Spring 2009

Optimization techniques in data mining with applications to biomedical and psychophysiological data sets

Zhaohan Yu
University of Iowa

Recommended Citation
Yu, Zhaohan. "Optimization techniques in data mining with applications to biomedical and psychophysiological data sets." MS (Master of Science) thesis, University of Iowa, 2009.
https://doi.org/10.17077/etd.u18d629r

Follow this and additional works at: https://ir.uiowa.edu/etd
Part of the Industrial Engineering Commons

OPTIMIZATION TECHNIQUES IN DATA MINING WITH APPLICATIONS TO BIOMEDICAL AND PSYCHOPHYSIOLOGICAL DATA SETS

by

Zhaohan Yu

A thesis submitted in partial fulfillment
of the requirements for the Master of
Science degree in Industrial Engineering
in the Graduate College of
The University of Iowa

May 2009

Thesis Supervisor: Assistant Professor Pavlo Krokhmal

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

____

MASTER'S THESIS

____

This is to certify that the Master's thesis of

Zhaohan Yu

has been approved by the Examining Committee
for the thesis requirement for the Master of Science
degree in Industrial Engineering at the May 2009 graduation.

Thesis Committee: ____
Pavlo Krokhmal , Thesis Supervisor

____
Yong Chen

____
Andrew Kusiak

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Data Mining Literature Review

Data Mining (DM) is the process of automatic discovery of useful information in large data repositories (Tan, Steinbach, Kumar, 2005). It is especially appropriate for the fields where researchers do not have a theoretical understanding but large amounts of data. Generally, DM can be divided into two categories according to the objectives of algorithms: Classification Analysis and Association Analysis.

Classification is a procedure of dividing data sets into different classes based either on the knowledge of the predefined classes or just on structure of data set itself, which are called supervised classification (or classification for short), and unsupervised classification (cluster), respectively.

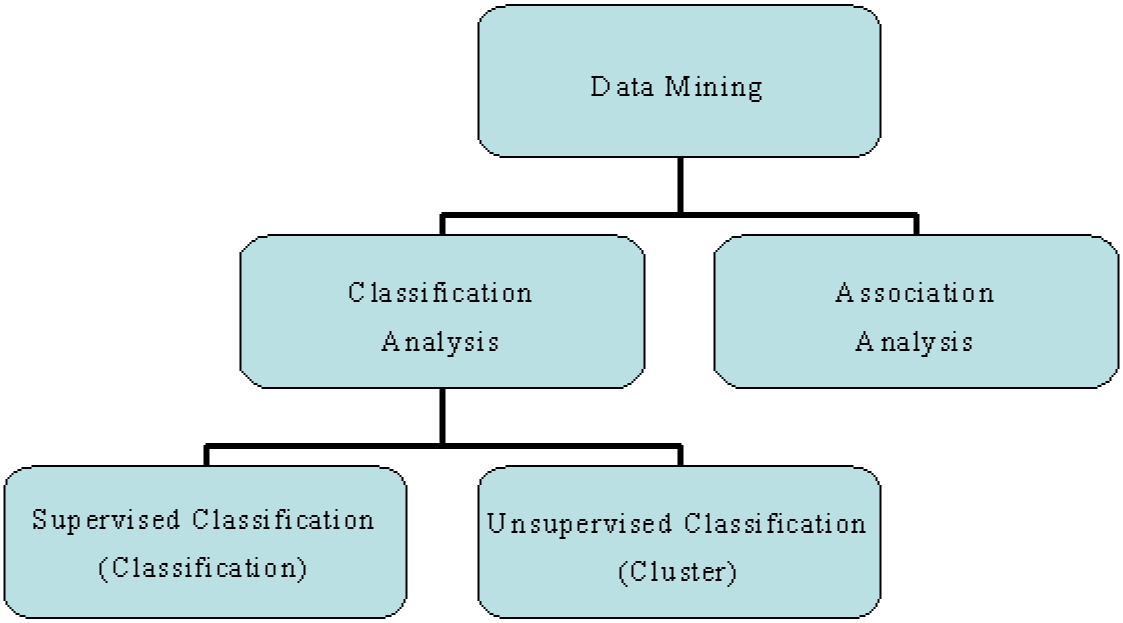

Data Mining
Classification Analysis
Association Analysis
Supervised Classification (Classification)
Unsupervised Classification (Cluster)

Figure 1.1 Structure of Data Mining

Association Analysis is used for discovering interesting relationships, which are called Association Rules, hidden in large data sets. Many fields such as web mining, document analysis, and bioinformatics have applied Association Analysis.

1.2 Optimization-Based Methods in Data Mining

Many DM methods involve with Mathematical Programming techniques. Optimization can contribute to DM in one of two ways: (1) Optimization can be a component of a larger DM process (Padmanabhan, 2003). For instance, in Artificial Neural Network (ANN), one of popular algorithms in pattern recognition, we minimize this function:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{1.1}$$

to obtain a set of parameters **w** in part of its procedure. (2) New DM techniques can be built using entirely optimization-based Method (Padmanabhan, 2003), which is also called Optimization-Based Approach (OBA) Data Mining.

OBA Data Mining techniques are applied mainly in Classification Analysis, whereas there are few algorithms in Association Analysis are based on Optimization. The reason may due to that objective of Association Analysis is not able to be directly formulated as optimization problem appropriately. Currently, several (OBA) algorithms are developed based on Support Vector Machine (SVM) method in Supervised Classification and also on k-Mean in Unsupervised Classification.

In certain classification cases, we could assume that we know the proper form of the discriminant functions, and use the samples to estimate the values of parameters of the classifier (Duda, Hart, Stork, 2001). If the assumption is based on linear model, then it is called linear discriminant function. For instance, in Support Vector Machine (SVM) method, the key is to discover a hyperplane (linear or non-linear, which corresponds to linear or non-linear functions) to separate data set in $\boldsymbol{R}^{n}$ space, and maximize the "margin" between different classes. Data sets can be categorized into separable cases and nonseparable cases.

1.2.1 Support Vector Machine Method

A linear SVM searches for a linear classifier $\mathbf{w}\cdot\mathbf{x}+b=1$ based on training data to label unknown data. This classifier is also known as a maximal margin classifier because it maximizes the "distance" between data points in different classes:

$$\begin{gathered}\min_{\mathbf{w}}\frac{\|\mathbf{w}\|^{2}}{2}\\ s.t.\ \mathbf{y}_i(\mathbf{w}\cdot\mathbf{x}_i+b)\geq 1, i=1,2...n\end{gathered}\tag{1.2}$$

It is a quadratic programming problem, and $\mathbf{w}$, $\mathbf{y}_i$, $\mathbf{x}_i$ are vectors, b is scalar, which can be solved by the standard Lagrange multiplier method (Tan, Steinbach, Kumar, 2005).

A more general form of linear SVM which can handle the condition that there is noise in training data or classes are overlapped is:

$$\begin{gathered}\min_{\mathbf{w}}\frac{\|\mathbf{w}\|^{2}}{2}+C(\sum_{i=1}^{n}\xi_i)\\ s.t.\ \mathbf{y}_i(\mathbf{w}\cdot\mathbf{x}_i+b)\geq 1-\xi_i, i=1,2...n\end{gathered}\tag{1.3}$$

where C and k are user-specified penalty parameters of misclassifying the training instances, and ξ_i represent errors introduced by the classifier. This nonlinear programming problem can be converted to Lagrangian dual problem and solved numerically by using quadratic programming techniques. The formulation of nonseparable case is general, and can be applied for both separable and nonseparable cases.

1.3 Linear programming and Multi-surface Method

1.3.1 K-Mean Method

K-Mean is a cluster analysis algorithm and could be treated as an optimization programming problem, which minimizes the sum of the “distances” of each point to its nearest centroid. The clustering problem is then formulated as in (Bradley, Fayyad, Mangasarian, 1999):

$$\begin{aligned}
&\min_{c^1,\ldots,c^k} \sum_{i=1}^{m} \min_{l=1,\ldots,k} \left\| x^i - c^l \right\| \\
&x^i,\ i = 1,\ \ldots m\ are\ given\ data\ po\,\mathrm{int}\,s \\
&c^l,\ \ l = 1,\ \ldots k\ \ are\ centroids\ of\ \ k\ clusters \\
&\|\bullet\|,\ is\ some\ arbitrary\ norm\ on\ R^n
\end{aligned} \qquad (1.4)$$

For different definitions of “distance”, there are different specific objective functions. The following table shows some choices for proximity function, centroid, and objective function that can be used in the basic k-mean algorithm. For instance, if we apply 1st-Norm (Manhattan Distance), then the centroid of a cluster will be the median of the data points belong to that cluster(so it is also called k-median algorithm), which have been proved mathematically.

Proximity Function	Centroid	Objective Function
Manhattan (L_1)	median	Minimize sum of the L_1 distance of an object to its cluster centroid
Squared Euclidean (L_2^2)	mean	Minimize sum of the squared L_2 distance of an object to its cluster centroid
cosine	mean	Maximize sum of the cosine similarity of an object to its cluster centroid
Bregman divergence	mean	Minimize sum of the Bregman divergence of an object to its cluster centroid

Source: Tan, Steinbach, Kumar, Introduction to Data Mining, 2005

Table 1.1 K-mean: Common choices for proximity, centroids, and objective functions

Although we can explicitly formulate the objective function for k-mean algorithm, the optimization problem can only solved by iteration instead of a closed-form solution. Moreover, the algorithm can only guarantee a local optimum. The algorithm is shown below. The two important issues in k-means algorithm are: (1) how to initial the K centroids because the final outcome is sensitive to initial starting condition, (2) how to update the centroids so they can converge to a local optimum or global optimum. In the basic k-means algorithm, because the initial K centroids are selected randomly, it usually takes several runs to guarantee that the result is optimal. Basic k-means and k-median algorithms are shown below:

1. Select K points as initial centroids
2. **Repeat**
3. Form K clusters by assigning each point to its closest centroid
4. Compute the centroid of each cluster

5. **Until** Centroids do not change

(Busygin, Prokopyev, Pardalos, 2007) included some definitions of k-means algorithm in their optimization-based approach for data classification. However, instead of applying k-means in clustering, it was applied in classification. Before assigning test data to certain clusters, they apply k-mean criterion to choose the feature weight of the training set.

$$\max_{x} \sum_{i=1}^{m} x_i$$

$$\textbf{s.t.} \quad \sum_{i=1}^{m} (a_{ij} - c_{ik'})^2 x_i \leq \sum_{i=1}^{m} (a_{ij} - c_{ik})^2 x_i \;\;\textbf{,}\; k = 1,\; \ldots,\; r \qquad (1.5)$$

$$0 \leq x_i \leq 1,\; i = 1,\; \ldots,\; m$$

In the formulation above, x_i is the weight for feature *i*. a_{ij} is the ith feature in jth sample and m is the total number of training data points. Moreover, c_{ik} is the ith feature of kth centroid which can be calculated by a_{ij}. *r* is the number of clusters. For a test sample b, we will assign it to class $S_{\hat{k}}$, if for all *k*=1 … *r* the following inequality hold:

$$\sum_{i=1}^{m} (b_i - c_{i\hat{k}})^2 x_i \leq \sum_{i=1}^{m} (b_i - c_{ik})^2 x_i \qquad (1.6)$$

1.3.2 Others Optimization Based Methods

Logical Analysis of Data is another OBA algorithm. It builds a classifier for a binary target variable based on learning a logical expression that can distinguish between positive and negative examples in a data set (Padmanabhan,Tuzhilin, 2003). If some attributes of data set are non-binary, cutoff value is applied to convert them into binary variable. And a table with all the binary attributes and target variables are obtained. The objective then becomes to explore a partially defined Boolean function (pdBf), with all the binary attributes as input and target variable as output.

Density Estimation method is based on Bayes' Theorem and can be formulated as mathematical programming problem. First, we assume an appropriate probability distribution for each cluster, and then we tune parameters of the distribution from minimizing the negative log-likelihood for the given data (Bradley, Fayyad, Mangasarian, 1999). An algorithm called Expectation-Maximization can be applied to find a local minimum for this problem.

1.4 Time-Series Data Mining

In the last decade there has been an explosion of interest in mining of time-series data. Literally hundreds of papers have introduced new algorithms to index, classify, cluster and segment time series (Keogh and Kasetty, 2003). Partly, such a great interest to time series mining is explained by the challenges that the classical methods of machine learning and clustering have faced when applied to time series data (Keogh and Pazzani, 1998). Another reason is that time-series data become readily available and increasingly important in many areas, such as economics and finance (prices of stocks), environmental sciences (daily sea-surface temperature and weather patterns) etc.

A time series is a sequence of real numbers, each number representing a value at a time point (Rafiei and Mendelzon, 1997). If there are several measures at the same time, then there are several sequences of real numbers corresponding to the same time period. This type of time-series data is called multidimensional data sequence. Typical examples of a multidimensional data sequence include video stream and image (Lee, etc, 2000).

One of the first papers in time-series data mining has been written by (Agrawal, Faloutsos and Swami, 1993). In this paper, the authors proposed to use Discrete Fourier Transform (DFT) to map time sequences from time domain to frequency domain and just keep the first few frequencies, then an algorithm called R*-tree (Bechmann, Kriegel, Schneider, and Seeger, 1990) was applied to index the sequences and efficiently answer similarity queries. R*-tree is a variant of R-tree, which is one of the most popular access methods (Guttman, 1984). DFT also acts as filter that eliminates noise and unimportant information in time-series data. Since then, more and more researchers accepted the idea that time-series data need to be preprocessed before performing the "actual" data-mining operations on it, thus the original data sequences are usually called "raw data". Some of the techniques used to preprocess the raw data sequences include DFT, Discrete Cosine Transform (DCT), Singular Value Decomposition (SVD), Haar Wavelet (Popivanov, Miller, 2002). All these techniques are involved with sigmoid shape functions, but a few papers also applied a piece-wise linear segmentation, which attempts to model the data as sequences of piecewise patterns (Keogh and Pazzani, 1998), (Geurts, 2001).

An important aspect of the data mining process is selection of the *similarity measure* that defines what is "similar" and what is not and is usually application dependent. Many researchers apply a traditional and intuitive Euclidean norm or generally, L_p norm to measure the difference between two data sequences, which is also one of the reasons why DFT is popular. According to Parseval's theorem, the Fourier transform preserves the Euclidean distance in the time or frequency domain (Agrawal, Faloutsos and Swami, 1993). However, the nature of time series data introduces also a number of challenges in selecting a "good" similarity measure, namely the presence of noise, offset translation, amplitude scaling, longitudinal scaling, linear drift, discontinuities (Keogh and Pazzani, 1998). Note that, "similarity measure" may be used in whole matching, i.e., for comparing equal length data sequences, as well as in the subsequence matching, where one looks for a subsequence of the large sequence that matches the query sequence best.

Generally, whole matching is considered to be easier than subsequence matching because for subsequence matching, certain length subsequence need to be extracted appropriately from whole sequence first, and then compares to sample sequence. There are a few methods developed in the literature that give a satisfactory solution on how to deal subsequence matching. In clustering of time-series data, similar problems exist. Some researchers formed subsequences by sliding a window through the time-series data to resolve this difficulty (Das, 1998).

According to (Keogh and Pazzani, 2002), most of time-series data mining tasks can be categorized into four kinds:

- Indexing (Query by Content): Given a query time series Q, and some similarity/dissimilarity measure D, find the nearest matching time series in database DB.
- Clustering: Find natural groupings of the time series in database DB under some similarity/dissimilarity measure D.
- Classification: Given an unlabeled time series Q, assign it to one of two or more predefined classes.
- Segmentation: Given a time series Q containing n data points, construct a model $\bar{Q}$ from K piecewise segments (K<< n) such that $\bar{Q}$ closely approximates Q.

A general procedure for processing time-series data is listed below:

1. Preprocessing data: filter noise, outliers, and divide whole sequences into equal length subsequences if it is necessary.
2. Processing data: Fourier transformation, Segmentation (piecewise linear approximation)
3. Similarity Measures: Euclidean distance, L_p norm
4. post-processing (optional)

CHAPTER 2
P-NORM MEASURES IN LINEAR PROGRAMMING DISCRIMINATION

2.1 Introduction

In the SVM method, nonlinear programming is employed to estimate the parameters of the linear discriminant function. In the paper “Robust linear programming separation of two linearly sets” (Bennett, Mangasarian, 1992), a linear programming algorithm was introduced, which is similar to the SVM but computationally simpler. The mathematical programming models of the linear programming and the SVM methods have similar constraints but differing objective functions. In the SVM, the objective of the equations (1.2) and (1.3) is to minimize the weighted sum of the margin of hyperplane and the error of misclassification, whereas the linear programming only provides an error-minimizing plane that minimizes an average sum of misclassified points belonging to two disjoint point sets in n-dimensional space. Although linear programming does not consider the margin of plane in its objective function, computational results do not show consistent disadvantages of the LP-based approach.

In this chapter we propose a new p-norm linear discrimination model that generalizes the model of (Bennett, Mangasarian, 1992) and reduces to linear programming problems with p-order conic constraints. We demonstrate that the developed model has nice methodological and computational properties (for examlpe, it does not allow for a null separating hyperplane when the sets are linearly separable). The presented approach for handling linear programming problems with p-order conic constraints relies on construction of polyhedral approximations for p-order cones. A case study on several popular data sets that illustrates the advantages of the developed model is conducted.

Consider two discrete sets $A, B \in \mathbb{R}^n$ comprised of m and k points, respectively: $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$, $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$. One of the principal tasks arising in machine learning and data mining is that of *discrimination* of these sets, namely, constructing a surface $f(\mathbf{x}) = 0$ such that $f(\mathbf{x}) \le 0$ for any $\mathbf{x} \in A$ and $f(\mathbf{x}) \ge 0$ for all $\mathbf{x} \in B$.

Of particular interest is the linear separating surface (hyperplane):

$$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} - \gamma = 0. \qquad (2.1)$$

Clearly, existence of such a separating hyperplane is not guaranteed; in general, a separating hyperplane that minimizes some sort of *misclassification error* is desired.

Observe that if points $\mathbf{y}^{(1)}, \mathbf{y}^{(2)} \in \mathbb{R}^n$ satisfy the inequalities

$$\mathbf{w}^{\mathrm{T}}\mathbf{y}^{(1)} - \gamma > 0, \quad \mathbf{w}^{\mathrm{T}}\mathbf{y}^{(2)} - \gamma < 0$$

for some $\mathbf{w}$ and γ, then they are located on the opposite sides of the hyperplane $\mathbf{w}^{\mathrm{T}}\mathbf{x} - \gamma = 0$. Consequently, the discrete sets A, $B \subset \mathbb{R}^n$ are considered *linearly separable* if and only if there exist $\mathbf{w} \in \mathbb{R}^n$ such that

$$\mathbf{w}^{\mathrm{T}}\mathbf{a}_i > \gamma > \mathbf{w}^{\mathrm{T}}\mathbf{b}_j \quad \text{for all } i = 1, \ldots, m,\ j = 1, \ldots, k,$$

with an appropriately chosen γ, or, equivalently,

$$\min_{\mathbf{a}_i \in A} \mathbf{a}_i^{\mathrm{T}}\mathbf{w} > \max_{\mathbf{b}_j \in B} \mathbf{b}_j^{\mathrm{T}}\mathbf{w}. \qquad (2.2)$$

Definition (2.2) is not suitable for use in mathematical programming models since it involves strict inequalities. However, the fact that the separating hyperplane can be scaled by any non-negative factor allows one to formulate the following result, whose proof we include for completeness.

