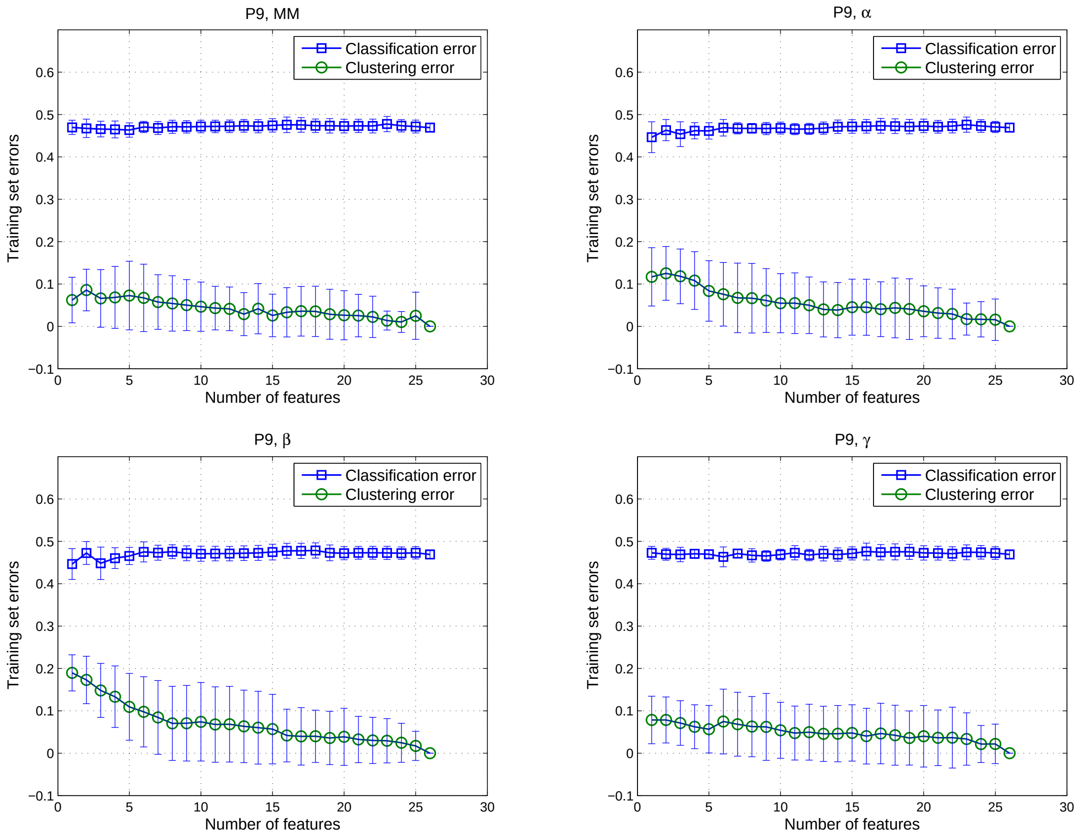


Figure B.17: Error curves generated by FSS feature selection methods on the P9 data set

