CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

EEG Signal Analysis by Using SVM and ELM:

A graduate project submitted in partial fulfillment of the requirements

For the degree of Master of Science in Electrical Engineering

By

Yun Sun

August 2015

The graduate project of Yun Sun is approved:

Professor Benjamin F. Mallard	Date
Dr. Xiaojun, Geng	Date
Dr. Xiyi Hang, Chair	Date

California State University, Northridge

Table of Contents

List of Figures

List of Tables

Abstract

EEG Signal Analysis by Using SVM and ELM:

By

Yun Sun

Master of Science in Electrical Engineering

Brain Computer Interface (BCI) is a communication interface between the brain and an external device, which is often used to assist or repair human cognitive or sensory-motor functions. One type of brain signal used in BCI systems is the electroencephalogram (EEG). In this project, the two EEG signals that were analyzed were obtained by recording the EEG activity that was occurring when a person was moving their arms for the first case and their legs for the second case. These EEG signals were processed using a power line rejection notch filter, power spectral density analysis, Principal Component Analysis (PCA), and finally the mathematical based machine learning analysis using both Support Vector Machine (SVM) and Extreme Learning Machine (ELM). Machine learning is used to generate a model that could be used to accurately predict whether a person is moving their arms or their legs by applying the EEGs as inputs to the generated model and reading the output of the model. The goal of this project is to compare the

performance of SVM and ELM by using the accuracy of classification that each model produces.

Chapter 1: INTRODUCTION

A Brain Computer Interface (BCI) is a way to communicate between the brain and an external device, such as a computer. By using sensors put on the scalp, the external device can catch the brain response signal and do some processing, while the body is doing some movements or the brain is thinking. An application of BCI is intelligent prosthesis, which use the brain signal to control the fake leg or arm movements. In this project, the BCI input signal is the EEG signal that results from movement of the legs and hands.

For most BCIs, an electroencephalogram (EEG) is an electric recording of brain activity caused by extremely sensitive currents and the EEG signal, which is considered as the source signal, is sent to a computer to do the signal processing. For the brain signal measurement, the electroencephalogram (EEG) is one of the main methods, which provides a non-invasive measurement without implanting any foreign object inside the body. Depending on different situations, conditions and thought, the brain produces a different EEG signal, which can be captured by the sensors on the scalp. Because the amplitude of the EEG signal is very small, it is difficult to acquire the signal and once it is acquired, it often very noisy and requires filtering. The type of noise that surrounds the signal of interest is in the form of background noise or other signals generated by the

brain that are not pertinent to the particular application of interest. This is because of an external disturbance or other mental activity.

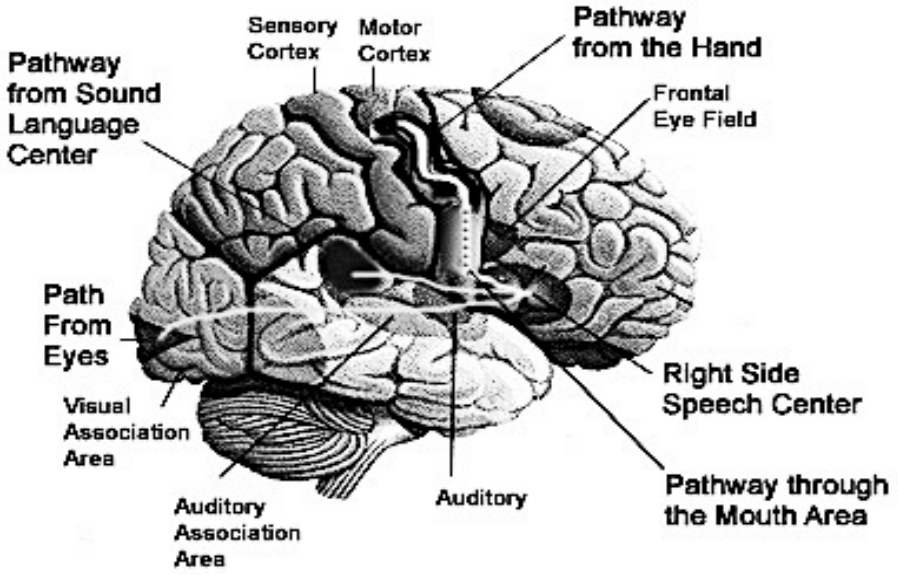

Figure 1: The areas of different brain signal processing (Stroke Family: The Sensory Trigger Method: Making New Pathways for Speech After Stroke or Brain Injury)