Proposition 1 (Bennett, Mangasarian, 1992) *Discrete sets* $A, B \subset \mathbb{R}^n$ *represented by matrices* $\mathbf{A} = (\mathbf{a}_1, \dots, \mathbf{a}_m)^{\mathrm{T}} \in \mathbb{R}^{m \times n}$ *and* $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_k)^{\mathrm{T}} \in \mathbb{R}^{k \times n}$, *respectively, are linearly separable if and only if*

$$\mathbf{Aw} \geq \gamma\mathbf{e} + \mathbf{e}, \quad \mathbf{Bw} \leq \gamma\mathbf{e} - \mathbf{e} \quad \text{for some } \mathbf{w} \in \mathbb{R}^n, \gamma \in \mathbb{R}, \tag{2.3}$$

where ***e*** *is the vector of ones of the appropriate dimension,* $\mathbf{e} = (1, \dots, 1)^{\mathrm{T}}$.

Proof. Let A and B be linearly separable, then in accordance to definition (2.2), there exists $\mathbf{v} \in \mathbb{R}^n$ such that

$$\min_{i=1,\dots,m} \mathbf{a}_i^{\mathrm{T}}\mathbf{v} =: a_* > b^* := \max_{j=1,\dots,k} \mathbf{b}_j^{\mathrm{T}}\mathbf{v} \tag{2.4}$$

Denote $\mathbf{w} = 2\mathbf{v} / (a_* - b^*)$, and $\gamma = (a_* + b^*) / (a_* - b^*)$; then for any $\mathbf{a}_i \in A$

$$\mathbf{a}_i^{\mathrm{T}}\mathbf{w} - \gamma - 1 = \frac{2}{a_* - b^*}\mathbf{a}_i^{\mathrm{T}}\mathbf{v} - \frac{2a_*}{a_* - b^*} = \frac{2}{a_* - b^*}\left(\mathbf{a}_i^{\mathrm{T}}\mathbf{v} - \min_{i=1,\dots,k} \mathbf{a}_i^{\mathrm{T}}\mathbf{v}\right) \geq 0, \tag{2.5}$$

which means that $Aw - e\gamma - e \geq 0$. The second inequality in (2.3) follows analogously

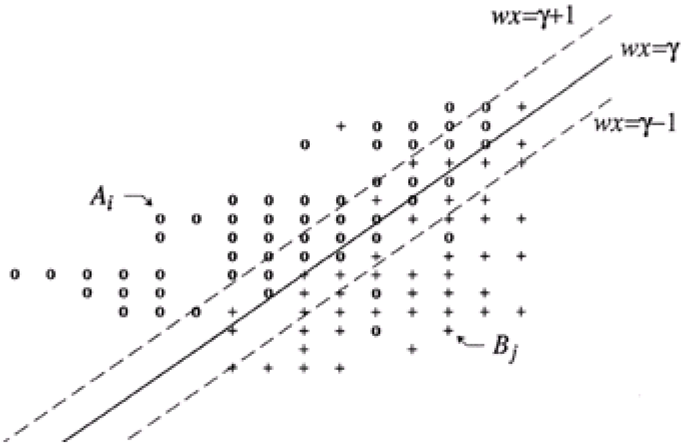


Figure 2.1An optimal separator $\mathbf{wx} = \gamma$ for linearly inseparable sets: A (o) and B (+) (Mangasarian, Bennett, 1991)

2.2 P-Norm Separation Model

In this paper generalize the *robust linear discrimination* model proposed by (Bennett, Mangasarian, 1992)

$$\begin{aligned} &\min_{\mathbf{w},\gamma,\mathbf{y},\mathbf{z}} \frac{\mathbf{e}^{\mathrm{T}}\mathbf{y}}{m} + \frac{\mathbf{e}^{\mathrm{T}}\mathbf{z}}{k} \\ &\text{s.t.} \quad \mathbf{Aw} - \gamma\mathbf{e} + \mathbf{y} \geq \mathbf{e}, \\ &\quad -\mathbf{Bw} + \gamma\mathbf{e} + \mathbf{z} \geq \mathbf{e}, \\ &\quad \mathbf{y} \geq \mathbf{0}, \ \ \mathbf{z} \geq \mathbf{0} \end{aligned} \tag{2.6}$$

The linear programming model (2.6) determines a hyperplane $\mathbf{w}^{*\mathrm{T}}\mathbf{x} - \gamma^* = 0$ that minimizes the average misclassification error. Indeed, in accordance to the definition (2.3), the points of sets *A* and *B* that violate (2.3) will correspond to the non-zero components of vectors y and z in the first and second constraints of problem (2.6), respectively.

This interpretation allows us to reformulate the optimization problem (2.2) in the form of a stochastic programming problem

$$\min_{(\mathbf{w},\gamma)\in\mathbb{R}^{n+1}} \{\mathrm{E}[(-\mathbf{a}^{\mathrm{T}}\mathbf{w}+\gamma+1)_+]+\mathrm{E}[(\mathbf{b}^{\mathrm{T}}\mathbf{w}-\gamma+1)_+]\}, \qquad (2.7)$$

where $\mathbf{a}$ and $\mathbf{b}$ are uniformly distributed random vectors with support sets *A* and *B*, correspondingly

$$\mathrm{P}\{\mathbf{a}=\mathbf{a}_i\}=\frac{1}{m},\quad \mathrm{P}\{\mathbf{b}=\mathbf{b}_j\}=\frac{1}{k}\quad \text{for all}\quad \mathbf{a}_i\in A,\ \mathbf{b}_j\in B, \qquad (2.8)$$

and $(x)_\pm=\max\{0,\pm x\}$. In this sense, the misclassification errors of points from *A* and/or B can be viewed as realizations of random variables $X_A=X_A(\mathbf{w},\gamma)$ and $X_B=X_B(\mathbf{w},\gamma)$, whose smaller values are preferred, and thus the parameters $\mathbf{w}$ and γ must be selected so as X_A and X_B assume values that are “small”.

As it is well known in stochastic programming and risk analysis, the “risk” associated with random outcome is often attributed to the “heavy” tails of the probability distribution. The risk-inducing “heavy” tails of probability distributions, are, in turn, characterized by the distribution's higher moments. Thus, if the misclassifications introduced by a separating hyperplane can be viewed as “random”, the misclassification risk may be controlled better if one minimizes not the average (expected value) of the misclassification errors, but their moments of order $p>1$. This gives rise to the following formulation for linear discrimination of sets *A* and *B*:

$$\min_{(\mathbf{w},\gamma)\in\mathbb{R}^{n+1}} \delta_1\,\|(-\mathbf{Aw}+\gamma\mathbf{e}+\mathbf{e})_+\|_p+\delta_2\,\|(\mathbf{Bw}-\gamma\mathbf{e}+\mathbf{e})_+\|_p,\quad p\in[1,+\infty] \qquad (2.9)$$

where $\|\cdot\|_p$ is the “functional” L_p norm, which in the probabilistic context can be written as

$$\| X \|_p = \begin{cases} (\mathrm{E}\,| X |^p)^{1/p}, & p \in [1, \infty) \\ \sup | X |, & p = \infty \end{cases} \tag{2.10}$$

Assuming again that points of the sets *A* and *B* are “equiprobable” (or, in other words, all points of set *A*, and, correspondingly, *B*, have equal “importance”), linear discrimination problem (11) can be written as follows

$$\begin{array}{ll} \min & \delta_1 \xi + \delta_2 \eta \\ \text{s. t.} & \xi \geq \| \mathbf{y} \|_p \\ & \eta \geq \| \mathbf{z} \|_p \\ & \mathbf{y} \geq -\mathbf{A}\mathbf{w} + \mathbf{e}\gamma + \mathbf{e} \\ & \mathbf{z} \geq \mathbf{B}w - \mathbf{e}\gamma + \mathbf{e} \\ & \mathbf{z}, \mathbf{y} \geq 0, \xi, \eta \geq 0 \end{array} \tag{2.11}$$

In the mathematical programming formulation (2.11), $\| \cdot \|_p$ denotes the “vector” norm in finite-dimensional space, i.e., for $\mathbf{x} \in \mathbb{R}^n$,

$$\begin{array}{ll} (| x_1 |^p + \ldots + | x_n |^p)^{1/p}, & p \in [1, \infty) \\ \max\{| x_1 |, \ldots, | x_n |\}, & p = \infty \end{array} \tag{2.12}$$

(in what follows, it will be clear from the context whether the “functional” or “vector” definition of p-norm is used). Note that in the formulation (2.11) the parameters δ_1 and δ_2 represent weights of misclassification errors. In this study, we consider *p*-norm linear separation models where δ_1 and δ_2 take values $\{ \delta_1 = \delta_2 = 1 \}$, $\{ \delta_1 = \frac{1}{m}, \delta_2 = \frac{1}{n} \}$ and $\{ \delta_1 = \frac{1}{m^p}, \delta_2 = \frac{1}{n^p} \}$.

Model (2.11) constitutes a linear programming problem with p-order conic constraints. Using the “vector” norm notation, formulation (2.11) can be more succinctly presented as

$$\min_{(\mathbf{w},\gamma)\in\mathbb{R}^{n+1}} \delta_1 \,\|(-\mathbf{Aw}+\mathbf{e}\gamma+\mathbf{e})_+\|_p + \delta_2 \,\|(\mathbf{Bw}-\mathbf{e}\gamma+\mathbf{e})_+\|_p \tag{2.13}$$

The p-conic programming linear separation model (2.11) shares many key properties with the LP separation model of (Bennett, Mangasarian, 1992), including the guarantee that the optimal solution of (2.11) is none-zero in $\mathbf{w}$ for linearly separable sets.

Proposition 2. *When sets A and B, represented by matrices* $\mathbf{A}$ *and* $\mathbf{B}$, *are linearly separable (i.e., they satisfy* (2.2) *and* (2.3) *), the separating hyperplane* $\mathbf{w}^{*\mathrm{T}}\mathbf{x}=\gamma^*$ *given by an optimal solution of* (2.11) *satisfies* $\mathbf{w}^* \neq \mathbf{0}$.

Proof:. By definition, separability of sets *A* and *B* immediately implies that at optimality $\mathbf{y}^* = \mathbf{z}^* = 0$, or, equivalently,

$$-\mathbf{Aw}-\gamma\mathbf{e}+\mathbf{e}\le 0 \qquad \text{and} \qquad \mathbf{Bw}-\gamma\mathbf{e}+\mathbf{e}\le 0$$

which is equivalent to the definition of linear separable $\mathbf{Aw}\ge\gamma\mathbf{e}+\mathbf{e}$, $\gamma\mathbf{e}-\mathbf{e}\ge\mathbf{Bw}$. To see that ($\mathbf{w}=0,\ \gamma$) cannot be optimal for (2.12), note that if we set $\mathbf{w} = \mathbf{0}$, then:

$$\min_{\mathbf{r}}\ \delta_1\left\|\gamma\mathbf{e}+\mathbf{e})_+\right\|_p + \delta_2\left\|-\gamma\mathbf{e}+\mathbf{e})_+\right\|_p > 0 \tag{2.14}$$

and the optimal value of (2.11) is zero. If one assumes that $\mathbf{w}^* = \mathbf{0}$, then the above inequalities require that

$$\gamma^* \le -1, \quad \gamma^* \ge 1.$$

This contradiction proves the proposition.

Secondly, the p-norm separation model (2.11) can produce a $\mathbf{w} = \mathbf{0}$ solution only in a rather special case that is identified by Theorem 1 below.

Theorem 1. *Assume that the* p*-order conic programming problem* (2.11) *is strictly feasible and, without loss of generality,* $\delta_2 k^{1/p} > \delta_1 m^{1/p}$. *Then, for any* $p \in (1,\infty)$ *the* p*-order conic programming problem (2) has an optimal solution where* $\mathbf{w}^* = \mathbf{0}$ *if and only if*

$$\frac{\mathbf{e}^{\mathbf{T}}\mathbf{A}}{m} = (\mathbf{t}^{'})^{\mathbf{T}}\mathbf{B},\ \mathbf{e}^{\mathbf{T}}\mathbf{t}^{'} = 1,\ \mathbf{t}^{'} \ge 0,\ \frac{\delta_2}{\delta_1 m^{1/p}} \ge \left\|\mathbf{t}^{'}\right\|_q \tag{2.15}$$

where q *satisfies* $\frac{1}{p}+\frac{1}{q}=1$. *In other words, the arithmetic mean of the points in A must be equal to some convex combination of points in B. In the case of* $\delta_2 k^{1/p} = \delta_1 m^{1/p}$ *condition reduces to*

$$\frac{\mathbf{e}^{\mathbf{T}}\mathbf{A}}{m} = \frac{\mathbf{e}^{\mathbf{T}}\mathbf{A}}{k} \tag{2.16}$$

i.e., the arithmetic means of the points of sets A and B must coincide

Proof: Consider the dual of the *p*-order conic programming problem (2.11):

$$\begin{gathered}
\max_{\mathbf{r},\mathbf{t},u,v}\ \mathbf{e}^{\mathbf{T}}\mathbf{r}+\mathbf{e}^{\mathbf{T}}\mathbf{t} \\
s.t.\ \mathbf{A}^{\mathbf{T}}\mathbf{r}-\mathbf{B}^{\mathbf{T}}\mathbf{t}=0 \\
\mathbf{e}^{\mathbf{T}}\mathbf{r}-\mathbf{e}^{\mathbf{T}}\mathbf{t}=0 \\
\mathbf{r}+\mathbf{u}=\mathbf{0} \\
\mathbf{t}+\mathbf{v}=\mathbf{0} \\
\delta_1 \ge \|\mathbf{u}\|_q \\
\delta_2 \ge \|\mathbf{v}\|_q \\
\mathbf{r} \ge \mathbf{0}, \mathbf{t} \ge \mathbf{0}
\end{gathered} \tag{2.17}$$

where q is such that $\frac{1}{p}+\frac{1}{q}=1$.

First, by assuming $\delta_2 k^{1/p} \geq \delta_1 m^{1/p}$ one does not lose any of generality because the roles of the sets A and B can be switched to obtain this inequality. Observe that point $(\mathbf{w}=0,\ \gamma,\ \mathbf{y},\ \mathbf{z})$ being an optimal solution for primal implies that the first two constraints of (2.18) become:

$$\begin{aligned} \mathbf{y} &\geq (1+\gamma)_+ \mathbf{e} \\ \mathbf{z} &\geq (1-\gamma)_+ \mathbf{e} \end{aligned} \tag{2.19}$$

Whereby the objective of the primal problem (2.20) takes the form

$$\min_{\gamma}\ \delta_1 m^{1/p}(1+\gamma)_+ + \delta_2 k^{1/p}(1-\gamma)_+ \tag{2.21}$$

Since $\delta_2 k^{1/p} \geq \delta_1 m^{1/p}$, then obviously, the objective value for primary problem is $2\delta_1 m^{1/p}$ with $\gamma = 1$. Also, because the primal is strictly feasible, duality gap is zero for the primal-dual pair (2.12) and (2.17). Then we have

$$\begin{aligned} \mathbf{e}^{\mathbf{T}}\mathbf{r} + \mathbf{e}^{\mathbf{T}}\mathbf{t} &= 2\delta_1 m^{1/p} \\ \mathbf{e}^{\mathbf{T}}\mathbf{r} - \mathbf{e}^{\mathbf{T}}\mathbf{t} &= 0 \end{aligned} \tag{2.22}$$

From (2.22) we have that $\mathbf{e}^{\mathrm{T}}\mathbf{r} = \delta_1 m_{,}$, and from the third constraint of the dual (2.17) we obtain $\mathbf{r} = -\mathbf{u}$, by substitution of which in the fifth constraint of (2.17) we obtain a system of equation and inequality that must be satisfied by vector $\mathbf{r}$ at optimality and $\mathbf{r} = -\mathbf{u}$, we know for vector $\mathbf{r}$

$$\begin{aligned} r_1 + r_2 + \ldots r_m &= m\delta_{\mathbf{1}} / m^{1/q} \\ r_1^q + r_2^q + \ldots r_m^q &\leq \delta_1^q \end{aligned} \tag{2.23}$$

To derive the solution of (2.23) , consider the following convex problem

$$\begin{aligned} &\min_{\mathbf{r}}\ \mathrm{g}(\mathbf{r}) = r_1^q + r_2^q + \ldots r_m^q \\ &s.t.\quad \mathrm{c}(\mathbf{r}) = r_1 + r_2 + \ldots r_m - m\delta_{\mathbf{1}} / m^{1/q} = 0 \end{aligned} \tag{2.24}$$

Using Lagrange multiplier method, we form the Lagrange function of problem (2.25)

$$L = r_1^q + r_2^q + \ldots r_m^q + \lambda(r_1 + r_2 + \ldots r_m - m\delta_{\mathbf{1}} / m^{1/q}) \tag{2.25}$$

whose saddle point is determined from equations

$$\begin{aligned} &\frac{\partial L}{\partial r_1} = q r_1^{q-1} - \lambda = 0 \\ &\qquad \vdots \\ &\frac{\partial L}{\partial r_m} = q r_m^{q-1} - \lambda = 0 \\ &\frac{\partial L}{\partial \lambda} = r_1 + \ldots r_m - m\delta_{\mathbf{1}} / m^{1/q} = 0 \end{aligned} \tag{2.26}$$

It is easy to see that $r_1 = r_2 = \ldots r_m = \delta_{\mathbf{1}} / m^{1/q}$ is the only stationary point for function g(**r**). And because g(**r**) is strictly convex continue, the saddle point is the minimum point and minimum value for the objective is δ_1^q. And at the same time, in (2.23) $g(\mathbf{r}) \le \delta_1^q$. Therefore, solutions to (2.23) with $r_1^q + r_2^q + \ldots r_m^q < \delta_1^q$ do not exist because minimum value of g(**r**) under constraint c(**r**) is equal to δ_1^q. Therefore, we can conclude that for (2.23), it has an unique solution, which is $r_1 = r_2 = \ldots r_m = \delta_{\mathbf{1}} / m^{1/q}$ or $\mathbf{r} = \dfrac{\delta_{\mathbf{1}}}{m^{1/q}}\mathbf{e}$

Furthermore, because $\mathbf{A}^{\mathbf{T}}\mathbf{r} - \mathbf{B}^{\mathbf{T}}\mathbf{t} = 0$ under solution ($\mathbf{w} = 0,\ \gamma,\ \mathbf{y},\ \mathbf{z}$):

$$
\begin{aligned}
&\mathbf{r}^{\mathrm{T}}\mathbf{A} = \mathbf{t}^{\mathrm{T}}\mathbf{B} \\
&\frac{\delta_1}{m^{1/q}}\mathbf{e}^{\mathrm{T}}\mathbf{A} = \mathbf{t}^{\mathrm{T}}\mathbf{B} \qquad (2.27) \\
&\frac{\mathbf{e}^{\mathrm{T}}\mathbf{A}}{m} = \frac{\mathbf{t}^{\mathrm{T}}}{\delta_1 m^{(1-1/q)}}\mathbf{B} = \frac{\mathbf{t}^{\mathrm{T}}}{\delta_1 m^{1/p}}\mathbf{B}
\end{aligned}
$$

Let $\mathbf{t}' = \frac{\mathbf{t}}{\delta_1 m^{1/p}}$, then from (2.27), $\frac{\mathbf{e}^{\mathrm{T}}\mathbf{A}}{m} = (\mathbf{t}')^{\mathrm{T}}\mathbf{B}$. Finally since $\mathbf{r} = -\mathbf{u},\ \mathbf{t} = -\mathbf{v}(\mathbf{r},\ \mathbf{t} \geq 0)$ and $\mathbf{e}^{\mathrm{T}}\mathbf{t} = \delta_1 m^{1/p}$:

$$
\begin{aligned}
&\frac{\mathbf{e}^{\mathrm{T}}\mathbf{A}}{m} = (\mathbf{t}')^{\mathrm{T}}\mathbf{B} \qquad (2.28) \\
&\mathbf{e}^{\mathrm{T}}\mathbf{t}' = 1 \text{ and } \frac{\delta_2}{\delta_1 m^{1/p}} \geq \left\| t' \right\|_q
\end{aligned}
$$

Therefore, the theorem holds. This theorem theoretically explains the reason that why $\delta_1 = 1/m^{1/p}$, $\delta_2 = 1/k^{1/p}$ should be chose. When $\delta_2 k^{1/p} \geq \delta_1 m^{1/p}$ then from the theorem we know that if the arithmetic mean of the points in A coincides with a convex combination of some points of B, the formulation will obtain a worthless optimal solution $\mathbf{w} = 0,\ \gamma,\ \mathbf{y},\ \mathbf{z}$,Moreover, if $\delta_2 k^{1/p} = \delta_1 m^{1/p}$, then the theorem degenerates to (2.16), which is the arithmetic mean of the points in *A* equals the arithmetic mean of the points in *B*. The advantage of this is that the satisfaction of (2.16) by a real world data set is much rarer than satisfaction of (2.28). In other words, using $\delta_2 k^{1/p} \geq \delta_1 m^{1/p}$ is more likely to obtain a null solution in real problem. The demonstration is in Figure 2.2.