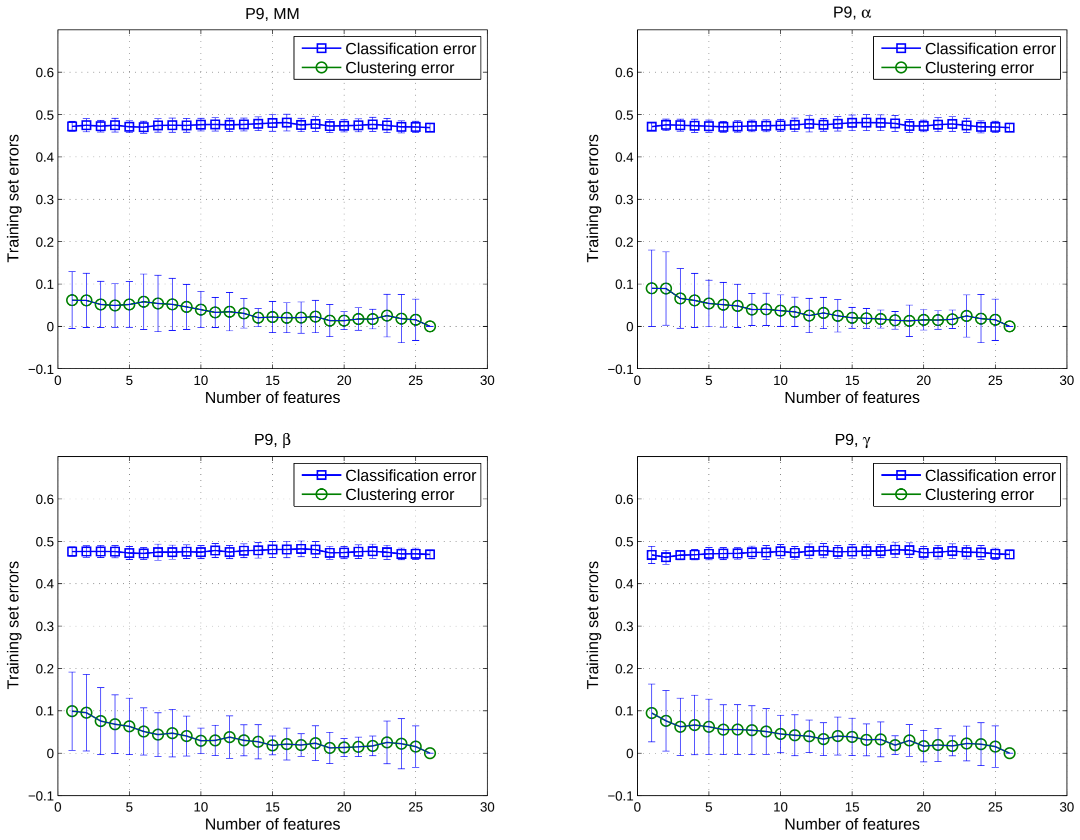


Figure B.18: Error curves generated by BSS feature selection methods on the P9 data set

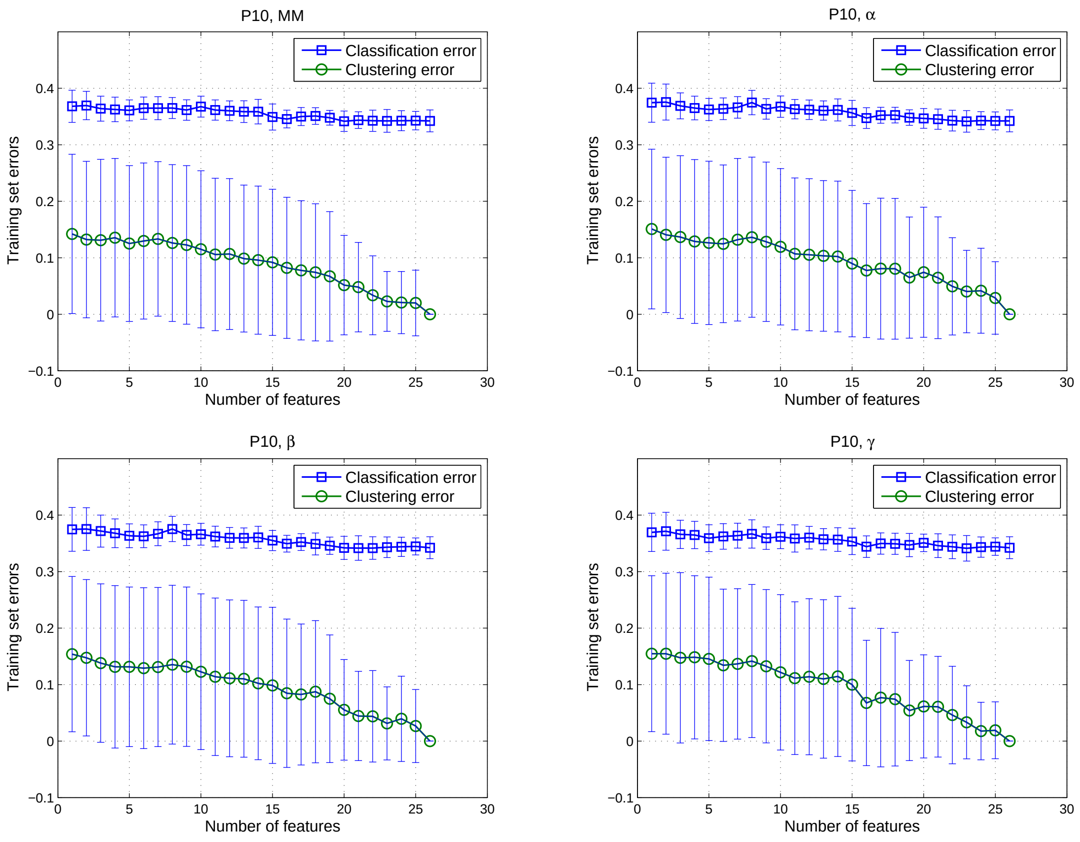


Figure B.19: Error curves generated by FSS feature selection methods on the P10 data set