Different activities will cause different parts of the brain to process more intensively or relax more, so by analyzing the origin of a signal, one can gain insight as to what kind of activity is performed. Besides the origin of the signal, the amplification or attenuation of the signals, as well as the frequency of the signal help categorized the type of signal being observed. The frequency bands often used in analyzing EEG signals are defined as Delta (0 Hz – 4 Hz), Theta (4 Hz – 7 Hz), Alpha (8 Hz – 12 Hz), Beta (12 Hz – 30 Hz), Gamma (30 Hz – 100 Hz). The excess or lack of activity can be used to measure the state of a person. For example, if the brain of an awake adult is experiencing large amount of

delta activities or theta activities, it means the person has some neurological diseases. A person's state of alertness can be predicted by looking at the amplitude of alpha rhythms, which increase when the body is relaxed and the eyes are closed, otherwise it attenuates. Beta rhythms relate with muscle movements. The gamma waves relate to certain muscle functions and perceptions [1].

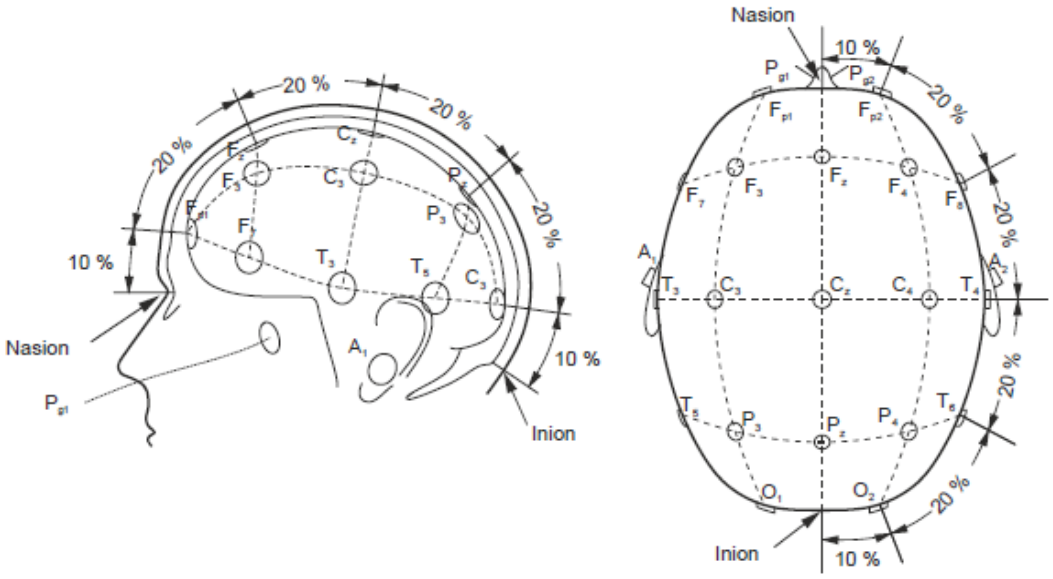

Figure 2: Electrode Placements over Scalp (Gomez-Gil, Sensors 2012, 12(2), 1211-1279; doi:10.3390/s120201211)

Machine learning plays a major role in BCI systems. It was developed from the study of pattern recognition and computational learning theory in artificial intelligence and in this project, it is used to learn the patterns in EEG activity for cases when a person is moving their arms and when they're moving their legs. There are two main applications of machine learning: regression and classification. The regression analysis is widely used

for prediction and forecasting. The regression analysis The regression analysis is a process for estimating the relationships and the conditional expectation among the dependent variable and one or more independent variables, which includes many techniques for modeling and analyzing several variables. The estimation target is a function of the independent variables called the regression function [2]. In regression, the outputs of the system are continuous rather than discrete. On the other hand, classification is the process of identifying which set of discrete categories certain data belongs to. Machine learning can generate a model, which sets boundaries to classify data as belonging to one class or another. This is achieved by first extracting features from the training dataset and then progressively updating the model to increase the accuracy with which it predicts the class of training data. The category of machine learning that contains a label for each of the trials in a dataset is called supervised learning. This data and the labels are used to generate a mathematical model that can be used to predict which class other EEG data belongs to. This is done by feeding features from a trial into the model and looking at the predications that the model makes. If the model works well, it will correctly predict which class the data belongs to. To build this model, training data is required to progressively help shape the parameters in the model so that it can be trained to make data driven predictions and perform accurate signal classification on future data. The two types of data in machine learning are training data and testing data. Training data is used to build the model by providing some feedback to the system that indicates which class the data belongs to. This allows the model to constantly change its parameters to

allow the training data to be classified correctly. The ultimate goal of machine learning does not lie in the accuracy of prediction for the training data, but instead what is most desired is to have the model be general enough that the model can correctly classify future data that the model has not seen before. To get a measure of how good the model will perform, testing data is used to test the predications that the model produces and check the accuracy of the model. This testing data is data that the model has not seen before and therefore this accuracy is a measure of how accurate it is in the general case (not just for data that it used to generate the model). In this project, there are two machine learning algorithms used: Support Vector Machine (SVM) and Extreme Machine Learning (ELM). The models that were generated in this project are specifically catered to learning the difference between the patterns in the two classes of EEGs (hands and legs). More details about machine learning are described in Chapter 3.