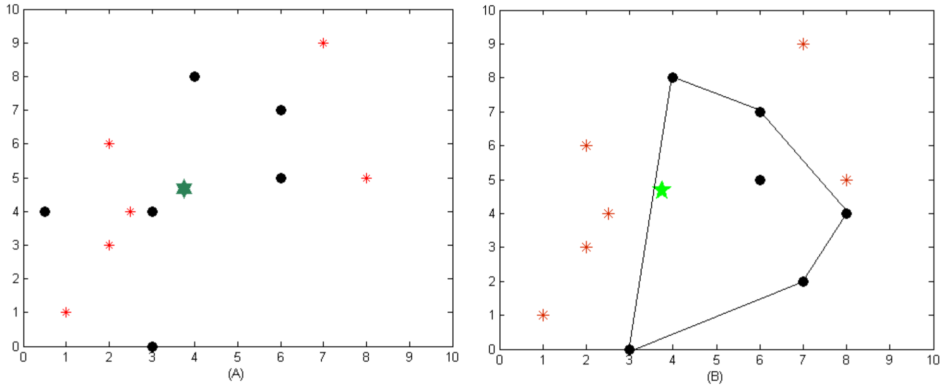


Figure 2.2 Demonstration of two data sets have the same arithmetic mean (A), and one data set has the arithmetic mean in the convex hull of data points from another data set (B)

In order to have the stricter condition (2.16) of the form for the occurrence of $\mathbf{w}^* = \mathbf{0}$ solution in the situation when the preferences for misclassification error are different for sets *A* and *B*, the p -norm linear discrimination model can be extended to the case where misclassifications of points in *A* and *B* are measured using norms of different orders:

$$\min_{(\mathbf{w},\gamma)\in\mathbb{R}^{n+1}} \delta_1 \| (-\mathbf{A}\mathbf{w} + \gamma\mathbf{e} + \mathbf{e})_+ \|_{p_1} + \delta_2 \| (\mathbf{B}\mathbf{w} - \gamma\mathbf{e} + \mathbf{e})_+ \|_{p_2}, \; p_{1,2} \in [1,\infty) \quad (2.29)$$

Intuitively, a norm of higher order places more “weight” on outliers; indeed, application of $p = 1$ norm would minimize the average misclassification error, in effect regarding all misclassifications as equally important. In contrast, application of the $p = \infty$ norm would minimize the largest misclassification error. Thus, by selecting appropriately the orders p and q in (2.29) one may introduce tolerance preferences on misclassifications in sets *A* and *B*. At the same time, it can be shown that the occurrence of $\mathbf{w}^* = \mathbf{0}$ solution in (2.29) would signal the presence of the aforementioned singularity about the sets A, B.

Namely, we it is easy to see from the proof of Theorem 1 that its statement carries over to model (2.29) practically without modifications.

2.2.1 Solving linear programming problems with p-order conic constraints using polyhedral approximations of p-order cones

When p is greater than 2, one way to solve this specific p-order cone programming problem is to approximate a p-order conic constraint by several linear constraints. For a set,

$$L^2 = \{(x_1, x_{2,} x_3) \,|\, \sqrt{x_1^2 + x_2^2} \le x_3\} \tag{2.30}$$

(Ben-Tal and Nemirovski, 2001) applied a polyhedral approximation of L^2, and via a way called "Tower of variables", any second order constraint can be expressed by several 3 dimensional second order constraints. Therefore, in terms of this polyhedral approximation, any second order constraints can be approximated by linear constraints.

Following Ben-Tal and Nemirovski's idea, (Krokhmal, 2007) developed a method that can approximate a 3-dimensional p-order cone in the positive orthant of $\mathbf{R}^3$ by a set of linear equalities.

For $p > 1$, $x_3 \ge (x_1^p + x_2^p)^{1/p},\ x_{1,} x_2, x_3 \ge 0,$ an internal approximation can be formulated as

$$\begin{aligned} & x_3(\sin^{2/p}\alpha_{i+1}\cos^{2/p}\alpha_i - \cos^{2/p}\alpha_{i+1}\sin^{2/p}\alpha_i) \\ & \quad \ge x_1(\sin^{2/p}_{\alpha_i+1} - \sin^{2/p}_{\alpha_i}) + x_2(\cos^{2/p}_{\alpha_i} - \cos^{2/p}_{\alpha_{i+1}}), i = 0 \ldots m-1 \end{aligned} \tag{2.31}$$

and an external approximation can be written in the form

$$\begin{aligned} & x_3(\cos^p\alpha_i - \sin^p\alpha_i)^{\frac{p-1}{p}} \\ & \quad \ge x_1\cos^{p-1}\alpha_i + x_2\sin^{p-1}\alpha_i, i = 0 \ldots m \end{aligned} \tag{2.32}$$

where $0 \equiv \alpha_0 < \alpha_1 < \ldots < \alpha_m \equiv \pi / 2$ Therefore, to approximate a $2^d + 1$-dimensional p-order conic constraint:

$$\begin{aligned} & t \ge (x_1^p + \ldots + x_J^p)^{1/p} \\ & t, x_j^p \ge 0, \; j = 1 \ldots J (J = 2^d) \end{aligned} \tag{2.33}$$

First, we represent this constraint by a set of 3-dimensional p-order conic inequalities

$$\begin{aligned} & x_j^{(k)} \ge [(x_{2j-1}^{(k-1)})^p + (x_{2j}^{(k-1)})^p]^{1/p} \\ & j = 1 \ldots 2^{d-k}, k = 1 \ldots d, \end{aligned} \tag{2.34}$$

where $x_1^{(d)} \equiv t, x_j^0 \equiv x_j (j = 1 \ldots 2^d)$. On the second step, every 3-dimensional p-order conic constraint is approximated either by the internal approximation or the external approximation. The final LP approximation is shown in (2.35).

$$\begin{aligned} \min_{\mathbf{w},\theta,\mathbf{y},\mathbf{z}} \;\; & \delta_1 u + \delta_2 v \\ s.t. \;\; & \mathbf{Aw\text{-}e}\gamma + \mathbf{y} \ge \mathbf{e} \\ & \mathbf{\text{-}Bw + e}\gamma + \mathbf{z} \ge \mathbf{e} \\ & f_j^{(k)} (\sin^{2/p} \alpha_{e+1} \cos^{2/p} \alpha_e - \cos^{2/p} \alpha_{e+1} \sin^{2/p} \alpha_e) \\ & \quad \ge f_{2j-1}^{(k-1)} (\sin_{\alpha_e+1}^{2/p} - \sin_{\alpha_e}^{2/p}) + f_{2j}^{(k-1)} (\cos_{\alpha_e}^{2/p} - \cos_{\alpha_{e+1}}^{2/p}), \; j = 1, \ldots, 2^{d-k}, \; k = 1, \ldots, d, \; e = 0, \ldots l - 1 \\ & \mathrm{g}_h^{(i)} (\sin^{2/p} \alpha_{e+1} \cos^{2/p} \alpha_e - \cos^{2/p} \alpha_{e+1} \sin^{2/p} \alpha_e) \\ & \quad \ge g_{2h-1}^{(i-1)} (\sin_{\alpha_e+1}^{2/p} - \sin_{\alpha_e}^{2/p}) + g_{2h}^{(i-1)} (\cos_{\alpha_e}^{2/p} - \cos_{\alpha_{e+1}}^{2/p}), \; h = 1, \ldots, 2^{c-i}, \; i = 1, \ldots, c, \; e = 0, \ldots l - 1 \\ & f_1^{(d)} = u \\ & f_{2^{d-k+1}-1}^{(k-1)} = y_{d-k} \\ & g_1^{(c)} = v \\ & g_{2^{c-h+1}-1}^{(i-1)} = z_{c-i} \\ & \mathbf{y} \ge 0, \mathbf{z} \ge 0 \end{aligned} \tag{2.35}$$

Finally, we can take the advantages of certain commercial software to solve this linear programming problem. Note, for these p-order conic constraints (2.33), which $J \neq 2^d + 1$, theoretically ($f(J)-J$) number of slack variables can be added into conic constraints to satisfy this condition $J=2^d+1$, and set as zero in additional linear equalities, where $f(J)=2^{(\log_2 J)_{round}}$. However, when J is comparable large, a huge number of slack variables are needed; therefore, we deal it in a different way in the computational procedure. Instead of adding all the slack variables at one time, we put one slack variable a time during each level of “the tower” depending on whether or not the number of the variables in that level is even or odd. For instance, for a conic constraint:

$$\begin{aligned} & t \ge (x_1^p + \ldots + x_J^p)^{1/p} \\ & t, x_j^p \ge 0,\ j=1 \ldots J(J \ne 2^d) \end{aligned} \tag{2.36}$$

If J is an even number, then no slack variable is needed for the first “level of tower”, then

$$\begin{aligned} & x_j^1 \ge [(x_{2j-1}^0)^p + (x_{2j}^0)^p] \\ & j = J/2 \end{aligned} \tag{2.37}$$

If J is an odd number, then one slack variable is added, so

$$\begin{aligned} & x_j^1 \ge [(x_{2j-1}^0)^p + (x_{2j}^0)^p] \\ & x_J = 0 \\ & j = (J+1)/2 \end{aligned} \tag{2.38}$$

So on and so forth for the other levels. It is obvious that approach can significantly reduce the number of slack variables when J is large. When j=700, for first approach 1024-700=324 slack variables are needed, but for second method, only 3 slack variables are necessary.

2.3 Data Set Information and Computation Results

Several real-world data sets from UCI Machine Learning Repository (University of California-Irvine) are classified by the method we proposed. The method is implemented in C++ environment and CPLEX Solver (ILOG CPLEX® 10.0) is used to solve the linear programming problem formulated.

Wisconsin Breast Cancer Data Set (Original): This breast cancer databases was obtained from the University of Wisconsin Hospitals, Madison by Dr. William H. Wolberg. There are 10 feature values and an ID number for each data point in the dataset, which are obtained by medical examination on certain breast tumors. There are 699 data points in the data sets, but because some values are missing, only 682 data points are used in the experiment. The whole data set consist of two classes of data points, 444 (65.1%) data points represent benign tumors, and the rest of 238 (34.9%) points correspond to malignant cases. Other information is included in Table 2.1.

Data Set Characteristics:	Multivariate	Number of Instances:	699	Area	Life
Attribute Characteristics:	Integer	Number of Attributes:	10	Data Donated	7/15/1992
Associated Tasks:	Classification	Missing Values?	Yes	Number of Web Hits:	543

Table 2.1 Description of Wisconsin Breast Cancer Data Set (Original)

In this numerical experiment, we use α and β to indicate benign and malignant data sets, we randomly divide each set into a training set possessing 2/3 of the data and a test set including the remaining 1/3. So if M data points are in set α and N data points are in set β, then the numbers of benign and malignant data points in the training set are $m = (2/3)M$ and $n = (2/3)N$. The p-norm linear programming formulation is formulated base on the training set and the obtained optimal classifier classify the test set. For every specific value *p*, this procedure repeats 10 times and average error percentage for whole data set is calculated. *p* increases from 1 to 5 by 0.1 step.

For a data point belongs to dataset α, the probability of this data point being misclassified is $e_\alpha/m'\times100\%$, and similarly, the probability is $e_\beta/n'\times100\%$ for a data point belong to β, where $e_{\alpha,}$ e_β indicate the number of data points misclassified in set α and β , and *m'*, *n'* are the number of test data points in set α and β.

Therefore, for an arbitrary point in the whole data set, the probability of being misclassified is equal to $P_\alpha\times e_\alpha/m'+P_\beta\times e_\beta/n'=(e_\alpha+e_\alpha)/(m'+n')$, where P_α, P_β are the probabilities of a random point belonging to set α or β. Then we define error percentage as $\varepsilon=[(e_a+e_b)/(m'+n')]\times100\%$.

The complete results are shown in the Appendix. In Table 2.2, optimal order and optimal average error percentage for different objective weights are collected. As Table 2.2 shows, when $\delta_1=1/m,\delta_2=1/n$ and *p*=1.9, we obtain the lowest average error percentage 2.77%. Moreover, compare this with the average error percentage obtained by $\delta_1=1/m,\delta_2=1/n$, *p*=1, which is also the average error percentage obtained by original formulation, we can see that the average error percentage decrease 3.48%.

In the results appended, with the same order but different values of δ_1, δ_2, objective coefficients $\delta_1=1/m,\ \delta_2=1/n$ "most time" obtain a better result than others'.

	$\delta_1 = \delta_2 = 1$	$\delta_1 = 1/m$ $\delta_2 = 1/n$	$\delta_1 = 1/m^p$ $\delta_2 = 1/n^p$
Optimal Order	2	1.9	1.5
Optimal Error %	3.22%	2.77%	2.82%
The percentage of error decreased Compare w/ order 1	8.78%	3.48%	1.74%

Table 2.2 Comparison of classification error between different orders and different δ_1, δ_2 for Wisconsin Dataset

Similar tests also run on Pima Indians Diabetes Data Set, Connectionist Bench (Sonar, Mines vs. Rocks) Data Set, and Ionosphere Data Set, all of which are obtained from UCI Machine Learning Repository. Note, among these tests only $\delta_1 = \delta_2 = 1$ and $\delta_1 = 1/m^p, \delta_2 = 1/n^p$ are used. The results of average error percentage over all the orders of *p* are shown in appendix. The value of parameter *p* increases from 1 to 5 by every 0.1 step.

Coefficients Data Sets	$\delta_1 = \delta_2 = 1$	$\delta_1 = 1/m^p$ $\delta_2 = 1/n^p$	Results from other algorithms
Ionosphere	16.60%	18.13%	12.3% (Radivojac, Obradovic, Dunker, Vucetic, 2004)
Pima	29.51%	31.07%	26.3% (Radivojac, Obradovic, Dunker, Vucetic, 2004)
Sonar	30.23%	30.45%	24% (Tan, Dowe, 2004)

Table 2.3 Average classification error with different δ_1, δ_2 for different data sets

As the results shown above, these data sets are "linear inseparable", that means an optimum hyperplane obtained in vector space, cannot classify "the most of" data points correctly. Therefore, a nonlinear classifier or multi-hyperplane is necessary for this kind of data sets. One thing need to be point out is that when compare to the result obtained by other methods, our result is only slight worse, but our method is much simpler both in theoretical and practical.

CHAPTER 3
LINEAR DISCRIMINANT FUNCTION, K-NEAREST NEIGHBOR METHODS AND NEURAL NETWORKS IN CLASSIFYING PSYCHOPHYSIOLOGICAL DATA

3.1 Data Set Information and Characteristics

3.1.1 Background

Psychophysiology is a branch of physiology which is focused on the relationship between mental (psyche) and physical (physiological) processes; it studies the interaction between mind and body. Applied psychophysiology investigates the effects of emotional states on the central nervous system, by observing and recording data on such physiological processes as sleep rhythms, heart rate, gastrointestinal functioning, immune response, and brain function. Techniques that are used to measure such factors include electroencephalography (EEG), magnetic resonance imaging (MRI), computerized axial tomography (CAT) scans, electrocardiography (ECG), and electrooculography (EOG).

ECG records the electrical activity of human heart over time, EEG measures the electrical activity of human brain and EOG is a technique for measuring the resting potential of the retina. Electroencephalogram, electrocardiogram and electrooculogram are the resulting signals measured by EEG, ECG, and EOG. These psychophysiological measures techniques are employed for classifying cognitive work load in laboratory and real-world setting. Because the EEG, ECG, and EOG data can be recorded without interfering task performance of the human subject, they are suitable for estimating operator functional state. Since they are recorded in real time, they are time-series data sets.

3.1.2 Data Set Information

The data set we used in our research comes from a project in which the human operators performed a set of tasks of varying levels of cognitive difficulty. The specific tasks concerned navigation and guidance of unmanned aerial vehicles (UAVs) in a simulated environment. The goal of this study is to estimate the operators' cognitive states by analyzing their psycho-physiological measurements. The data were collected during the simulation tasks from three participants, 'A', 'E' and 'F', each of whom completed two trials, denoted as '01' and '02'. For instance, the second trial completed by participant E is denoted as 'E02'. The simulation experiments and data collection have been performed at the Air Force Research Lab at Wright-Patterson AFB (Dayton, OH).

In each trial, ECG, horizontal EOG, vertical EOG and five channels of EEG signals were recorded. The EEG data were recorded from the scalp sites F7, Fz, Pz, T5, and O2 of the 10/20 electrode system using an Electro Cap. The EOG electrodes were placed above and below the midline of the right eye to record vertical movement and blink activity. The ECG electrodes were placed on the sternum and the left clavicle. The sampling rate was 200 Hz with a band pass from 0.5 to 52.4 Hz.

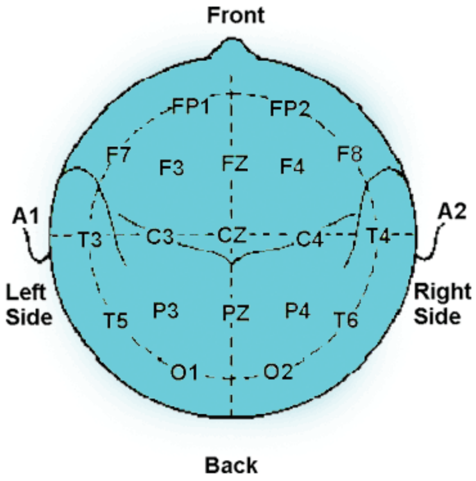


Figure 3.1 EEG 10-20 System Diagram

During the simulation, the operators have been subjected to three tasks of zero, low, and high levels of cognitive workload, indicated by 0, 1, and 2:

'0': simulation running, no cognitive task being performed

'1': simulation running, low cognitive workload task being performed.

'2': simulation running, high cognitive workload task being performed.

These are the three classes we aim to classify. Sometimes we also refer to task 0 as None Cognitive Load (NCL) task and tasks 1and 2 as Cognitive Load (CL) task. A time period in which the test subjects continually performs under the same level of cognitive workload is called a “task”. For instance, if a subject works under the level 1 cognitive workload for 20 seconds, then we say that a task 1 happened for 20 seconds. A fraction of this time period is called a sub-task. In each trial, a subject performs under different situations for about 20 minutes, which is consisted of eight tasks 0 (no cognitive load), four tasks 1 (low cognitive load), and four tasks 2 (high cognitive load). The tasks 0 last for about 50 to100 seconds, while the tasks 1 and 2 usually take about 20 seconds. Note that this asymmetry in the durations of NCL and CL tasks may have an impact on the computational results of our study.