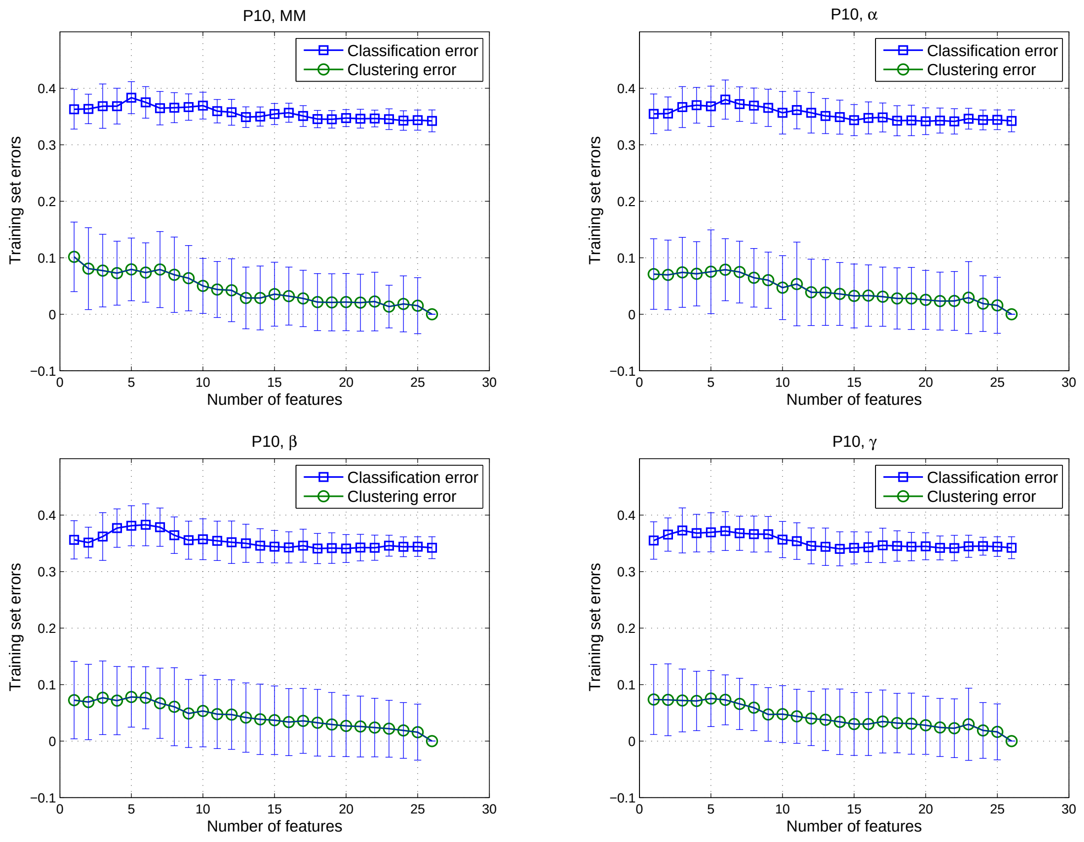


Figure B.20: Error curves generated by BSS feature selection methods on the P10 data set

References

[1] B. J. Anderson, D. S. Gross, D. R. Musicant, A. M. Ritz, T. G. Smith, and L. E. Steinberg. Adapting k-medians to generate normalized cluster centers. In *Proceedings of the Sixth SIAM International Conference on Data Mining, Society for Industrial and Applied Mathematics*, pages 165–175, Bethesda, MD, 2006.

[2] J. A. Anderson. Constrained discrimination between k populations. *Journal of the Royal Statistical Society Series B*, 31:123–139, 1969.

[3] A. Asuncion and D. J. Newman. UCI machine learning repository, 2007.

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, volume 14, pages 585–591, 2002.

[5] M. Belkin and P. Niyogi. Using manifold structure for partially labelled classification. In *Advances in Neural Information Processing Systems 14*, pages 929–936, 2003.