The dataset provided for this project is large and not every single point of data is needed to describe the characteristics or pattern of the signal. The system attempts to classify the data as being arm movement related or leg movement related using whatever data is provided to it, but for better processing efficiency, the number of data points that are processed can be reduced by extracting certain patterns from the data and then use this as the input to the learning algorithm. This provides a more efficient way of processing the signal, since instead of processing a huge set of data, the system only has

to process a few of patterns/features. The process of extracting these patterns is called feature extraction and is discussed in Chapter 2.

On this project, the purpose is to compare two machine learning models performances: Support Vector Machine (SVM) and Extreme Machine Learning (ELM) by measuring the accuracy of the predictions that each model makes based on the analysis and feature extraction of the electroencephalogram (EEG) signals for the movement of either the legs or arms moving. The procedure followed in this project can be summarized as following: run the data through filters, use PCA to extract features of the filtered data set, run the features through a machine learning algorithm (SVM and ELM) to generate a model, test the model and record accuracies of the learning algorithm using different machine learning functions, parameters and models.

In Chapter 2, the paper describes the pre-processes, which include the following steps: transform the dataset from time domain to frequency domain (power density spectrum) and feature extraction. In Chapter 3, the paper describes the math details of ELM and SVM. Project details, results and the conclusion can be found in Chapter 4 and Chapter 5.

Chapter 2: PRE-PROCESSING

The datasets used in this project is the EEG activity signals, which are collected over 15 channels and are recorded at a sampling frequency of 512 Hz when the hands or foot of participants were moving. Because the useful parts of the collected brain response signals are lower than 100Hz, signals are band pass filtered from 0.5 Hz to 100 Hz. Because the data used in this project is from Germany, there is a very strong noise at 50 Hz, which will affect the result. Therefore, there is a four-tap notch filter in the system, which cleans the power noise as much as possible. The data used in the project is after the filter and contains only the relevant portions from the data.

Frequency Transformation

The data is transformed from time domain to frequency domain so that the latent periodic components can show up and be identified. It is also easier to see the data differences in frequency domain and figure out which frequency bands of the brain signal are active, when the body is moving the arms or legs.

In this project, the time domain EEG signals are transformed into the frequency domain using spectrum density estimation to obtain the power density spectrum. The power spectrum shows how the power varies across different frequencies, which can also be used to show how the energy is distributed. In statistical signal process, spectrum

density estimation is a tool to estimate the spectral density of a random signal from a sequence of time samples of the signal, which is a way to describe the frequency features and content of spectrum density. One purpose of spectral density estimation is to detect any periodicities in the data. There two main kinds of SDE: parametric SDE and non-parametric SDE [3]. For non-parametric spectrum estimation, no special model is assumed to generate the frequency response data. In this case, it can use Fourier methods to analyze the power in the frequency domain. The parametric spectrum estimation uses a predefined model to represent the data in the frequency domain. These models require calculating certain parameters to fit the model to the data.

In this project, a non-parametric spectrum estimation method was used to transform the time domain signal into one in the frequency domain. The name of this method is called Welch's method. Welch's method is based on the standard density spectrum estimation and one of its features is that it can be used to reduce noise in exchange for reducing the frequency resolution to the desire of the user. In this method, the signal data segments are split up into L data segments, which have M points, and is overlapped by D points. If D = M/2, the overlap is 50% and if D = 0, the overlap is 0%. Each of these data segments can then be multiplied with a window function and the periodogram of the product is calculated for the individual segments and the average of these periodograms is the result of Welch's method. The equations for the process that was just described are shown below.

$$\tilde{P}_{xx}^{(i)}(f) = \frac{1}{MU}\left|\sum_{n=0}^{M-1} x_i(n)w(n)e^{-j2\pi fn}\right|^2, \quad i = 0,1,\ldots,L-1 \tag{2.1}$$

$$U = \frac{1}{M}\sum_{n=0}^{M-1} w^2(n)$$

$$P_{xx}^{W}(f) = \frac{1}{L}\sum_{i=0}^{L-1} \tilde{P}_{xx}^{(i)}(f)$$

The data used in this project is cut into no more than 8 segments with 50% overlap and the modified periodgram is computed for each segment using a Hamming window. Compared to the rectangular window, the magnitude of the side lobes for the hamming window is much less than that of the rectangular, which is desirable, but the main lobe of the hamming is also wider, which is undesirable.