3.1.3 Data Set Transformation and Characteristics

The Fourier Transform is a mathematical operation that transforms a signal from the time domain to the frequency domain, and vice versa. We are accustomed to time-domain signals in the real world. In the time domain, the signal is expressed with respect to time. In the frequency domain, a signal is expressed with respect to frequency. We apply the Discrete Fourier Transform (DFT) to convert our time-series data from the time domain to the frequency domain, and analyze the magnitudes over certain frequency band (e.g. from 1 Hz to 10 Hz) among the different tasks.

Discrete Fourier Transform takes as input a sequence of N real or complex numbers $\mathbf{x}=(x_1,x_2,...,x_N)$ and generates its N-dimensional transformation $\mathbf{X}=(X_1,X_2,...,X_N)$. The DFT and its inverse are given by the following formulas:

$$\begin{aligned} X(k) &= \sum_{j=1}^{N} x(j)\omega_N^{(j-1)(k-1)} \\ x(j) &= (1/N)\sum_{k=1}^{N} X(k)\omega_N^{-(j-1)(k-1)} \end{aligned} \tag{3.1}$$

where $\omega_N = e^{(-2\pi i)/N}$ and $i = \sqrt{-1}$.

Since the goal of this research is to find out the operators' cognitive states at real time, DFT should be applied to the data over a relatively short time period. In addition, the length of the time period shouldn't vary, which means the numbers of data collected in these two periods are also the same assuming the sample rates are the same; otherwise it is not rational to compare the magnitudes between two different long time periods. A straightforward application of the DFT to signals of different lengths (or even the same periodic signal, but over time windows of different length) may result in different outputs (i.e., magnitudes of the frequencies comprising the signal). Thus, a special precaution needs to be exercised when working with signals of different durations. In practice, this difficulty can be circumvented by fixing the length of the input data for the DFT transform.

The main application of the Fourier transform in signal analysis is to obtain a spectral decomposition of a given signal into a set of harmonic frequencies that this signal is comprised of. When applied to a stationary periodic signal that is a mixture of several stationary harmonics, the graph of DFT typically looks as a series of "peaks", where the location of each peak identifies the frequency of a harmonic that is present in the input signal, and the "height" of the peak corresponds to this harmonic's amplitude.

This method works very well even when the periodic signal may contain a substantial noise component.

However, the resolving power of the DFT diminishes when the input signal changes dynamically in time. In the data set used in this study, all signals, except, maybe, the ECG signal, exhibit a high degree of non-stationary and noisiness. In such a case, to identify the composition of the input signal at any given time moment, the DFT can be applied over shorter time windows. The length of the time window over which the DFT is computed then becomes a crucial issue, and the computational results may significantly depend on the choice of this parameter. If one chooses a large time window, the results may be insensitive to the temporal changes in psychophysiological measurements caused by the changes in the operator's cognitive state. If the window is too short, the resulting DFT transforms may be too noisy, and some lower frequency information in the signals may be lost.

Moreover, the length should be decided before the DFT. If the lengths of these two or several signals are about the same, then choosing the shortest length among all these signals is a good choice. However, if the lengths of these signals are rather different, then we may lose quite substantial amount of information by applying the method mentioned. In this case, we select a comparatively small length, and divide all the signals by this length. Then we apply DFT on all these "standard- length" signals, so the magnitude of a signal is the average of the magnitudes of all the "standard length" signals. The limitation of this method is that if the standard length is too short, then some lower frequency information in the signals may be lost. Figure 3.2 provides a graphical illustration of the method that we used to process the time-series data set before DFT. We introduce two time windows, a larger one of length W_1, and a shorter window of length W_2. For instance, the data in Figure 3.2, which is the Fz EEG signal, recorded from subject A in task 0. In this particular case, we first divide the data of task 0 into subtasks of equal length W_1=3000, which corresponds to time interval of 15 seconds (since the sampling frequency is 200 Hz). Then, each of the W_1-long sub-tasks is divided into six windows, each of which includes W_2=500 data points.

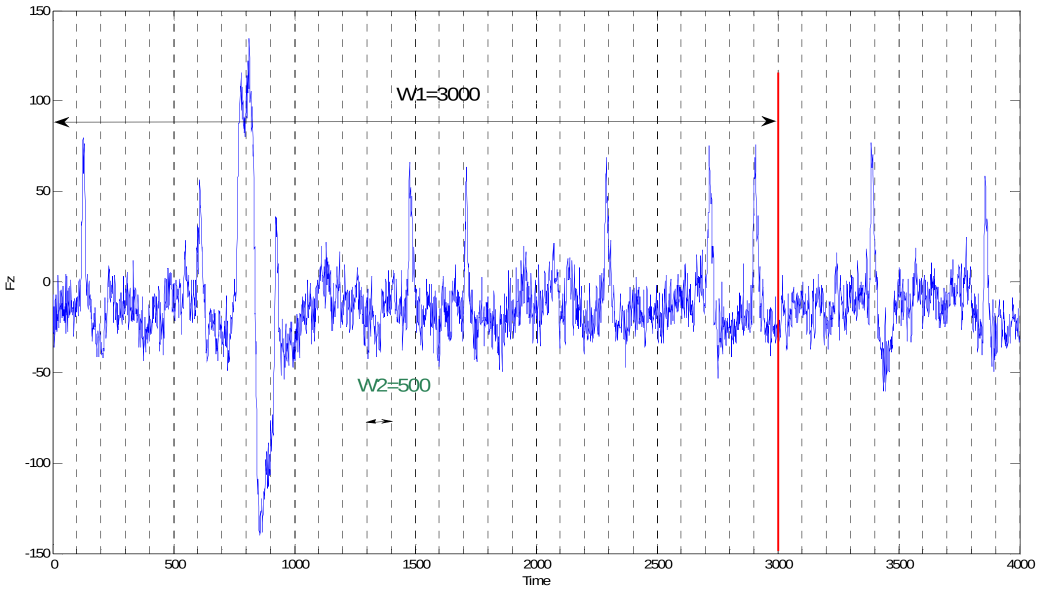
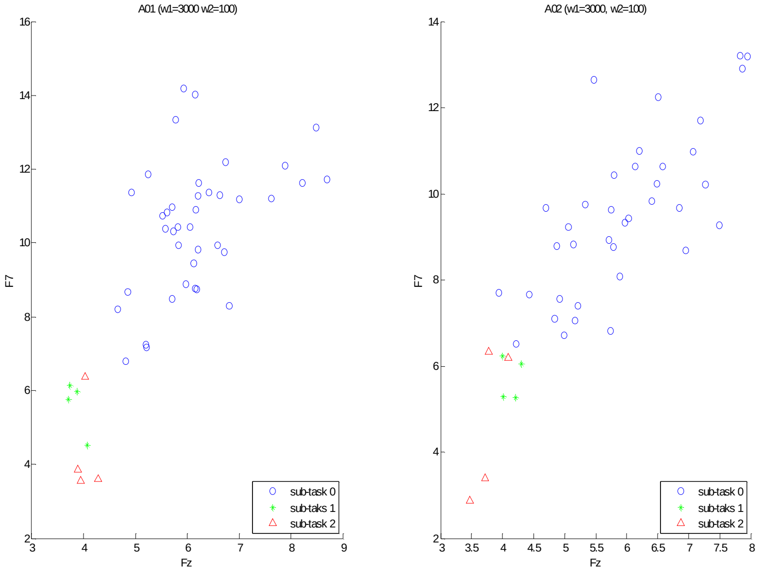


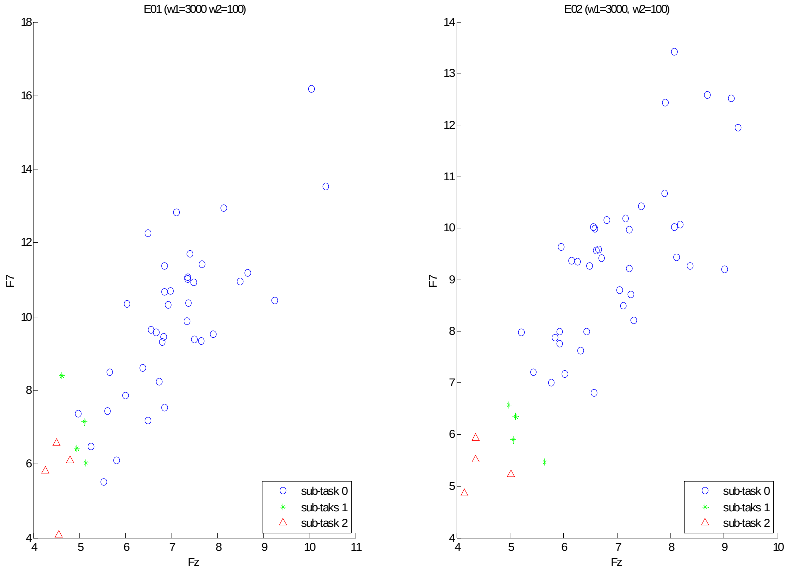
Figure 3.2 Illustration of data preparation before DFT

After investigation, two EEG signals (Fz, F7) have interesting patterns, and could be used to classify the data. We plot the magnitudes of these two signals in a two-dimensional space for each task, and obtain plots for each trial as in Figure 3.3. As just mentioned, we split a certain task into equal long sub-tasks, divide these sub-tasks into certain long intervals, and then apply DFT to transform these intervals from time domain to frequency domain. Finally, the average magnitudes over certain frequency band (1 Hz to 10 Hz here) are computed, and the magnitudes of a certain sub-tasks are the average magnitudes of these intervals in this sub-task. In Figure 3.3, it shows different tasks have different magnitudes in these two signals. The task 0 usually has the biggest magnitudes in both Fz and F7 among the three, followed by task 1. So the conclusion is the higher cognitive level the task is in, the smaller magnitudes it has. Also, although there are similarities between two trials of the same subject, for two subjects, we cannot say the magnitudes of the signals are related. Similar pattern also be discovered in two EOG signals.

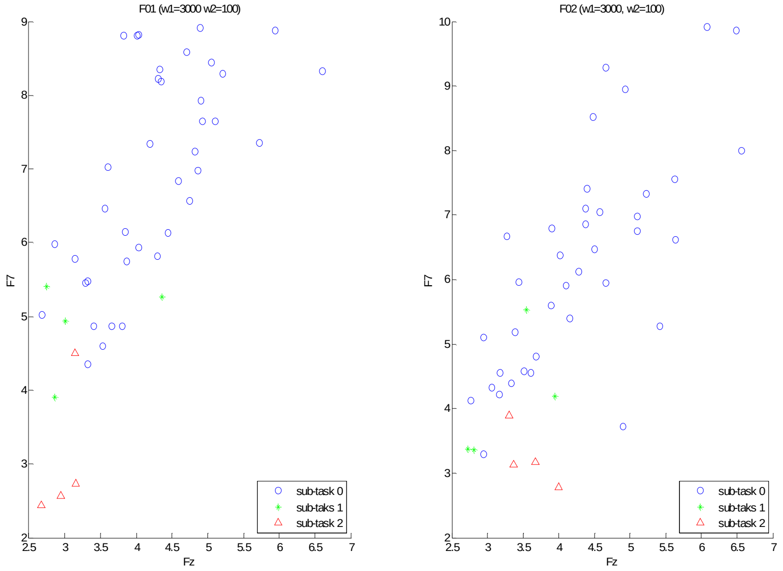
Moreover, the smallest theoretical length of input data for calculating the average magnitude from 1 Hz to 10 Hz is 20. Because the sample frequency of data is 200 Hz, after using DFT convert 20 data points from the time domain to the frequency domain, 10 data points are outputted, which represent the magnitudes of frequencies from 0 Hz to 190 Hz by every 10 Hz. In this case, the magnitude at 10 Hz is used to approximate the average magnitude from 1 Hz to 10 Hz. When the length is decided, a certain task is divided by this length and several time periods are obtained for the task. Then DFT is performed on the data of these equal long time periods and the average magnitude over certain frequencies is calculated.



Subject A



Subject E



Subject F

Figure 3.3 DFT transforms of Fz and F7 signals in the frequency band 1~10 Hz for subjects A, E, and F

The information in Table 3.1 shows the frequency bands for EEG signals defined by some researchers and the brain activities corresponding to these certain bands. In general, low frequency signals indicate low cognitive activities, and high frequency signals are related to the activities which require thinking and concentration. Also, we can see that low frequency bands (0~12) include Delta, Theta and Alpha, and high frequency bands (>12) incorporate Beta and Gamma. In our research, we discover that low frequency signals perform better in classifying no-cognitive and cognitive tasks than higher frequency signals, which is also the reason why a frequency band from 1 Hz to 10 Hz is applied.

Type	Frequency (Hz)	Related Activities
Delta	0 – 3 Hz	▪ Adults slow wave sleep ▪ In babies
Theta	4 – 7 Hz	▪ Young children ▪ Drowsiness or arousal in older children and adults ▪ Idling
Alpha	8 – 12 Hz	▪ Relaxed/reflecting ▪ Closing the eyes
Beta	12 – 30 Hz	▪ Alert/working ▪ Active, busy or anxious thinking, active concentration
Gamma	34 – 100 Hz	▪ Certain cognitive or motor functions

Table 3.1 Explanation of EEG frequency Bands

In order to see the inter-relations between these eight signals the correlation values are calculate between each signal in each trial. The values for subject A are shown in the table 3.1, others are listed in the appendix. From the correlation coefficient, it is noticed that for subject A and E, vertical EOG is strongly correlated with Fz (>0.68), and Horizontal EOG closely related to F7 (>0.7). And on the other side, Fz and F7 only have a medium correlation coefficient (usually < 0.5). However for subject F, only the correlation between Horizontal EOG and F7 EEG exists (>0.79). These statistics can also explain that why there are similar patterns between VEOG, HEOG and Fz, F7 EEGs. Note, the correlation coefficients are calculated before the DFT. We also calculated the coefficients after the DFT, the correlation values become even higher, which may be due to some unique characteristics are tossed away in the process.

A01	1(ECG)	2(VEOG)	3(HEOG)	4(Fz)	5(F7)	6(Pz)	7(P7/T5)	8(O2)
1	1	-0.0089	0.0102	-0.0734	-0.0277	-0.085	-0.0675	-0.0691
2	-0.0089	1	-0.4725	**-0.716**	**-0.7652**	-0.1095	-0.2623	0.1504
3	0.0102	-0.4725	1	-0.048	**0.8768**	-0.3512	0.0055	-0.5768
4	-0.0734	-0.716	-0.048	1	0.3948	0.5663	0.4795	0.3203
5	-0.0277	-0.7652	0.8768	0.3948	1	-0.1166	0.1957	-0.4052
6	-0.085	-0.1095	-0.3512	0.5663	-0.1166	1	**0.7155**	**0.8212**
7	-0.0675	-0.2623	0.0055	0.4795	0.1957	0.7155	1	0.5837
8	-0.0691	0.1504	-0.5768	0.3203	-0.4052	0.8212	0.5837	1

A01

A02	1(ECG)	2(VEOG)	3(HEOG)	4(Fz)	5(F7)	6(Pz)	7(P7/T5)	8(O2)
1	1	-0.0085	0.0268	-0.0917	-0.021	-0.0974	-0.0775	-0.0849
2	-0.0085	1	-0.4712	**-0.6821**	**-0.7454**	-0.0985	-0.2559	0.146
3	0.0268	-0.4712	1	-0.0929	**0.8794**	-0.3753	-0.023	-0.5828
4	-0.0917	-0.6821	-0.0929	1	0.3329	0.5726	0.4675	0.3405
5	-0.021	-0.7454	0.8794	0.3329	1	-0.1551	0.1597	-0.4194
6	-0.0974	-0.0985	-0.3753	0.5726	-0.1551	1	**0.7014**	**0.8256**
7	-0.0775	-0.2559	-0.023	0.4675	0.1597	0.7014	1	0.5817
8	-0.0849	0.146	-0.5828	0.3405	-0.4194	0.8256	0.5817	1

A02

Table 3.2 Raw signal correlations in trials A01 and A02 of subject A

Furthermore, the distribution of each signal from the raw data is also plotted in Figure 3.4, some appealing phenomenon is found: in each task the signals are unimodal distributed and has comparable mean, however the difference between task 0 and task 1, 2 is that task 0 has a two side heavy-tails. To utilize this interesting point in classification, we can measure the variance of each task or measure the variance in certain intervals, like the tail areas of the distribution. However, a major drawback is that it not always applicable to all the tasks.

So far, there are only four signals out of eight are found interesting for the classification, vertical, horizontal EOGs and Fz, F7 EEGs. The ECG is unsurprisingly unrelated with the cognitive level, and the other EEG signals seem to react randomly on different tasks. The understanding is that for certain functions only left hemisphere have, such like the analytical and logical abilities. And from the task description, we know that the tasks are highly demanded for analytical and logical abilities and Fz, F7 positions are located on the left side of brain.

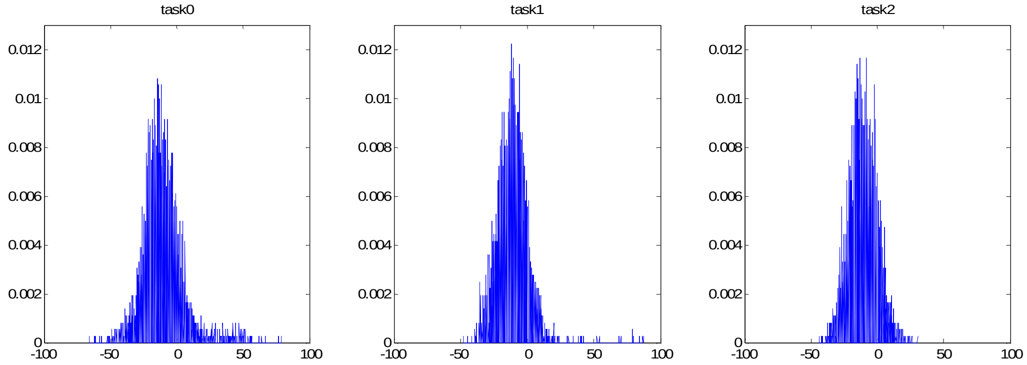


Figure 3.4 The Amplitude distributions of raw signal from A01 for task 0, 1 and 2

3.1.4 Error Measurements

For the most common way to measure the classification error, error=*n*/*m*, where *n* is the number of parts classified inaccurately, and *m* is the number of total number of parts in testing sets, it cannot provide us a general perspective on how well an algorithm performs due to the special condition of the data set. For instance, a testing set has been divided into 81 parts, only 17 parts correspond to task 1 and the rest represent task 0. Even an algorithm classified all the parts as task 0, the error only equals to 20.988%. Therefore, we apply another type of error measure: $error = n_i / m_i$, n_i is the number of sub-task *i* misclassified, m_i is the total number of sub-task *i*. For the same instance above, the errors become error1=20.988% and error2=100%. Therefore, the second measurement can provide more details about how well we classify the data set than the first approach.

3.2 Linear Programming Method

3.2.1 Algorithm for linear programming Discrimination Method

Based on the characteristic in the frequency domain, a linear programming method is applied to classify the data. The algorithm is shown as below, the input parameters include: W_1, W_2 , F_1, and F_2. "Training" and "classifying" are two consecutive phases in this algorithm. Moreover, DFT is only applied for one-dimensional decomposition, for a multiple dimensional data set, the data in each dimension could be transformed separately.

"Training Stage"

1. In the original training data, use W_1 as the length of sub-tasks, separate the known tasks in the data set and split every identified task into equal long sub-tasks.
2. Split every sub-task into W_2 length and then apply DFT on every W_2-long data points.
3. Compute the average magnitudes over F_1 to F_2 from the outputs; calculate the average of (W_1/W_2) values; it is the "characteristic value" of this sub-task. Therefore, each sub-task has its own characteristic value.
4. Apply the linear optimization method (2.6) , which tries to find an optimum hyperplane that can separate different sub-tasks most correctly based on the characteristic values. The optimum solution of this linear programming problem includes a vector ω and a scalar *r*, which are the direction and location of this hyperplane. The dimension of this hyperplane depends on the dimension of the characteristic value, which depends on the dimension chosen from the data set.