[6] K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[7] K. P. Bennett and O. L. Mangasarian. Bilinear separation of two sets in n-space. *Computational Optimization and Applications*, 2:207–227, 1993.

[8] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of KDD-94: AAAI Workshop on Knowledge Discovery in Databases*, pages 359–370, July 1994.

[9] J. R. Beveridge, K. She, B. A. Draper, and G. H. Givens. A nonparametric statistical comparison of principal component and linear discriminant subspaces for face recognition. IEEE Conference on Computer Vision and Pattern Recognition, Kauai, HI, December 2001.

[10] E. Boros, P. L. Hammer, T. Ibaraki, and A. Kogan. Logical analysis of numerical data. *Math. Program.*, 79(1-3):163–190, 1997.

[11] E. Boros, P. L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, and I. Muchnik. An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering*, 12(2):292–306, Mar/Apr 2000.

[12] G. E. P. Box and G. M. Jenkins. Statistical models for prediction and control. Technical Reports #72, 77, 79, 94, 95, 99, 103, 104, 116, 121, and 122, Department of Statistics, University of Wisconsin, Madison, Wisconsin, 1967.

[13] P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained k-means clustering. Technical report, MSR-TR-2000-65, Microsoft Research, 2000.

[14] P. S. Bradley and O. L. Mangasarian. k-plane clustering. *Journal of Global Optimization*, 16(1):23–32, January 2000.

[15] P. S. Bradley, O. L. Mangasarian, and W. N. Street. Clustering via concave minimization. *Advances in Neural Information Processing Systems*, 9:368–374, 1997.

[16] P. S. Bradley, O. L. Mangasarian, and W. N. Street. Feature selection via mathematical programming. *INFORMS Journal on Computing*, 10:209–217, 1998.

[17] Paul S. Bradley and Usama M. Fayyad. Refining initial points for k-means clustering. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 91–99, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[18] C. Burges. Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[19] W. A. Chaovalitwongse, Y. J. Fan, and R. C. Sachdeo. On the time series k-nearest neighbor for abnormal brain activity classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 37(6):1005–1016.

[20] W. A. Chaovalitwongse, Y. J. Fan, and R. C. Sachdeo. Novel Optimization Models for Abnormal Brain Activity Classification. *Operations Research*, 56(6):1450–1460, December 2008.

[21] W. A. Chaovalitwongse, P. M. Pardalos, L. D. Iasemidis, D. S. Shiau, and J. C. Sackellares. Applications of global optimization and dynamical systems to prediction of epileptic seizures. In P.M. Pardalos, J.C. Sackellares, L.D. Iasemidis, and P.R. Carney, editors, *Quantitative Neuroscience*, pages 1–36. Kluwer Academic Publishers, 2003.

[22] W. A. Chaovalitwongse, P. M. Pardalos, and O. A. Prokoyev. Electroencephalogram (EEG) Time Series Classification: Applications in Epilepsy. *Annals of Operations Research*, 148:227–250, 2006.

[23] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *In Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 378–388, 1999.

[24] M. Charikara, S. Guhab, . Tardosc, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, August 2002.

[25] D. Chhajed and T. J. Lowe. m-median and m-center problems with mutual communication: Solvable special cases. *Operations Research*, 40:S56–S66, 1992.

[26] A. Cord, C. Ambroise, and J.-P. Cocquerez. Feature selection in robust clustering based on laplace mixture. *Pattern Recognition Letters*, 27(6):627 – 635, 2006.

[27] Y. Crama, P. L. Hammer, and T. Ibaraki. Cause-effect relationships and partially defined boolean functions. *Annals of Operations Research*, 16(1):299–325, December 1988.

[28] A. Davison and D. Hinkley. *Bootstrap Methods and Their Application.* Cambridge University Press, 1997.

[29] M. Davy, A. Gretton, A. Doucet, and P. J. W. Rayner. Optimized support vector machines for nonstationary signal classification. *Signal Processing Letters, IEEE*, 9(12):442–445, Dec 2002.

[30] P. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach.* Prentice Hall, 1982.

[31] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proc. VLDB Endow.*, 1(2):1542–1552, 2008.

[32] D. Dong and T. J. McAvoy. Nonlinear principal component analysis–based on principal curves and neural networks. *Computers and Chemical Engineering*, 20(1):65 – 78, 1996.

[33] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification, Second Edition.*

[34] Jennifer G. Dy and Carla E. Brodley. Feature subset selection and order identification for unsupervised learning. In *In Proc. 17th International Conf. on Machine Learning*, pages 247–254. Morgan Kaufmann, 2000.

[35] V. Faber. Clustering and the continuous k-means algorithm. *Los Alamos Science*, 22:138–144, 1994.

[36] Y. J. Fan and W. A. Chaovalitwongse. Optimizing feature selection to improve medical diagnosis. *Annals of Operations Research.*

[37] M. C. Ferris. Matlab and gams: Interfacing optimization and visualization software. Mathematical Programming Technical Report 98-19, Department of Computer Science, University of Wisconsin-Madison, Madison, WI, November 1998.

[38] D. Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2007.

[39] G. Fung and O. L. Mangasarian. A feature selection newton method for support vector machine classification. *Computational Optimization and Applications*, 28:185–202, 2002.

[40] G. Fung and O. L. Mangasarian. Finite Newton method for Lagrangian support vector machine classification. *Neurocomputing*, (1-2):39–55, September 2003.

[41] R. J. Gallagher, E. K. Lee, and D. A. Patterson. Constrained discriminant analysis via 0/1 mixed integer programming. *Annals of Operations Research*, 74:65–88, 1997.

[42] E. Gholami and M. M. Borujerdi. Fuzzy knowledge discovery from time series data for events prediction. *PRICAI 2008: Trends in Artificial Intelligence*, 5351:646–657, 2008.

[43] P. L. Hammer and T. O. Bonates. Logical analysis of data - an overview: From combinatorial optimization to medical applications. *Annals of Operations Research*, 148(1):203–225, November 2006.

[44] A. Hayashi, Y. Mizuhara, and N. Suematsu. Embedding time series data for classification. *Machine Learning and Data Mining in Pattern Recognition*, 3587:356–365, August 2005.

[45] L. J. Heyer, S. Kruglyak, and S. Yooseph. Exploring expression data: Identification and analysis of coexpressed genes. *Genome Research*, 9:1106–1115, 1999.

[46] F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23(1):67–72, 1975.

[47] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[48] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[49] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001.

[50] M. W. Kadous and C. Sammut. Constructive induction for classifying time series. In *Machine Learning: ECML 2004, Lecture Notes in Computer Science*, volume 3201, pages 192–204, Berlin/Heidelberg, 2004. Springer.

[51] Y. Kakizawa, R. H. Shumway, and M. Taniguchi. Discrimination and clustering for multivariate time series. *Journal of the American Statistical Association*, 93(441):328340, month = , year = 1998.

[52] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, July 2002.

[53] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, March 2005.

[54] E. Keogh, L. Wei, X. Xi, S. H. Lee, and M. Vlachos. Lb_keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 882–893. VLDB Endowment, 2006.

[55] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. The ucr time series classification/clustering, 2006.

[56] S. S. Khan and A. Ahmad. Cluster center initialization algorithm for k-means clustering. *Pattern Recogn. Lett.*, 25(11):1293–1302, 2004.

[57] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, February 1991.

[58] W. J. Krzanowski. Between-groups comparison of principal components. *Journal of the American Statistical Association*, 74(367):703–707, 1979.

[59] M. H. C. Law, M. A. T. Figueiredo, and A. K. Jain. Simultaneous feature selection and clustering using mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:1154–1166, 2004.

[60] E. K. Lee, R. J. Gallagher, and D. A. Patterson. A Linear Programming Approach to Discriminant Analysis with a Reserved-Judgment Region. *INFORMS Journal on Computing*, 15(1):23–41, 2003.

[61] H. Lian. Bayesian nonlinear principal component analysis using random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):749–754, April 2009.

[62] A. Likas, N. Vlassis, and J. J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36:451–461, 2001.

[63] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[64] S. Y. Lu and K. S. Fu. A sentence-to-sentence clustering procedure for pattern analysis. *IEEE Transactions on Systems, Man and Cybernetics*, 8(5):381–389, 1978.

[65] C. M. Bishop M. E. Tipping. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, February 1999.

[66] C. M. Bishop M. E. Tipping. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B: Statistical Methodology*, 61(3):611–622, 1999.

[67] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *L. Le Cam and J. Neyman, Editors, Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume I*, pages 281–297, 1967.