"Classifying Stage"

5. Divide the testing data into equal W_1 long consecutive segments in the time domain. These segments are the unknown sub-tasks needed to be classified in the following steps. It is possible that one unknown time period is contains by two different sub-tasks. Then, just discard this kind of segments and don't classify them.
6. Compute the characteristic values of all segments as the same way in the studying stage using the same parameters, W_1, W_2 , F_1, and F_2.
7. Use the optimum hyperplane in learning stage to classify these unknown sub-tasks.

Here only one hyperplane is applied to classify two conditions, non-cognitive workload and cognitive workload. If classifying more conditions or higher classify accuracy is needed, we could apply the greedy linear-programming-based algorithm MSMT (Multi Surface Method Tree) and MSM (Multi Surface Method) (Bennett, Mangasarian, 1992). Both these two methods are based on solving the linear programming problem formulated above. For MSMT, the idea is to continue bi-split data sets until all sets contain only one kind of data points or to some desired percentage so there are at most 2^i LP's in every iteration. In comparison with MSMT, the advantage of MSM is that only a single linear program needs to be solved at each step because when a linear surface obtained, points classified correctly are discarded by the surface and formulate linear programming problem based on the rest of data. So at the end, a piecewise-linear surface is generated.

3.2.2 Computational Procedure

For each subject we have two trials, thus three subjects and six trials in all (A01, A02, E01, E02, F01, and F02). For a subject, one trial is used as training set and the other trial is used as testing set, then switch the training and testing set. Note, because individual has his/her own "brain characteristics", it haven't found any promising algorithms that are able to use the data from one subject to classify the data for another subject.

For the features selection, four combinations are employed: (Fz, F7), (HEOG, VEOG), (Fz, HEOG), (Fz, F7, HEOG, VEOG). For the parameters in the algorithm, we calculate the average magnitude from 1 to 10 Hz, so F_1 equals 1 and F_2 equals to 10. For W_1 and W_2, multiple values are chosen in order to find a pair of optimal values. The values selected in the computational experiment include: [4000, 20/50/100/4000], [3000, 20/50/100/3000], [2000, 20/50/100/2000], [1000, 20/50/100/1000]. The value before the comma is the value for parameter W_1, and the values after the comma are the values for W_2. For instance, in [4000, 20/50/100/4000], W_1 equals to 4000, and W_2 equals to 20, 50, 100, and 4000. Therefore, in each brackets, there are four pairs of values for the W_1 and W_2.

3.2.3 Computational Results

The results are presented in the Table 3.3. We can see that for the parameters W_1 =2000 and W_2=20, the average error rate for both No Cognitive Load (NCL) tasks and Cognitive Load (CL) tasks are the optimum among all the parameters. By applying DFT and linear programming discrimination method, we can classify the NCL tasks and CL tasks at the average accuracy of about 14.8% and 20.6%. Furthermore, from the Table 3.4, using Fz and F7 signals provide us not only a better average accuracy but also smaller variance for the different values of the parameters (W_1, W_2).

W_1	4000				3000			
W_2	20	50	100	4000	20	50	100	3000
CL error	0.4974	0.3001	0.304	0.2968	0.2334	0.2633	0.249	0.2702
NCL error	0.319	0.2947	0.2991	0.2508	0.2508	0.2539	0.2355	0.2702
W_1	2000				1000			
W_2	20	50	100	2000	20	50	100	1000
CL error	0.1476	0.2256	0.2104	0.2759	0.1929	0.2891	0.275	0.264
NCL error	0.2057	0.2196	0.2026	0.2541	0.2699	0.2872	0.2712	0.2435

Table 3.3 The test results of linear separation algorithm for EEG signals Fz, F7 with different values of W_1, W_2

Signals	Fz, F7	HEOG, VEOG	HEOG, Fz	Fz,F7, HEOG,VEGOG
AVE/STDV CL error	0.147625 /0.057568	0.145733 /0.07094	0.164567 /0.178737	0.154575 /0.07228
AVE/STDV NCL error	0.2056917 /0.09144	0.21505 /0.117397	0.220658 /0.217710	0.219200 /0.116993

Table 3.4 Average and standard deviation of classification error for using different combinations of features

3.3 Principal Component Analysis Method

Principal Component Analysis (PCA) Method is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences (Smith, 2002), which transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components. In principal components analysis, we attempt to explain the total variability of *n* correlated variables through the use of *n* orthogonal principal components. The components themselves are merely weighted linear combinations of the original variables. Each principal component is a linear combination of the original variables. Moreover, each principal component is a single axis in the space of the data, and as a whole they form an orthogonal basis for the space. The first principal component can be expressed as:

$$\begin{aligned} X_1 &= a_{11}M_1 + a_{21}M_2 + \ldots + a_{n1}M_n \\ or\ X_1 &= \mathbf{a_1}^{\mathbf{T}}\mathbf{M} \end{aligned} \tag{3.2}$$

$\mathbf{a}_1$ is the first principal component coefficients, **M** is a variable in *n* dimensions, X_1 accounts for the maximum variability of the *p* variables of any linear combination. In other words, projecting the data onto the first principal component, the projection data has the maximal variance among the entire axis in the space. And the second principal component X_2 is formed such that its variance is the maximum amount of the remaining variance and that it is orthogonal to the first principal component. That is $\mathbf{a}'_1\mathbf{a}_2 = 1$. For a variable $\mathbf{M} \in \mathbf{R}^p$, we have

$$\begin{aligned} & X_i = \mathbf{a}_i^{\mathbf{T}}\mathbf{M} \\ & \mathbf{a}_j{}'\mathbf{a}_i = 0,\ \mathbf{for}\ j = 1,\ \ldots,\ i-1 \\ & i = 1,\ \ldots,\ p \end{aligned} \tag{3.3}$$

Here, $\mathbf{a}_i$ is called the principal component coefficients.

For a matrix **M,** each column represents a variable, and each row comes from the observations of these variables. And if $\mathbf{M}^{\mathbf{T}}$ has a zero empirical mean, then the PCA transformation is given by

$$\begin{aligned}\mathbf{X}^{\mathbf{T}} &= \mathbf{M}^{\mathbf{T}}\mathbf{W} \\ &= \mathbf{V\Sigma}\end{aligned} \tag{3.4}$$

Where $\mathbf{V\Sigma W}^{\mathrm{T}}$ is the singular value decomposition of $\mathbf{X}^{\mathrm{T}}$.

Similar to the procedure in the linear programming algorithm, to plot Figure 3.6, each trial is divided into several tasks according to the prior information, and each task is split into equal long sub-tasks. Then, DFT is utilized to transform each sub-tasks from time domain into frequency domain and calculate the average magnitudes over 1Hz to 10 Hz on signal Fz and F7 for these sub tasks. So each sub task has a two dimensional value, the value for the whole task equals to the average value of all the sub tasks belong to itself.

The length of sub-task is 100 data points or in other words half seconds, which is relatively small comparing to duration of a whole task (usually having more than 4000 points). The dots, the stars and the triangles in the plots represent the value calculated for every task. The blue dots represent task 0. The green stars represent task 1 and the red triangle indicates the task 2. The advantage of this approach over the previous one is more information could be extract from a task. And compare to the method we earlier, a “better” plots is achieved (different kinds of tasks are more separable).

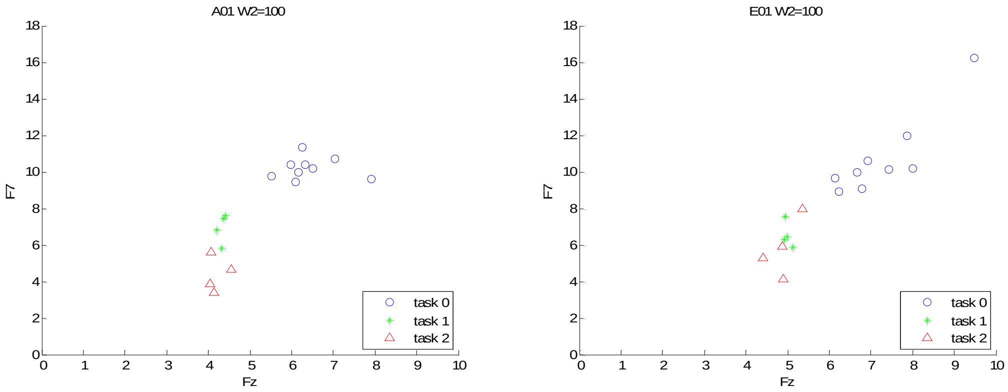


Figure 3.5 The magnitudes of task 0, 1, 2 of Subject A and E in Fz-F7 spaces

In Figure 3.5, NCL tasks and CL tasks are separable so the optimum value of objective function always equals to zero and the LP formulations (2.6) has optimal solutions. However, in Figure 3.6, it is obvious the linear separator varies dramatically from one trial to the other trial for the same subject. This trait is undesirable for the classification and contradict to the assumption that one subject should have similar characteristics in different trials. As a result, another attribute of a “good” separator is “robustness”, which loosely defined as the stability between the different trials for the same subject.

Applying an algorithm involving PCA, a more “robust” separating algorithm can be acquired. In the algorithm, PCA is used to obtain the first principal component coefficient (it is also the direction vector of the axis which is vertical to the first principal axis in the). Then, set the first principal component coefficient as the value of **w** in the LP formulation (2.6); therefore, **w** is fixed before solving the LP, and only the optimum value of **r** need to compute. In other words, we utilize the first principal component as the direction of the separating hyperplane and solve the LP to find a location of the hyperplane at where this hyperplane can separate all data points in the space. In Figure 3.6, separating planes for subject E are generated by this new method and the LP method.

We can conclude by visual observation that the new approach is more “robust”. All of the three subjects have similar results.

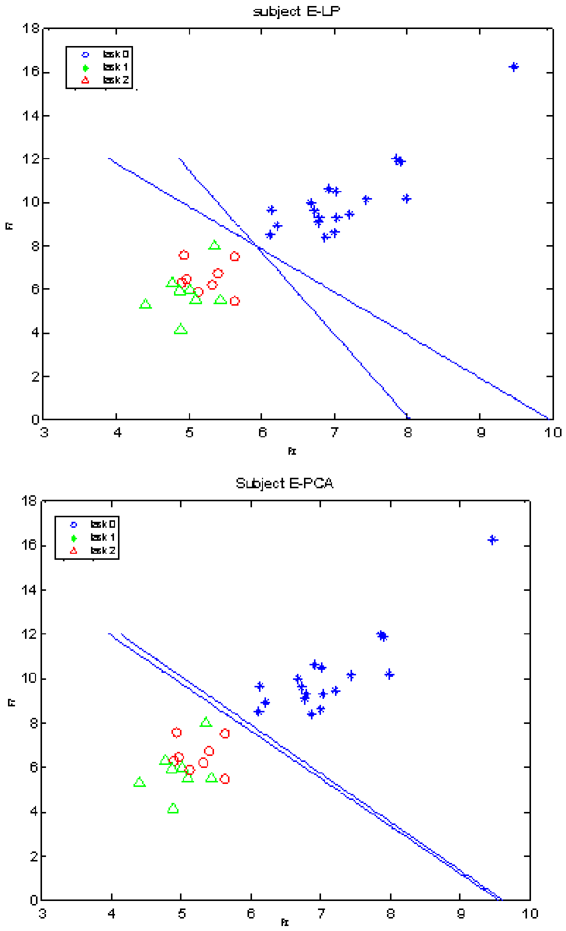


Figure 3.6 Comparison between the linear separator obtained by LP and PCA for two trials of subject E

3.4 K-Nearest-Neighbor Method

3.4.1 Introduction

Linear discriminant function method is one of parametric techniques which are based on the assumption that the underlying discriminant functions are known, with several parameters need to adjust. However, this assumption is suspect in many situations, and some nonparametric procedures can be used without this assumption.

There are several types of nonparametric methods in pattern recognition, among which is k nearest-neighbor rule. General speaking, the rule of this method is simple: find the “nearest k known points” of a certain test points and classify the test point into the category in which “most” of the nearest points are.

A general procedure for K-nearest neighbor method:

Training:

Build the set of training examples D

Classification:

Given a query instance x_i to be classified,

Let $kNN = \{x_1 \ldots x_k\}$ denote the *k* instances from *D* that are nearest to x_i

Then

$$y(x_i) = \arg\max_{p} \sum_{x_j \in kNN} f(x_j, c_p) \qquad (3.5)$$

Where $f(x_j, c_p) \in \{0,1\}$ indicates whether x_j belongs to class c_p, *p* is the number of classes in a data set. When *k*=1, then k-nearest-neighbor method becomes nearest-neighbor method, and in 2-dimensional space, the decision surface is a Voronoi diagram.

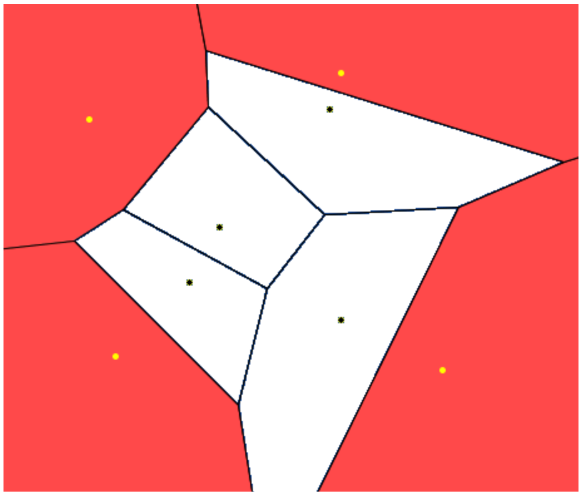


Figure 3.7 A Voronoi diagram

3.4.2 The Algorithm and Application

In the data set, although the training data points are usually linear separable, the testing data points in the space are generally linear inseparable. But the linear hyperplane can still separate most of the NCL and CL tasks. Meanwhile, for low cognitive and high cognitive load tasks, the linear separator approach doesn’t perform well because of the nonlinearity. Therefore, k-nearest neighbor method may be a good choice for a nonlinear separator. In the implementation of k-nearest neighbor method, the classification is also based on individual subject.

The algorithm using k-nearest neighbor method is similar to the algorithm using linear optimization method; the only difference is that after the characteristic values of training data are obtained, we applied k-nearest neighbor method instead of solving linear programming problem to classify the testing data.

There are details in our research need to be specified here. First, we apply “Euclidean norm” as the measure of distance in this method. Second, k-nearest neighbor rule generally doesn’t consider the number of sample size, however in our algorithm, we not only consider the “k nearest points’’ but also the number of training data where these points belong to. Therefore, after the k nearest training points of one test point are located, we count the number of training points in each class, then the numbers are divided by the size of each class. Here the number of data points that each class has in the training set, has an effect on the decision of k-nearest neighbor rule. The more data one class has, the less effect a point in this class has on its neighbors. This method should offer better result than the original one under the condition that unbalanced information on different classes is given. For instance, in one of our research data set, because of the design of experiment, the researchers collected more data in one class than the data in the other two classes. Since this biased information was made by man-induced factor and it is not part of the characteristics of the data set itself, without eliminating this factor, it would deteriorate the classification results

The procedure for building training set and testing set:

For training data set:

1. For the original training data, identify the known tasks in the data set.
2. Split every known task into W_2 length sub-tasks and then apply DFT on every W_2 data points.
3. Compute the average magnitudes over F_1 to F_2on the outputs in step 2; calculate the average values of all the sub-tasks in each task; it is the “characteristic value” of this task. It is also the coordinates of a training data point in a multi-dimensional space.

For testing data set, the procedure is similar with a little difference:

1. Divide the testing data into equal W_1 long consecutive segments in the time domain. These segments are the unknown sub-tasks needed to be classified in the following steps.
2. Split every sub-task into W_2 long span and then apply DFT on every W_2 data points.

3. Compute the average magnitudes over F_1 to $F_2$from the outputs; calculate the average of total (W_1/W_2) values; it is the "characteristic value" of this sub-task, also the coordinates of a testing point.

   The k-nearest-neighbor algorithm applied in our research:

1. Select a testing point
2. Calculate the distances from this testing point to every training points (Euclidean Distance applied)
3. Sort the distances from smallest to the biggest
4. Find the training points corresponding to the first k distances, Count the classes of these training points
5. The testing point is classified as $j = Max\{\frac{n_i}{N_i}, i = 1, ..., c\}$, j is the class that the testing point assigned to, c the total classes the training data has, n_i is the number of the training points belong to class i among the k- nearest points, N_i is the number of the training points belong to class i among all the training points.

Also, four combinations of signals are tested in this method: 1. all eight features. 2. Fz and F7 of EEGs. 3. ECG, Pz, T5, and O2 of EEGs. 4. VEOG, HEOG, and EEGs on Fz, F7. First all signals are transformed to the frequency domain. Then the average magnitudes from 1 Hz to 10 Hz are computed. *k* equals to 1 is applied. As it is mentioned, there are six combinations for training and testing data set, A01/A02, E01/E02, F01/F02, A02/A01, E02/E01, F02/F01, the dataset before the slash is the testing set, the one after the slash is the training set.

Furthermore, we select different values for W_2, 3000, 2000, and 1000 which corresponds to 15s, 10s, and 5s time periods. The training error and testing error are obtained after each run. The error measurement described in 3.1.4 is applied. Since three classes are available in the data set, the error measurement is a 3×3 matrix, the diagonal are the accuracy percentage for task 0, 1and 2. The off diagonal x_{ij} elements indicate how many percentages data in class *i* are misclassified as class *j*. The results for 3 combinations are listed in Table 3.5, the rest can be found in the appendix.

Fz, F7		A02/A01			E02/E01			F02/F01		
Test Error	15s	0.714	0.262	0.024	0.698	0.116	0.186	0.667	0.222	0.111
		0.167	0.5	0.333	0.167	0.5	0.333	0.4	0.4	0.2
		0.2	0.4	0.4	0.167	0.5	0.333	0	0	1
	10s	0.651	0.286	0.064	0.641	0.219	0.141	0.609	0.219	0.172
		0	0.778	0.222	0.222	0.333	0.444	0.222	0.111	0.667
		0	0.625	0.375	0.1	0.2	0.7	0	0.25	0.75
	5s	0.633	0.258	0.109	0.638	0.213	0.15	0.577	0.162	0.262
		0	0.563	0.438	0.25	0.25	0.5	0.188	0.375	0.438
		0.063	0.25	0.688	0.158	0.263	0.579	0.059	0	0.941
Train Error	15s	1	0	0	1	0	0	0.889	0.111	0
		0	0.75	0.25	0	0.5	0.5	0.25	0.75	0
		0	0.25	0.75	0	0.5	0.5	0	0.25	0.75
	10s	1	0	0	1	0	0	0.889	0.111	0
		0	0.75	0.25	0	0.5	0.5	0.25	0.75	0
		0	0.25	0.75	0	0.5	0.5	0	0.25	0.75
	5s	1	0	0	1	0	0	0.889	0.111	0
		0	0.75	0.25	0	0.5	0.5	0.25	0.75	0
		0	0.25	0.75	0	0.5	0.5	0	0.25	0.75

Table 3.5 Testing and training errors for k-nearest neighbor method with different time windows

We also calculate out the mean and standard deviation for accuracy over all the different combinations in Table 3.6. The results show that k-nearest neighbor method has a lower accuracy in separation of tasks (0) and tasks (1, 2) than the linear separation method. Moreover, the best average classification accuracy is from using all the features in the data set and with $W_1 = 3000$. The longer the time periods is, the better the classification results is. This is because that the larger W_1 is, the more data points one time window has, and the more accurate information about the cognitive status we can extract. Nevertheless, the disadvantage of a larger W_1 is that we would not be able to identify a subject’s state in a short time scale and make it unpractical in real-time data mining. Applying all the signals gives us the best result among all the four combinations’. Also, the standard deviations are around 0.2 to 0.3, which may due to the individual characteristics of each subject.

	All features		Fz,F7		ECG,Pz,T5,O2		VEOG,HEOG,Fz,F7	
	Mean	STDEV	Mean	STDEV	Mean	STDEV	Mean	STDEV
15s	0.6797	0.3118	0.6594	0.3026	0.6733	0.3110	0.6732	0.3104
	0.4000	0.2143	0.3714	0.2870	0.4000	0.2143	0.3191	0.2707
	0.5946	0.3157	0.4946	0.3247	0.5898	0.3385	0.5898	0.3385
10s	0.6204	0.2922	0.6270	0.2958	0.6071	0.2912	0.6182	0.2900
	0.4286	0.2755	0.3413	0.2836	0.4306	0.2827	0.4127	0.2921
	0.5119	0.2931	0.5492	0.2986	0.4940	0.2797	0.5119	0.2931
5s	0.5476	0.2512	0.5697	0.2618	0.5487	0.2541	0.5442	0.2492
	0.3680	0.2261	0.2411	0.1961	0.3666	0.2157	0.3257	0.1809
	0.5668	0.2725	0.6127	0.2935	0.5668	0.2725	0.5743	0.2753

Table 3.6 Mean and standard deviation of testing accuracy for nearest-neighbor method with different W_1 and different signals when $W_2 = 100$

Another similar method called Nearest-centroid algorithm:

1. Compute the centroids for all the classes in the training set: $\mathbf{c}_i = \frac{\mathbf{a}_{1i} + \mathbf{a}_{2i} + ... + \mathbf{a}_{ni}}{n}$, $\mathbf{c}_i$ is the centroid of class *i*, $\mathbf{a}_{ji}$ is a training point belong to class *i*, and *n* is the total number of training points in class *i*.
2. Select a testing point; calculate the distances from this testing point to the centroid of each class (Euclidean Distance applied).
3. The testing point is assigned to the class, which its nearest centroid belong to.

This algorithm is a combination of k-mean method and nearest-neighbor rule: first within training data, the centroid of each the class is found, and then testing data is classified by applying these centroids and nearest neighbor method. The same parameters and combinations of signals in k-nearest neighbor method are used in the nearest-centroid method. The computational results are shown in Table 3.7. The k-nearest neighbor method outperforms nearest-centroid method in all scenarios. Remark, using the combination of ECG, Pz, T5 and O2 we obtain the same results as applying all the features. It may be explained by that some of these four signals have higher magnitudes in frequency domain and overshadow the other signals in the classification procedures.

	All features		Fz,F7		ECG,Pz,T5,O2		VEOG,HEOG,Fz,F7	
	Mean	STDEV	Mean	STDEV	Mean	STDEV	Mean	STDEV
15s	0.5442	0.2804	0.6262	0.2959	0.5442	0.2804	0.5810	0.2813
	0.4000	0.3151	0.4619	0.2542	0.4000	0.3151	0.4000	0.3372
	0.5435	0.2574	0.4946	0.3035	0.5435	0.2574	0.5660	0.3028
10s	0.5534	0.2876	0.5957	0.2967	0.5534	0.2876	0.5692	0.2820
	0.3294	0.2118	0.3651	0.2681	0.3294	0.2118	0.3333	0.2546
	0.4560	0.2198	0.4849	0.2467	0.4560	0.2198	0.5401	0.2685
5s	0.4982	0.2472	0.5411	0.2488	0.4982	0.2472	0.5098	0.2390
	0.3403	0.2113	0.3124	0.1677	0.3403	0.2113	0.3481	0.2106
	0.4288	0.2823	0.6189	0.3023	0.4288	0.2823	0.5627	0.2761

Table 3.7 Mean and standard deviation of testing accuracy for nearest-centroid method with different W_1 and different signals when $W_2 = 100$

3.5 Feedforward Neural Network

3.5.1 Multi-layer feed forward Neural Networks

Artificial neural networks method is a branch of artificial intelligence, which is inspired by biological nervous systems. The similarities between biological nervous systems and artificial neural networks are that they are both composed of simple elements operating in parallel, and if the number of these simple elements reaches certain point, they can perform complex functions. This simple element is called neuron both in biological and artificial networks. However in artificial intelligence area, a “neuron” actually means a parameterized function. The variables of the function are often called inputs of the neuron and its value is its output (Dreyfus, 2005).

A neural network is a composition of several neurons, and a feed forward neural network can be demonstrated by the graph in Figure 3.1, where the edges are the connections and the vertices are neurons. Information always goes forward and never goes back. In addition, in a multi-layer network, there are usually more than three layers of neurons and they connect with each other one by one, from the input layer to output layer. Note, the “neurons” in the input layer simply provide the input data to hidden layer instead of processing the data, which is the reason, in some books and software the input units are not referred as “input neurons”, and for hidden layer, the neurons are usually sigmoid parameterized functions, while the linear functions are always applied in output layer.

There are values for all the connections between each neuron, which called weights. A neural network can be trained to perform a particular function by adjusting these weights. Proved by Kolmogorov but refined by others, any continuous function from input to output can be implemented in a three-layer net, give sufficient number of hidden units, proper nonlinearities, and weights (Duda, Hart, Stork, 2001). Therefore, theoretically a three-layer network with enough hidden neurons should be able to approximate any functions, but in practice, researchers also utilize networks more than one hidden layers due to the efficiency reason (with fewer total units).

As the rapid developments in computer technology, artificial neural network becomes more and more popular. A great amount of effort is spent on the development of neural networks for applications such as pattern recognition and modeling, data compression, optimization, etc. The advantages of neural network over conventional methods rely on its ability in solving problems which are not well understood or the solution is too complex to be found, but it doesn't provide any insight information about the problem.

A neural network performs in two different modes: learning (or training) and testing. In supervised classification, before a neural network is applied, training is necessary for an optimal result. A common training procedure is demonstrated in Figure 3.9. To be precise, "training" is defined as finding the appropriate weights between each connection so $f(output, target) \le \delta$. f is the performance function, and δ is the threshold chosen according to different situations.

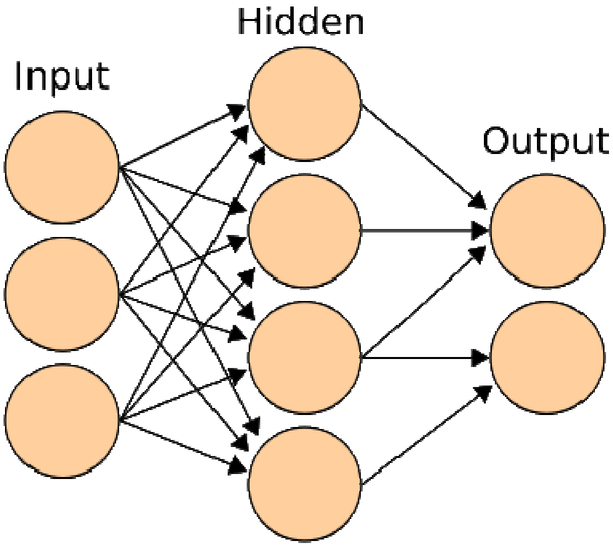


Figure 3.8 A typical three layers feed forward neural network

Before training a network, the number of layers, the number of neurons for each layer, the transfer functions for each layer, the training algorithm, and the performance functions, should be decided. At the beginning of the training process, a training set is selected, the network predict the target value for each training data based on initialized weights. However, as training goes on, the network adjusts internally by a certain training algorithm until it reaches a stable stage at which the outputs are considered satisfactory: $f(output, t\arg et) > \delta$. After that, all weights in the network should be fixed and stop training, go to the next stage: testing. In summary, learning is an adaptive process during which the weights change in order to offer the best response to all the observed stimuli. In the testing stage, the trained network is used to classify new, previously unseen inputs. At this stage, the network receives an input signal and processes it to generate an output.

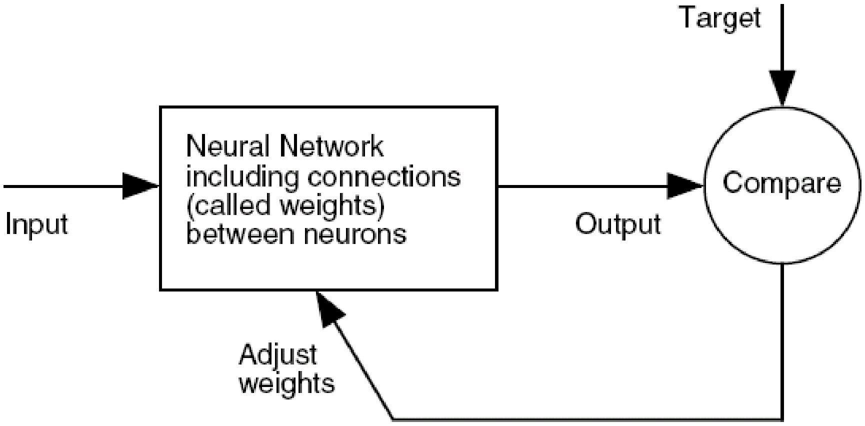

Target
Neural Network including connections (called weights) between neurons
Input
Output
Compare
Adjust weights

Figure 3.9 A procedure to train a neural network

3.5.2 Application

Previously research is based on the assumption that for the same subject, all the trials are related, so based on the information in one trial, one should be able to classify the tasks in another trial for the same subject. Nevertheless, it is possible that every trial has its distinctive characteristics while they are related with each other at the same time. More important, this unique characteristic may interfere with the performance of our algorithm. In order to extinguish this factor as much as possible, a potential way is to treat each trial as an independent trial and randomly select a fraction of the data as training data and the rest as testing data.

Two different algorithms are implemented by the Neural Network Toolbox (NNT) in the Matlab (The Mathworks, Inc), an all-purpose neural network environment. In the first algorithm, a three-layer neural network with three neurons in the output layer is applied since we have three classes in the dataset. Task 0, 1 and 2 correspond to target value [1, 0, 0], [0, 1, 0], [0, 0, 1], respectively. The number of inputs is related to the number of features we extract from the multidimensional time-series data set, and 20 neurons are in the hidden layer. All the information for the network is listed in Table 3.8.

For the second approach, two networks are utilized instead of one. The first network separate NCL tasks (0) from CL tasks (1 and 2), and the second network identify the low cognitive tasks (1) and the high cognitive tasks (2). Both networks have three layers with the same transfer functions, training algorithm and learning function as in Table 3.8. The differences between these two networks are: first, the output values have different meanings. The target value for NCL tasks is 0, for CL tasks is 1, while in the second network, low and high cognitive tasks correspond to target value 0 and 1. In addition, the first network has 5 neurons and the second has 20. The reason different numbers of neurons are chosen for the two networks, is not only based on experiment, but also based on the observation of the linearity between non-cognitive and cognitive tasks and the nonlinearity between low and high cognitive tasks.

Number of inputs	35, 28, 7
Number of outputs	3
Number of hidden neurons	20
Transfer function for hidden layer	Hyperbolic Tangent Sigmoid
Transfer function for output layer	Linear
Training algorithm	Levenberg-Marquardt back propagation
Performance function	Mean square error

Table 3.8 Details for the neural network

Note, different numbers of inputs are tested for the same data set in order to discover the combination of features which can give the best classification result. In Wilson and Russell's paper (Wilson and Russell, 2007), they calculated the average magnitudes over 5 bands from every EEG and EOG signal after the data was transferred from time domain to frequency domain by DFT: delta (2.0 to 40 Hz), theta (5.0 to 8.0 Hz), alpha (9.0 to 13.0 Hz), beta (14.0 to 32.0 Hz), and gamma (33.0 to 43.0 Hz). Include the 5 bands mentioned, 4 bands and 1 band are also tried. For 4 bands, they are 1 to 10 Hz, 11 to 20 Hz, 21 to 30 Hz, 31 to 40 Hz, and for 1 band, it is from 1 to 10 Hz. The 5 bands, 4 bands and 1band correspond to 35 inputs, 28 inputs and 7 inputs for the neural network since VEOG, HEOG, Fz, F7, Pz, T5 and O2 are chosen.

Each trial is treated as an individual data set. Since the tasks are performed continually during a certain time period, classifying every time point in the data set is unpractical. Hence, every single trial is divided into equal long consecutive segments along the time axis. Randomly 50% of the data is selected as training set, 25% of the data as cross validation set, and the other 25% as testing set. Furthermore, the length of task 0 is at least 6 times longer than the time of the task 1 and 2 in each trial. If data is arbitrarily selected as training set and testing set over the whole data, this unbalanced situation would be kept in the training and testing set. And after the training, the neural network would be biased: more sensitive to task 0. To avoid this situation, the same percentage of data is selected from each task, and the data in task 2 and 3 is replicated to make they have the same amount data as in task 1. A five-second long segment is used as input signal to the network. And for each trial, we run 10 times and compute mean and standard deviation of accuracy for each task. All the results can be found in the appendix, only the results for 35 inputs are listed in Table 3.9.

In Table 3.10, the accuracies of the algorithm 1 and algorithm 2 are comparable. The average accuracy for algorithm 1 among different trials is 0.85053, 0.47083, 0.59443 for task 0, task 1, and task 2, comparing to 0.85862, 0.375, 0.61667 from algorithm 2. The neural network with 35 inputs gives the similar results in algorithm 1 and 2, and task 0 is easier to identify, again. For 7 inputs network, algorithm 1 gives the worst accuracy, but algorithm 2 still have a similar or even better accuracy compare to the results from the network with 35 and 28 inputs. Base on these observations, conclusion is that using 35 inputs in algorithm 1 has no advantages over using 28 inputs. Additional, applying 7 inputs can achieve a similar result by algorithm 2, although algorithm 1 with 28 inputs has the best result over all.

Algor1	A01		E01		F01	
35 input	AVE	STDV	AVE	STDV	AVE	STDV
Task 0	0.8871	0.06842	0.7806	0.14489	0.8323	0.08439
Task 1	0.425	0.28988	0.55	0.28382	0.5	0.26352
Task 2	0.6	0.26296	0.7	0.24596	0.6333	0.33148
Algor1	A02		E02		F02	
35 input	AVE	STDV	AVE	STDV	AVE	STDV
Task 0	0.8581	0.08216	0.8903	0.04612	0.8548	0.05938
Task 1	0.45	0.2582	0.55	0.30732	0.35	0.33747
Task 2	0.5333	0.35833	0.4333	0.27444	0.6667	0.35137

Algorithm 1

Algor2	A01		E01		F01	
35 input	AVE	STDV	AVE	STDV	AVE	STDV
Task 0	0.8387	0.12814	0.8161	0.21183	0.8968	0.06933
Task 1	0.375	0.35843	0.275	0.24861	0.35	0.33747
Task 2	0.575	0.26484	0.725	0.24861	0.8	0.2582
Algor2	A02		E02		F02	
35 input	AVE	STDV	AVE	STDV	AVE	STDV
Task 0	0.871	0.11069	0.8194	0.09274	0.9097	0.0884
Task 1	0.4	0.35746	0.525	0.24861	0.325	0.26484
Task 2	0.525	0.27513	0.55	0.32914	0.525	0.18447

Algorithm 2

Table 3.9 The result of neural network with 35 inputs for (a) algorithm 1 and (b) algorithm 2

		35	28	7
Algor1	Task 0	0.85053	0.82902	0.73978
	Task 1	0.47083	0.5125	0.26667
	Task 2	0.59443	0.60002	0.47778
Algor2	Task 0	0.85862	0.87312	0.86988
	Task 1	0.375	0.42083	0.425
	Task 2	0.61667	0.59167	0.6

Table 3.10 The average classification accuracy for Algorithm 1 and 2 with different numbers of inputs

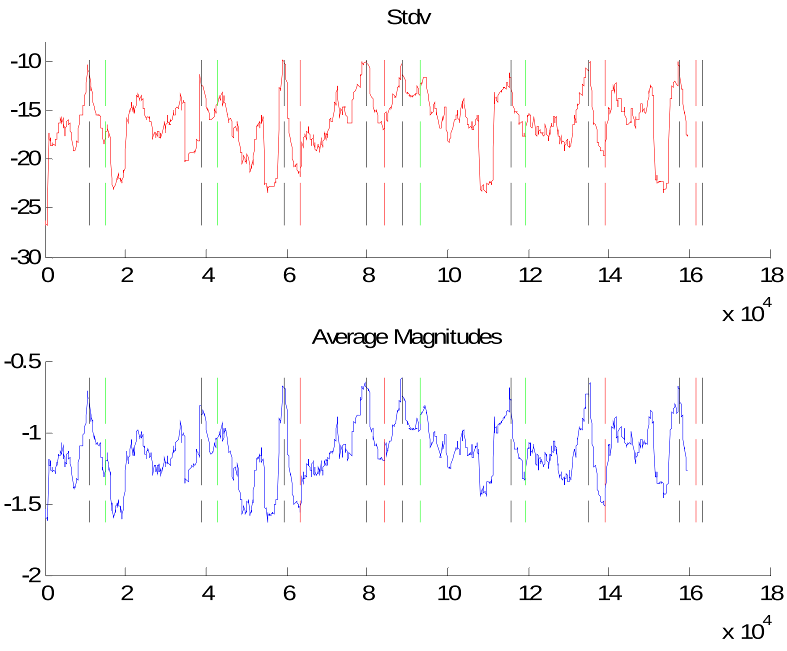
3.6 Peak Detection Method

The Peak Detection Method (PDM) is based on the assumption that the brain's state of a subject at a certain time point can be determined by the time period before and after this point. In this algorithm, a fixed-length window slides through the time axis and certain value are measured every time the window is located. Also, the value calculated is assigned to the time point at the middle of the window. For this algorithm, these parameters: w_1 the length of the window, and w_2 the span that every time the window slides, need to be pre-determined. There are two ways to gauge the window: the magnitude for a frequency band after DFT or the standard deviations for all the data in the window since interesting patterns are found in the previous section. Both of these two measures are applied on a single feature which also needs to choose prior.

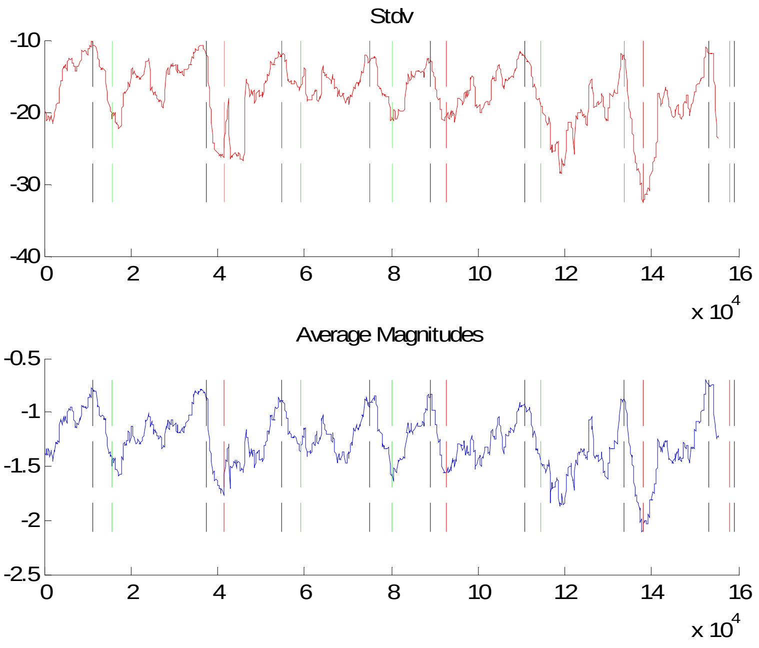
Figure 3.10 is generated by the PDM, the X axis is the time axis; the Y axis represents values of a certain measurement. The vertical lines indicate the finishing of events: the black lines mean task 0 ends, the green indicates the task 0 and the red indicates the task 2. Therefore, a task 2 happens between a black line and a red line, and a task 1 happens between a black line and a green line. The parameters chosen here are

$w_1 = 18\sec$ and $w_2 = 1\sec$, which imply the length of this moving window is 18 seconds and this window moves 1 seconds every time. Furthermore, the feature chosen from the data set is EEG signal on Fz spot and the frequency band is from 1 Hz to 10 Hz.