[68] M. Mahdian, Y. Ye, and J. Zhang. Approximation algorithms for metric facility location problems. *SIAM J. Comput.*, 36(2):411–432, 2006.

[69] O. L. Mangasarian. Data mining via support vector machines. In *IFIP Conference on System Modelling and Optimization*, pages 23–27. Kluwer Academic Publishers, 2001.

[70] O. L. Mangasarian, W. N. Street, and W. H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43:570–577, 1995.

[71] O. L. Mangasarian and E. W. Wild. Feature selection in k-median clustering. In *SIAM International Conference on Data Mining, Workshop on Clustering High Dimensional Data and its Applications*, pages 23–28, 2004.

[72] R. R. Mettu and C. G. Plaxton. The online median problem. In *In Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 339–348, 1999.

[73] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. R. Mullers. Fisher discriminant analysis with kernels. pages 41–48, Aug 1999.

[74] A. M. Molinaro, R. Simon, and R. M. Pfeiffer. Prediction error estimation: A comparison of resampling methods. *Bioinformatics*, 21:3301–3307, 2005.

[75] K. Morik and H. Kopcke. Analysing customer churn in insurance data v a case study. *Knowledge Discovery in Databases: PKDD 2004*, 3202:325–336, 2004.

[76] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover. Exact discovery of time series motifs. In *SIAM International Conference on Data Mining (SDM09)*. American Statistical Association (ASA), 2009.

[77] P. M. Pardalos, V. L. Boginski, and A. Vazacopoulos. *Data Mining in Biomedicine*. Springer, New York, 2007.

[78] E. Pekalska, R. P. W. Duin, and P. Pacl. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, 39(2):189 – 208, 2006. Part Special Issue: Complexity Reduction.

[79] R. Rosipal and M. Girolami. An expectation-maximization approach to nonlinear component analysis. *Neural Computation*, 13(3):505–510, March 2001.

[80] V. Roth and T. Lange. Feature selection in clustering problems. In *In Advances in Neural Information Processing Systems 16*. MIT Press, 2003.

[81] K. Saastamoinen and J. Ketola. Medical data classification using logical similarity based measures. In *Proceedings of 2006 IEEE Conference on Cybernetics and Intelligent Systems*, pages 1–5, 2006.

[82] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, Feb 1978.

[83] B. Scholkopf and A. J. Smola. *Learning with Kernels Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2002.

[84] M. Scholkopf, A. Smola, and K. R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, July 1998.

[85] H. Shimodaira, K. I. Noma, M. Nakai, and S. Sagayama. Dynamic Time-Alignment Kernel in Support Vector Machine. *Advances in Neural Information Processing Systems 14*, 2:921–928, 2001.

[86] A. Singhal and D. E. Seborg. Clustering of multivariate time-series data. volume 5, pages 3931–3936, 2002.

[87] T. Stamkopoulos, K. Diamantaras, N. Maglaveras, and M. Strintzis. Ecg analysis using nonlinear pca neural networks for ischemia detection. *IEEE Transactions on Signal Processing*, 46(11):3058–3067, Nov 1998.

[88] H. Suo, K. Nie, X. Sun, and Y. Wang. One optimized choosing method of k-means document clustering center. *Information Retrieval Technology*, 4993:490–495, May 2008.

[89] S. Tan. Neighbor-weighted K-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, 28(4):667 – 671, 2005.

[90] G. L. Tsirogiannis, D. Frossyniotis, J. Stoitsis, S. Golemati, A. Stafylopatis, and K. S. Nikita. Classification of medical data with a robust multi-level combination scheme. In *Proceedings of 2004 IEEE International Joint Conference on Neural Networks*, volume 3, pages 25–29, 2004.

[91] K. Ueno, X. Xi, E. Keogh, and D.-J. Lee. Anytime classification using the nearest neighbor algorithm with applications to stream mining. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 623–632, Washington, DC, USA, 2006. IEEE Computer Society.

[92] V. N. Vapnik. *Statistical Learning Theory.* John Wiley and Sons, Inc., New York, 1998.

[93] G. Wahba, Y. Wang, C. Gu, R. Klein, and B. Klein. Smoothing spline anova for exponential families, with application to the wisconsin epidemiological study of diabetic retinopathy. *The Annals of Statistics*, 23:1865–1895, 1995.

[94] D. R. Wilson and T. R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 3:257–286, March 2000.

[95] L. A. Wolsey. *Integer Programming.* Wiley-Interscience Series in Discrete Mathematics and Optimization, New York, 1998.