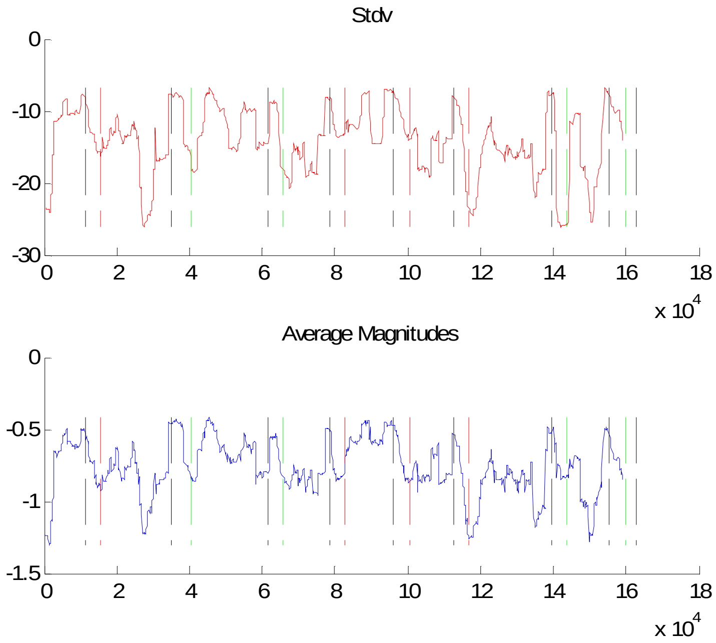
In Figure 3.10, for the same trial and the same subject, there are two plots, the plot above applies measure of the standard deviations of the raw data and the plot below calculates the average magnitude for the transformed EEG signal. Also, In Figure 3.10, at the beginning of every task 1 and task 2, a peak appears in both two measurements. It indicates that when a subject is performing under a CL task, the EEG data collected from Fz spot always has a lower average magnitude in frequency domain and a lower standard deviation in the time domain, than under a NCL task. Note, the negative values of both two measurements are used in Figure 3.10 just for the convenience of observation. Another interesting phenomenon is that these two measurements are closely correlated; the correlation coefficients are in


Stdv
-10
-15
-20
-25
-30
0
2
4
6
8
10
12
14
16
18
x 10^4
Average Magnitudes
-0.5
-1
-1.5
-2
0
2
4
6
8
10
12
14
16
18
x 10^4

A01


Stdv
-10
-20
-30
-40
0
2
4
6
8
10
12
14
16
x 10^4
Average Magnitudes
-0.5
-1
-1.5
-2
-2.5
0
2
4
6
8
10
12
14
16
x 10^4

E01



F01

Figure 3.10 The two measurements versus time plots from PDM for data set A01, E01, F01

Trail	A01	A02	E01	E02	F01	F02
Correlation	0.9488	0.9392	0.9266	0.8927	0.8848	0.7498

Table 3.11 The correlation between the average magnitudes and standard deviation in the Peak Detection Method

A heuristic peak detection algorithm is applied to discover the change during the trial:

Initial $y_{\max} = -\inf$, $max = 0$ with predefined δ, γ

For *i*=1…*n*, *n* is the number of data points along the time axis.

If $y_i > y_{\max}$, then $y_{\max} = y_i$ and $max = i$

Else if $y_{\max} > y_i + \delta$ and $y_{\max} > \gamma$ then Point ($x_{\max}, y_{\max}$) is a peak.

End

δ is the relative threshold to identify a peak and γ is the absolute threshold. Only a point satisfies these two at the same time would be labeled as a peak. In our computational experiment, δ equals to one standard deviation of all the points in the plot, and γ equals to the mean plus one standard deviation.

The result from the algorithm applied on the previous "standard deviation plot" of E01 is shown in Figure 3.11.In this figure, most of the peaks detected are the transitional point of different tasks, although the peaks are not all on the exact time point of task 0 shifting into task 1 or 2,, they are close to the points in a satisfied precision. The results for each data set are listed in Table 3.12. The table includes the number of transition points it has for each trial, how many of them are detected, and how many false alarms have been triggered. Note, for PDM, task 0 is considered as a baseline or benchmark, and this method cannot differentiate task 1 and task 2 now.

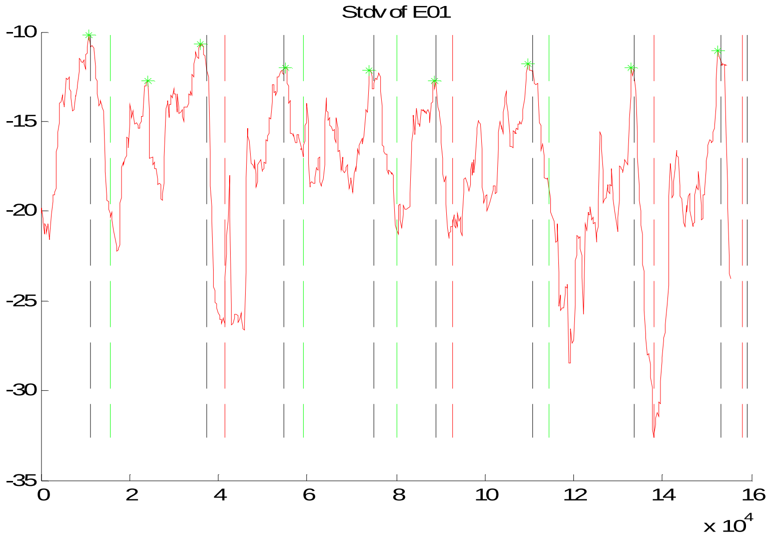


Figure 3.11The points detected by the peak detection algorithm in E01 data set

Table 3.13 shows that the percentage of task 1 detected by PDM is comparable to the percentage of task 2: average 87.50% and 83.33% respectively. The average total accuracy is 72.16%. The reason that the total accuracy is lower than both of the percentage of task 1 detected and task 2 detected is there are several false alarms during each trails. Therefore, to increase the total accuracy, we need to lower the false alarm rate.

Data Set	Transitional points	Correctly detected:	Task1 detected:	Tasks2 detected:	False alarm
A01	8	8	4	4	3
A02	8	7	3	4	1
E01	8	8	4	4	1
E02	8	6	3	3	3
F01	8	8	4	4	3
F02	8	4	3	1	1

Table 3.12 Detail results from PDM for each data set

After all, PMD is still in a preliminary stage and has its own merits and drawbacks. The method only adopts one signal out of eight so it is straightforward to apply but may miss useful information in the rest signals. Although it doesn’t need to be trained, several parameters need to be selected based on either the prior-experience or certain information of the data set.

	Accuracy			False Alarm Rate
	Task 1	Task 2	Total	
A01	100%	100%	72.73%	27.27%
A02	75%	100%	87.5%	12.5%
E01	100%	100%	88.89%	11.11%
E02	75%	75%	66.67%	33.33%
F01	100%	100%	72.73%	27.27%
F02	75%	25%	44.44%	55.56%
Average	87.50%	83.33%	72.16%	27.84%

Table 3.13 Accuracy and false alarm rate by PDM for each data set

CHAPTER 4
DISCUSSION AND CONCLUSION

Our research mainly consisted by two parts. First, apply p-norm error measure instead of 1-norm measure in the linear programming discrimination, which generates a linear hyperplane to classify two data sets. With this p-norm error measure, the errors generated by the classifier are not treated equally but rather biased. For $p>1$, the bigger one error is, the more weight it obtains in the objective function.

Numerical results show this method can improve the result of classification and the accuracy is promising for Breast Cancer Data Set. However, for other data sets, the computational results are not good enough, which is due to the nonlinear character of these data sets. When $\delta_1 = 1/m$, $\delta_2 = 1/n$, the result is always much better than $\delta_1 = 1$, $\delta_2 = 1$ and $\delta_1 = 1/m^p$, $\delta_2 = 1/n^p$.

Second, investigation is conducted on a psychophysiological data set. Various methods are tested on this multi-dimensional time-series data set, from the linear programming method to the neural network method. With the help of DFT, The data is able to be transferred from the time domain to the frequency domain, in which the data set has interesting patterns. Generally speaking, for some of the EEG signals, a higher average magnitude of a low frequency band (1-10 Hz, etc) indicates a low cognitive activity, and the lower the magnitudes is, the more vigorously the brain acts.

The linear programming method can classify the NCL task and CL task with average accuracies 14.8% and 20.6%. However, the linear programming method cannot differentiate the low cognitive task and high cognitive task with comparable accuracies. The K-nearest neighbor methods can be applied in categorizing 0, 1 and 2 tasks. The average accuracies for the three tasks are around 60%, 40%, 50%, which are depending on the parameters of the algorithm. Furthermore, artificial neural network have also been tested for this data set. The optimal average accuracy is 85.05%, 47.08%, and 59.44%.

However, the way that neural network is utilized makes it impossible for real-time data mining, which may be essential in practice.

In data mining, it is difficult to evaluate performance of an algorithm or compare two algorithms without a specific data set so it is hard to say a certain algorithm is “better” than another in general. An algorithm which wonderfully performs on a data set may misclassify many data points in another data set. This fact is all because the domain knowledge of specific data set is poor, which is also part of the reason why data mining algorithms are applied. It is a dilemma. Therefore, the challenging questions are: does that choosing a DM algorithm all depend on the type of data set or something else? Are there any measurements can indicate which algorithms should be applied after a data set is given but before any algorithms are tested on it?

In our case, even the same data set has been tested by different algorithms, comparison between the performances of two algorithms applied in two different procedures is inappropriate. We applied linear programming method, k-nearest neighbor method and artificial neural network on this psychophysiological data set. These three algorithms could be applied in different occasions and have their own advantages and limitations. Generally, Linear programming method has a better performance than k-nearest neighbor method in classify none cognitive tasks and cognitive tasks, which may due to that k-nearest neighbor method cannot capture the real linearity in the data set.

Additionally, According to the computational results, we could conclude that this psychophysiological data set is individual independent or even trial independent, which imply each subject or each trial has its distinct characteristics. Besides, there is no evidence so far that shows we can classify the data from one subject based on the data from another one. However, all the subjects do share some similar characteristics: in cognitive state: the signals tend to have the lower magnitudes in frequency domain and lower standard deviation in time domain. Nevertheless, these characteristics are not enough to identify the brain’ states.

From practical perspective, classifying the data set based on subject is more realistic and meaningful since only classifying the data from one trial cannot satisfy the requirements of real-time classification. The best and most reasonable way to apply classification in this situation is that: before an operator is allowed to control a real UAV, he/she goes to a training program and takes certain simulation tasks, in which personal data is collected and analyzed, then the parameters of the data mining algorithm are searched until required classification accuracy is achieved. Last, the operator can be in charge of a real UAV and performs in real missions, and the performer's functional states would be indicated by the algorithm with parameters previously fixed.

Furthermore, separating NCL task and CL task is much easier than categorizing different level of cognitive activities. An intuitive explanation would be that brain functions qualitatively differently in none cognitive behavior and cognitive behavior. Meanwhile, the difference between different level cognitive tasks is only quantitative. Besides, the tasks in the experiment may be too subjective to be differentiated. Therefore, making a clear definition or measurement on the cognitive level of tasks should be in the future research. Also, identifying the requisites from the task, and explore how these interact with the brain would help classifying this type of data set in the future. Due to the curse of dimensionality, an effective feature reduction method is in demand. Ideally, in order to reduce the dimensionality of the data set, a feature reduction method either chooses useful features out of total features or combines several features into one. In our research, the features are selected based on the data mining results, so they are simply chosen by trial and error.

Finally, quoting from Robert D. Small (Small, 1997), A great deal of what is said about data mining is incomplete, exaggerated, or wrong. When you undertake a data-mining project, avoid a cycle of unrealistic expectations followed by disappointment. Understand the facts instead, and your data-mining efforts will be successful.

APPENDIX A: CLASSIFICATION RESULTS FOR WISCONSIN BREAST CANCER DATA SET (ERROR RATE)

P-order	M=N=1			M=m, N=n			M=m^p, N=n^p		
	Benign	Malignant	total	Benign	Malignant	total	Benign	Malignant	total
1	2.64%	5.19%	3.53%	2.64%	3.29%	2.87%	2.64%	3.29%	2.87%
1.1	2.57%	5.19%	3.49%	2.77%	2.91%	2.82%	2.77%	2.91%	2.82%
1.2	2.57%	5.19%	3.49%	2.91%	3.16%	3.00%	2.84%	3.16%	2.95%
1.3	2.57%	4.81%	3.35%	2.91%	2.66%	2.82%	2.84%	3.04%	2.91%
1.4	2.64%	4.68%	3.35%	2.91%	2.53%	2.78%	2.84%	2.91%	2.86%
1.5	2.57%	4.68%	3.31%	2.91%	2.53%	2.78%	**2.77%**	**2.91%**	**2.82%**
1.6	2.57%	4.81%	3.35%	2.97%	2.53%	2.82%	2.77%	3.04%	2.86%
1.7	2.57%	4.68%	3.31%	2.97%	2.41%	2.77%	2.84%	3.04%	2.91%
1.8	2.57%	4.56%	3.27%	3.04%	2.41%	2.82%	2.84%	3.16%	2.95%
1.9	2.57%	4.56%	3.27%	**3.04%**	**2.28%**	**2.77%**	2.84%	3.29%	3.00%
2	**2.57%**	**4.43%**	**3.22%**	3.04%	2.28%	2.77%	2.84%	3.29%	3.00%
2.1	2.57%	4.68%	3.31%	3.11%	2.28%	2.82%	2.84%	3.42%	3.04%
2.2	2.57%	4.81%	3.35%	3.18%	2.28%	2.87%	2.77%	3.29%	2.95%
2.3	2.50%	4.94%	3.35%	3.24%	2.28%	2.90%	2.84%	3.42%	3.04%
2.4	2.50%	5.06%	3.39%	3.38%	2.28%	3.00%	2.84%	3.80%	3.18%
2.5	2.50%	5.19%	3.44%	3.38%	2.28%	3.00%	2.77%	4.05%	3.22%
2.6	2.50%	5.44%	3.53%	3.38%	2.28%	3.00%	2.70%	4.56%	3.35%
2.7	2.50%	5.57%	3.57%	3.38%	2.28%	3.00%	2.64%	4.43%	3.27%
2.8	2.43%	5.57%	3.53%	3.38%	2.53%	3.08%	2.64%	4.43%	3.27%
2.9	2.43%	5.57%	3.53%	3.31%	2.78%	3.12%	2.64%	4.43%	3.27%
3	2.43%	5.70%	3.57%	3.31%	2.91%	3.17%	2.64%	4.43%	3.27%
3.1	2.36%	5.82%	3.57%	3.31%	3.04%	3.22%	2.64%	4.68%	3.35%
3.2	2.36%	5.70%	3.53%	3.31%	3.16%	3.26%	2.50%	4.81%	3.31%
3.3	2.36%	5.95%	3.61%	3.31%	3.16%	3.26%	2.50%	5.19%	3.44%
3.4	2.30%	5.95%	3.58%	3.38%	3.16%	3.30%	2.50%	5.19%	3.44%
3.5	2.30%	6.08%	3.62%	3.38%	3.16%	3.30%	2.43%	5.19%	3.39%
3.6	2.30%	6.46%	3.75%	3.38%	3.29%	3.35%	2.43%	5.19%	3.39%
3.7	2.30%	6.58%	3.80%	3.31%	3.42%	3.35%	2.43%	5.44%	3.48%
3.8	2.30%	6.84%	3.89%	3.24%	3.42%	3.30%	2.36%	5.70%	3.53%
3.9	2.30%	6.96%	3.93%	3.11%	3.54%	3.26%	2.36%	5.70%	3.53%
4	2.30%	7.22%	4.02%	3.04%	3.54%	3.21%	2.36%	5.70%	3.53%

4.1	2.23%	7.22%	3.97%	3.04%	3.67%	3.26%	2.36%	5.95%	3.61%
4.2	2.23%	7.22%	3.97%	2.97%	3.80%	3.26%	2.36%	6.08%	3.66%
4.3	2.23%	7.09%	3.93%	2.91%	4.18%	3.35%	2.30%	6.33%	3.71%
4.4	2.23%	7.22%	3.97%	2.97%	4.18%	3.39%	2.30%	6.58%	3.80%
4.5	2.23%	7.34%	4.02%	2.97%	4.18%	3.39%	2.30%	6.58%	3.80%
4.6	2.16%	7.59%	4.06%	2.97%	4.18%	3.39%	2.30%	6.58%	3.80%
4.7	2.16%	7.72%	4.10%	2.97%	4.18%	3.39%	2.36%	6.71%	3.88%
4.8	2.23%	8.23%	4.33%	3.04%	4.30%	3.48%	2.36%	6.71%	3.88%
4.9	2.23%	8.61%	4.46%	2.97%	4.43%	3.48%	2.36%	6.96%	3.97%
5	2.23%	9.11%	4.63%	2.97%	4.43%	3.48%	2.36%	6.96%	3.97%

APPENDIX B: CLASSIFICATION RESULTS FOR THE DATA SETS FROM UCI

	M=N=1								
	Ionosphere			Pima			Sonar		
1	8.00%	33.10%	17.01%	41.56%	17.90%	26.14%	27.84%	31.25%	29.43%
1.1	8.00%	32.86%	16.92%	41.44%	18.26%	26.33%	29.73%	32.50%	31.03%
1.2	7.87%	32.38%	16.67%	41.89%	18.92%	26.92%	31.08%	32.19%	31.60%
1.3	8.13%	32.38%	16.84%	42.44%	19.28%	27.35%	27.84%	32.50%	30.02%
1.4	8.13%	30.71%	16.24%	41.56%	19.82%	27.39%	30.27%	31.88%	31.02%
1.5	8.00%	30.95%	16.24%	41.89%	20.30%	27.82%	28.65%	31.88%	30.16%
1.6	7.87%	30.71%	16.07%	42.11%	21.20%	28.48%	28.65%	28.75%	28.70%
1.7	8.00%	31.43%	16.41%	42.44%	21.38%	28.72%	27.84%	31.88%	29.73%
1.8	7.73%	31.19%	16.15%	42.33%	21.74%	28.91%	28.65%	31.56%	30.01%
1.9	8.00%	30.71%	16.15%	42.44%	21.86%	29.03%	31.35%	33.44%	32.33%
2	8.13%	30.95%	16.32%	42.67%	21.86%	29.11%	29.46%	31.25%	30.30%
2.1	8.13%	30.95%	16.32%	42.78%	21.98%	29.23%	29.19%	31.56%	30.30%
2.2	8.00%	31.19%	16.32%	42.67%	22.16%	29.30%	25.68%	32.81%	29.01%
2.3	7.60%	31.67%	16.24%	42.56%	22.22%	29.31%	26.76%	30.62%	28.57%
2.4	7.47%	31.90%	16.24%	42.78%	22.10%	29.30%	27.84%	35.00%	31.19%
2.5	7.60%	32.14%	16.41%	43.00%	22.28%	29.50%	28.38%	32.19%	30.16%
2.6	7.60%	32.38%	16.50%	43.22%	22.28%	29.57%	25.95%	33.75%	29.60%
2.7	7.60%	32.38%	16.50%	43.33%	22.34%	29.65%	31.89%	32.19%	32.03%
2.8	7.60%	32.38%	16.50%	43.44%	22.28%	29.65%	28.38%	33.13%	30.60%
2.9	7.60%	32.62%	16.58%	43.78%	22.40%	29.85%	28.65%	30.62%	29.57%
3	7.73%	33.10%	16.84%	43.78%	22.51%	29.92%	28.65%	30.94%	29.72%
3.1	7.60%	33.10%	16.75%	43.78%	22.69%	30.04%	26.22%	31.25%	28.57%
3.2	7.33%	33.10%	16.58%	43.78%	22.57%	29.96%	26.22%	33.44%	29.60%
3.3	7.33%	33.33%	16.66%	43.89%	22.75%	30.11%	27.84%	32.50%	30.02%
3.4	7.33%	33.10%	16.58%	43.89%	22.69%	30.07%	30.00%	30.94%	30.44%
3.5	7.20%	33.10%	16.50%	44.00%	22.87%	30.23%	26.22%	30.94%	28.43%
3.6	7.20%	33.33%	16.58%	44.11%	22.81%	30.23%	30.00%	30.63%	30.29%
3.7	7.20%	33.57%	16.67%	44.22%	22.99%	30.39%	27.57%	32.50%	29.88%
3.8	7.47%	33.57%	16.84%	44.33%	23.05%	30.46%	28.92%	30.62%	29.71%
3.9	7.33%	33.57%	16.75%	44.33%	22.99%	30.42%	28.38%	30.00%	29.14%
4	7.20%	33.81%	16.75%	44.44%	23.05%	30.50%	28.38%	28.75%	28.55%
4.1	7.20%	34.05%	16.84%	44.33%	23.23%	30.58%	28.92%	33.13%	30.89%

4.2	7.20%	34.29%	16.92%	44.22%	23.29%	30.58%	29.46%	34.06%	31.61%
4.3	7.20%	34.29%	16.92%	44.11%	23.29%	30.54%	26.76%	35.62%	30.90%
4.4	7.07%	34.29%	16.84%	44.11%	23.35%	30.58%	27.84%	35.00%	31.19%
4.5	6.93%	34.52%	16.83%	44.00%	23.41%	30.58%	27.84%	32.19%	29.87%
4.6	6.67%	34.76%	16.75%	44.00%	23.29%	30.50%	26.76%	33.44%	29.88%
4.7	6.67%	34.76%	16.75%	44.22%	23.35%	30.62%	29.19%	34.38%	31.62%
4.8	6.53%	34.76%	16.66%	44.22%	23.41%	30.66%	30.54%	32.50%	31.46%
4.9	6.40%	35.24%	16.75%	44.33%	23.53%	30.78%	29.19%	34.38%	31.62%
5	6.27%	36.43%	17.10%	44.22%	23.59%	30.78%	28.65%	33.12%	30.74%
average	7.47%	32.90%	16.60%	43.33%	22.13%	29.51%	28.48%	32.23%	30.23%

	M=m^p, N=n^p								
p	Ionosphere			Pima			Sonar		
1	11.33%	29.29%	17.78%	33.00%	26.83%	28.98%	22.97%	38.44%	30.20%
1.1	11.33%	29.05%	17.69%	33.00%	27.01%	29.10%	29.73%	32.50%	31.03%
1.2	11.33%	29.76%	17.95%	33.67%	26.77%	29.17%	31.08%	32.19%	31.60%
1.3	11.33%	30.48%	18.20%	34.44%	26.83%	29.48%	27.84%	32.50%	30.02%
1.4	11.87%	30.48%	18.55%	35.00%	26.83%	29.68%	30.27%	31.88%	31.02%
1.5	12.13%	29.52%	18.37%	35.56%	26.83%	29.87%	28.65%	31.88%	30.16%
1.6	12.13%	29.29%	18.29%	35.78%	26.71%	29.87%	28.65%	28.75%	28.70%
1.7	12.13%	28.81%	18.12%	36.22%	26.77%	30.06%	28.38%	34.69%	31.33%
1.8	12.00%	29.05%	18.12%	36.67%	26.53%	30.06%	29.73%	32.50%	31.03%
1.9	12.27%	29.05%	18.29%	37.22%	26.59%	30.29%	31.35%	33.44%	32.33%
2	12.27%	29.05%	18.29%	37.11%	26.77%	30.37%	29.46%	31.25%	30.30%
2.1	12.00%	30.00%	18.46%	37.44%	26.77%	30.49%	27.57%	28.12%	27.83%
2.2	11.87%	30.24%	18.46%	37.44%	26.95%	30.60%	25.68%	32.81%	29.01%
2.3	11.87%	30.48%	18.55%	37.67%	26.95%	30.68%	26.76%	30.62%	28.57%
2.4	12.27%	30.95%	18.98%	37.67%	27.13%	30.80%	27.84%	35.00%	31.19%
2.5	12.00%	31.19%	18.89%	37.89%	26.95%	30.76%	28.38%	32.19%	30.16%
2.6	12.13%	31.19%	18.97%	38.22%	27.13%	30.99%	27.30%	34.69%	30.76%
2.7	12.13%	30.95%	18.89%	38.67%	27.13%	31.15%	31.89%	31.88%	31.89%
2.8	11.87%	31.19%	18.81%	39.00%	27.19%	31.30%	29.73%	33.44%	31.46%
2.9	11.87%	31.19%	18.81%	39.33%	27.25%	31.46%	28.11%	31.56%	29.72%
3	11.73%	31.19%	18.72%	39.44%	27.31%	31.54%	28.38%	32.19%	30.16%
3.1	11.73%	30.95%	18.63%	39.56%	27.31%	31.58%	26.49%	34.06%	30.03%
3.2	11.60%	30.95%	18.55%	39.56%	27.31%	31.58%	29.19%	33.44%	31.18%

3.3	11.33%	30.95%	18.37%	39.78%	27.37%	31.69%	29.73%	33.75%	31.61%
3.4	11.33%	30.71%	18.29%	39.78%	27.31%	31.65%	31.35%	33.13%	32.18%
3.5	10.93%	30.71%	18.03%	39.89%	27.19%	31.61%	27.57%	30.31%	28.85%
3.6	10.80%	30.71%	17.95%	40.11%	27.25%	31.73%	29.73%	31.88%	30.74%
3.7	10.67%	31.19%	18.04%	40.11%	27.25%	31.73%	25.41%	35.31%	30.04%
3.8	10.13%	31.43%	17.78%	40.22%	27.19%	31.73%	27.03%	30.94%	28.86%
3.9	10.13%	31.43%	17.78%	40.67%	27.25%	31.92%	30.27%	29.38%	29.85%
4	10.13%	31.43%	17.78%	40.78%	27.25%	31.96%	28.92%	30.62%	29.71%
4.1	10.00%	31.67%	17.78%	41.00%	27.25%	32.04%	28.92%	31.25%	30.01%
4.2	9.47%	31.90%	17.52%	41.11%	27.31%	32.12%	29.19%	32.81%	30.88%
4.3	9.20%	32.14%	17.43%	41.00%	27.19%	32.00%	28.38%	35.31%	31.62%
4.4	9.07%	32.14%	17.35%	41.11%	27.31%	32.12%	28.38%	34.38%	31.19%
4.5	8.80%	32.86%	17.44%	41.11%	27.25%	32.08%	29.19%	32.19%	30.59%
4.6	8.80%	32.86%	17.44%	41.00%	27.25%	32.04%	27.57%	34.38%	30.75%
4.7	8.67%	33.33%	17.52%	41.00%	27.13%	31.96%	25.95%	33.75%	29.60%
4.8	8.80%	33.33%	17.61%	41.00%	27.01%	31.88%	29.46%	32.50%	30.88%
4.9	8.40%	33.81%	17.52%	41.00%	27.01%	31.88%	28.65%	34.69%	31.47%
5	8.27%	34.05%	17.52%	41.00%	26.95%	31.84%	16.76%	44.69%	29.82%
Average	10.93%	31.00%	18.13%	38.57%	27.06%	31.07%	28.24%	32.96%	30.45%

APEENDIX C: CORRELATION FOR EACH TRIAL IN THE PSYCHOPHYSIOLOGICAL DATA

E01	1	2	3	4	5	6	7	8
1	1	-0.012	0.0212	-0.0762	-0.0287	-0.0746	-0.0259	-0.019
2	-0.012	1	-0.1	-0.7283	-0.5557	-0.2712	-0.1926	-0.0067
3	0.0212	-0.1	1	-0.1232	0.7601	-0.2893	-0.0774	-0.4433
4	-0.0762	-0.7283	-0.1232	1	0.4693	0.6343	0.367	0.2765
5	-0.0287	-0.5557	0.7601	0.4693	1	0.0676	0.1338	-0.231
6	-0.0746	-0.2712	-0.2893	0.6343	0.0676	1	0.5531	0.6855
7	-0.0259	-0.1926	-0.0774	0.367	0.1338	0.5531	1	0.4617
8	-0.019	-0.0067	-0.4433	0.2765	-0.231	0.6855	0.4617	1
E02	1	2	3	4	5	6	7	8
1	1	-0.002	0.011	-0.076	-0.0424	-0.0807	-0.0335	-0.0278
2	-0.002	1	-0.0531	-0.8197	-0.629	-0.3855	-0.291	-0.0899
3	0.011	-0.0531	1	-0.1273	0.7084	-0.2657	-0.0634	-0.3531
4	-0.076	-0.8197	-0.1273	1	0.5214	0.6584	0.4296	0.2897
5	-0.0424	-0.629	0.7084	0.5214	1	0.1355	0.2014	-0.1252
6	-0.0807	-0.3855	-0.2657	0.6584	0.1355	1	0.6039	0.6551
7	-0.0335	-0.291	-0.0634	0.4296	0.2014	0.6039	1	0.5137
8	-0.0278	-0.0899	-0.3531	0.2897	-0.1252	0.6551	0.5137	1

F01	1	2	3	4	5	6	7	8
1	1	0.0044	0.0059	-0.0657	-0.0318	-0.0712	-0.047	-0.0324
2	0.0044	1	-0.106	-0.4065	-0.3115	-0.2249	-0.2417	-0.0659
3	0.0059	-0.106	1	0.0122	0.7998	-0.1453	0.0555	-0.3558
4	-0.0657	-0.4065	0.0122	1	0.3636	0.7001	0.6336	0.3733
5	-0.0318	-0.3115	0.7998	0.3636	1	0.0636	0.2686	-0.2126
6	-0.0712	-0.2249	-0.1453	0.7001	0.0636	1	0.7969	0.7387
7	-0.047	-0.2417	0.0555	0.6336	0.2686	0.7969	1	0.6411
8	-0.0324	-0.0659	-0.3558	0.3733	-0.2126	0.7387	0.6411	1
F02	1	2	3	4	5	6	7	8
1	1	-0.0066	0.0157	-0.0398	-0.0159	-0.0279	0.0059	0.0081
2	-0.0066	1	-0.1718	-0.365	-0.3203	-0.2529	-0.259	-0.1655
3	0.0157	-0.1718	1	0.032	0.8095	-0.0438	0.1535	-0.162
4	-0.0398	-0.365	0.032	1	0.4112	0.755	0.6482	0.5005
5	-0.0159	-0.3203	0.8095	0.4112	1	0.2447	0.4344	0.1042
6	-0.0279	-0.2529	-0.0438	0.755	0.2447	1	0.8417	0.8031
7	0.0059	-0.259	0.1535	0.6482	0.4344	0.8417	1	0.7922
8	0.0081	-0.1655	-0.162	0.5005	0.1042	0.8031	0.7922	1

APPENDIX D RESULT FOR K-NEAREST NEIGHBOR METHOD

Fz, F7		A02/A01			E02/E01			F02/F01		
Test Error	15s	0.909	0.091	0	0.81	0.167	0.024	0.818	0.182	0
		0.4	0	0.6	0.4	0.4	0.2	0.2	0.8	0
		0.2	0	0.8	0.429	0.143	0.429	0.167	0.333	0.5
	10s	0.894	0.106	0	0.825	0.127	0.048	0.769	0.2	0.031
		0.375	0.25	0.375	0.125	0.25	0.625	0.111	0.667	0.222
		0	0.125	0.875	0.3	0	0.7	0.111	0.444	0.444
	5s	0.812	0.135	0.053	0.691	0.183	0.127	0.638	0.26	0.102
		0.467	0.067	0.467	0.353	0.118	0.529	0.368	0.316	0.316
		0.063	0.188	0.75	0.105	0.263	0.632	0.15	0.15	0.7
Train Error	15s	1	0	0	1	0	0	0.889	0.111	0
		0	0.25	0.75	0	0.5	0.5	0.25	0.25	0.5
		0	0.5	0.5	0	0.5	0.5	0	0	1
	10s	1	0	0	1	0	0	0.889	0.111	0
		0	0.25	0.75	0	0.5	0.5	0.25	0.25	0.5
		0	0.5	0.5	0	0.5	0.5	0	0	1
	5s	1	0	0	1	0	0	0.889	0.111	0
		0	0.25	0.75	0	0.5	0.5	0.25	0.25	0.5
		0	0.5	0.5	0	0.5	0.5	0	0	1

APPENDIX E: RESULTS FOR NEURAL NETWORK

Algor1	A01		E01		F01	
28 input	AVE	STDV	AVE	STDV	AVE	STDV
state 0	0.7935	0.13618	0.7935	0.16878	0.829	0.10204
state 1	0.55	0.32914	0.35	0.24152	0.575	0.23717
state 2	0.6667	0.35137	0.5667	0.31624	0.6	0.21084
Algor1	A02		E02		F02	
28 input	AVE	STDV	AVE	STDV	AVE	STDV
state 0	0.8323	0.07573	0.8613	0.06978	0.8645	0.08437
state 1	0.55	0.28382	0.425	0.23717	0.625	0.29463
state 2	0.5667	0.27444	0.7333	0.21082	0.4667	0.32204

Algor2	A01		E01		F01	
28 input	AVE	STDV	AVE	STDV	AVE	STDV
state 0	0.9194	0.04623	0.8742	0.19021	0.8548	0.11608
state 1	0.4	0.21082	0.375	0.27003	0.4	0.31623
state 2	0.55	0.30732	0.5	0.26352	0.625	0.29463
Algor2	A02		E02		F02	
28 input	AVE	STDV	AVE	STDV	AVE	STDV
state 0	0.8935	0.05492	0.8323	0.1739	0.8645	0.04995
state 1	0.35	0.29345	0.5	0.20412	0.5	0.33333
state 2	0.475	0.2993	0.675	0.16874	0.725	0.2189

Algor1	A01		E01		F01	
7 input	AVE	STDV	AVE	STDV	AVE	STDV
state 0	0.7129	0.19442	0.8129	0.071	0.6774	0.149
state 1	0.275	0.14191	0.225	0.2189	0.2	0.22973
state 2	0.6667	0.31429	0.3333	0.27218	0.6333	0.29187
Algor1	A02		E02		F02	
7 input	AVE	STDV	AVE	STDV	AVE	STDV
state 0	0.771	0.08111	0.7	0.12075	0.7645	0.11076
state 1	0.2	0.22973	0.4	0.29345	0.3	0.2582
state 2	0.4667	0.32204	0.4	0.34428	0.3667	0.29188

Algor1	A01		E01		F01	
7 input	AVE	STDV	AVE	STDV	AVE	STDV
state 0	0.829	0.10757	0.8935	0.04574	0.8516	0.09273
state 1	0.375	0.3385	0.475	0.18447	0.35	0.29345
state 2	0.75	0.20412	0.6	0.35746	0.65	0.24152
Algor2	A02		E02		F02	
7 input	AVE	STDV	AVE	STDV	AVE	STDV
state 0	0.8871	0.0532	0.871	0.06083	0.8871	0.06671
state 1	0.525	0.2189	0.375	0.35843	0.45	0.22973
state 2	0.475	0.2189	0.6	0.1291	0.525	0.34258

REFERENCES

Keogh and Pazzani. (1998). An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feed back. *KDD'98* , 239-241.

Agrawal, Faloutsos and Swami. (1993). Efficient Similarity Search in Sequence Databases. *Proc. of the 4th Conference on Foundations of Data Organization and Algorithms.* Chicago.

Bechmann, Kriegel, Schneider, and Seeger. (1990). The R*-tree: an efficient and robust access method for points and rectangles. *ACM SIGMOD* , 322-332.

Bennett and Mangasariain. (1993). Bilinear Separtion of Two Sets in n-Space. *Computational Optimization and Application 2* , 207-227.

Bennett, Mangasarian. (1992). Robust Linear Programming Discrimination of Two Linearly Inseparable Sets. *Optimization Methods and Software 1* , 23-24.

Ben-Tal and Nemirovski. (2001). On Polyhedral Approximations of the Second-Order Cone. *Mathematics of Operations Research* , 193-205.

Bradley, Fayyad, Mangasarian. (1999). Mathematical Programming for Data Mining: Formulations and Challenges. *INFORMS Journal on Computing* .

Busygin, Prokopyev, Pardalos. (2007). An optimization-based approach for data classification. *Optimization Methods and Software* .

Das, e. (1998). Rule Discovery From Time Series.

Dreyfus, G. (2005). *Neural Networks-Methodology and Applications.* Pairs: Springer.

Duda, Hart, Stork. (2001). *Pattern Classification.*

Geurts, P. (2001). Pattern Extraction for Time Series Classification. *PKDD 2001* , 115-127.

Guttman, A. (1984). R-Tree A Dynamic Index Structure for Spatial Searching.

Keogh and Kasetty. (2003). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. *Data Mining and Knowledge Discovery* , 349-371.

Keogh and Pazzani. (2002). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration.

Krokhmal. (2007). Higher Moment Risk Measures. *Quantitative Finance* , 373-387.

Lee, etc. (2000). Similarity Search for Multidimensional Data Sequences.

Managasarian, O. (1992). *Mathematical Programming in Neural Networks.*

Mangasarian, Bennett. (1991). Robust Linear Programming Discrimination of Two Linearly Inseparable Sets. *Computer Sciences Technical Report #1054* .

Padmanabhan, T. (2003). On the use of optimization for data mining theoretical interactions and eCRM opportunities. *Management Science 2003 INFORMS*, (pp. 1327-1343).

Padmanabhan,Tuzhilin. (2003). On the use of optimization for data mining: theoretical interactions and eCRM opportunities. *Management Science 2003 INFORMS* , 1327-1343.

Pang-Ning Tan, Michael Steinbach, Vipin Kumar. (2005). *Introdunction to Data Mining.*

Popivanov, Miller. (2002). Similarity Search Over Time-Series Data Using Wavelets. *ICDE'02* .

Radivojac, Obradovic, Dunker, Vucetic. (2004). *Lecture notes in computer science.* Springer.

Rafiei and Mendelzon. (1997). similarity-Based Queries for Time Series Data.

Small. (1997). Debunking Data Mining Myths. *Information Week* .

Smith. (2002). *A Tutorial on Principle Componet Analysis.*

TAN. (1999). DATA MINING. *DATA MININNG JOURNAL* , 88-99.

Tan, Dowe. (2004). *Lecture Notes in Artificial Intelligence (LNAI).* Springer.

Tan, Steinbach, Kumar. (2005). *Introduction to Data Mining.*

University of California-Irvine, S. o. (n.d.). *UC Irvine Machine Learning Repository*. Retrieved from http://archive.ics.uci.edu/ml/

Wilson and Russell. (2007). Performance Enhancement in an Uninhabited Air Vehicle Task Using Psychophysiologically Determined Adaptive Aiding. *Human Factors* , 1005-1018.