[96] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana. Fast time series classification using numerosity reduction. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 1033–1040, New York, NY, USA, 2006. ACM.

[97] K. Yang and C. Shahabi. A pca-based similarity measure for multivariate time series. In *MMDB '04: Proceedings of the 2nd ACM international workshop on Multimedia databases*, pages 65–74, New York, NY, USA, 2004. ACM.

[98] K. Yang and C. Shahabi. A pca-based similarity measure for multivariate time series. In *MMDB '04: Proceedings of the 2nd ACM international workshop on Multimedia databases*, pages 65–74, New York, NY, USA, 2004. ACM.

[99] K. Yang, H. Yoon, and C. Shahabi. A supervised feature subset selection technique for multivariate time series. In *In Proceedings of the Workshop on Feature Selection for Data Mining: Interfacing Machine Learning with Statistics, 92-101*, 2005.

[100] H. Zhang, H. H. Zhang, G. Wahba, Y. Lin, M. Voelker, M. Ferris, R. Klein, and B. Klein. Variable selection and model building via likelihood basis pursuit. *Journal of the American Statistical Association*, 99(467):659–672, 2004.

[101] J. Zhang and I. Mani. kNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. Workshop on Learning from Imbalaned Datasets II, ICML.

Vita

Ya-Ju Fan

EDUCATION

09/2006 - 05/2010	**Ph.D.** in Industrial and Systems Engineering Rutgers University, Piscataway, NJ
09/2002 - 05/2005	**M. S.** in Industrial and Systems Engineering Decision Science/Operations Research University of Wisconsin-Madison, Madison, WI
09/1997 - 06/2001	**B. B. A.** in Production and Operations Management Fu Jen Catholic University, Taipei, Taiwan

OCCUPATION

09/2006 - 05/2010	**Research Assistant** Department of Industrial and Systems Engineering Rutgers University, Piscataway, NJ
09/2008 - 01/2010	**Teaching Assistant** Department of Industrial and Systems Engineering Rutgers University, Piscataway, NJ
01/2003 - 05/2005	**Research Assistant** Department of Industrial and Systems Engineering University of Wisconsin-Madison, Madison, WI
01/2004 - 05/2005	**Teaching Assistant** Department of Industrial and Systems Engineering University of Wisconsin-Madison, Madison, WI

PUBLICATIONS

Journal Articles/Proceedings

Y. J. Fan, E. Borenstein and W. A. Chaovalitwongse. "Feature Space Sample-Preserved k-Median Clustering and Its Feature Selection". Submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence*, April, 2010.

Y. J. Fan, W. A. Chaovalitwongse, O. Seref. "Bilinear Program Sample-Preserved k-Median Clustering for Arbitrary Distance Measures". Submitted to *INFORMS Journal on Computing*, March, 2010.

Y. J. Fan, and W. A. Chaovalitwongse. “Optimizing Feature Selection to Improve Medical Diagnosis”. *Annals of Operations Research*, 174(1), pp. 169-183, February, 2010.

W. Chaovalitwongse, **Y. J. Fan** and R.C. Sachdeo. “Novel Optimization Models for Abnormal Brain Activity Classification”. *Operations Research*, 56(6): 1450-1460, December, 2008.

W. Chaovalitwongse, **Y. J. Fan** and R. C. Sachdeo. “On the Time Series K-Nearest Neighbor Classification of Abnormal Brain Activity”. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 37(6): 1005-1016, November, 2007.

W. Chaovalitwongse, **Y. J. Fan** and R.C. Sachdeo. “Support Feature Machine for Classification of Abnormal Brain Activity”. *The Thirteenth ACM SIGKDD International Conference On Knowledge Discovery and Data Mining (SIGKDD 2007)*, pp. 113-122.

Book Chapters

Y. J. Fan, C. Iyigun and W. Chaovalitwongse. “Recent Advances in Optimization Models for Data Mining: Clustering and Classification”. *CRM Proceedings and Lecture Notes of the American Mathematical Society (AMS)*, 45: 67-93, 2008.

Y. J. Fan and W. Chaovalitwongse. “Deterministic and Probabilistic Optimization Models for Data Classification”. In C.A. Floudas and P.M. Pardalos (Eds.), *Encyclopedia of Optimization*, Vol. II. Springer, New York, pp. 694-702, 2009.