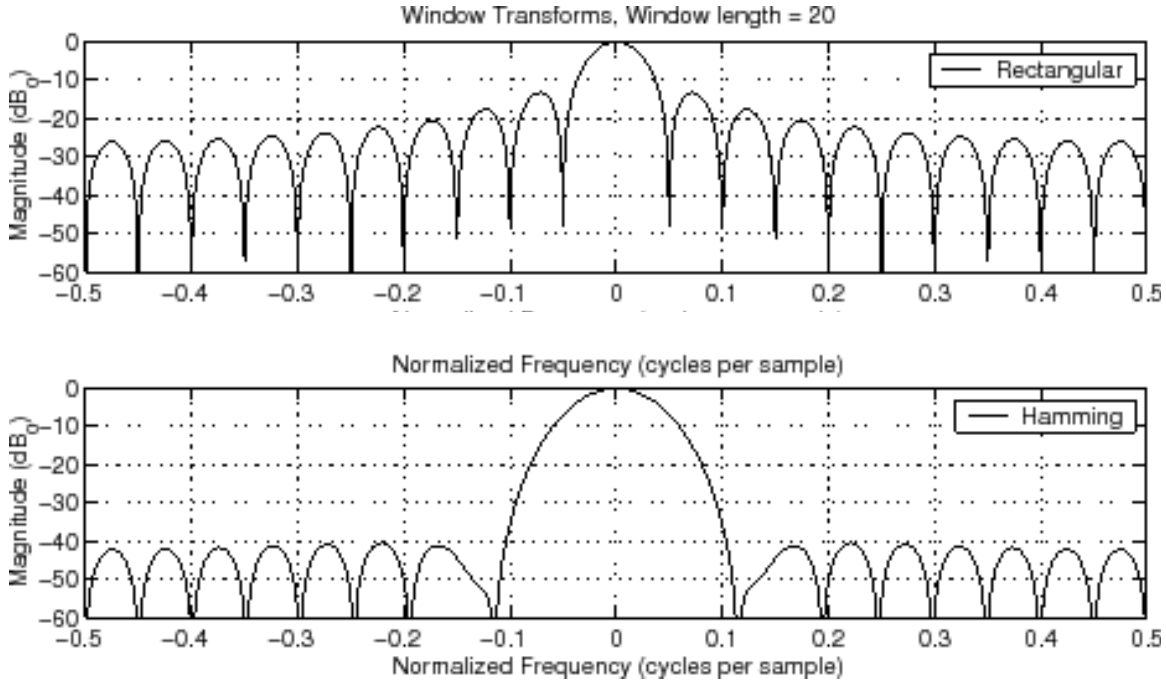

Figure 3: Compare the rectangular window with hamming window

Feature Extraction

A feature is an obvious characteristic, which can describe the data. As mentioned in Chapter 1 Introduction, there is a large set of data provided in this project. In order to process data more efficiently, the project used feature extraction to get the features, which can be very useful to describe the pattern or trend, since they provide a more compact way of describing the pattern.

Principal Component Analysis (PCA) is a very effective algebraic feature extraction method to deal with the information in the sample, compression and extraction based on the variables covariance matrix. The core idea of PCA is using the characteristics of less data to describe samples in order to reduce the dimension of the feature space. PCA is often used to reduce the dimensions of the data sets, while keeping the largest contribution to the variance of characteristics of the data set, by retaining lower-order principal component and ignoring higher-order principal component. In this way, lower-order components retain the most important aspect of the data.

To build up the PCA model, the first step is to get the mean of each channel and subtract the mean from every data dimensions in order to set the means of every dimension data to zero. Then, the covariance matrix is calculated using the zero mean data. After this, the the eigenvalues and eigenvectors of the covariance matrix are found, the eigenvectors are sorted by the eigenvalues from highest to lowest, which results in data from most significance to lowest significance. Lastly, components of lowest

significance can be ignored without much loss in information and multiply the original data with the eigenvector matrix with to get the new data.

Chapter 3: CLASSIFICATIONS

Machine learning

Machine learning is an algorithm operation that builds up a model with the training data and uses the model to process some test data. Machine learning is used in many applications. The one used in this project is supervised learning and classification. To supervise the learning process, the data provided to the system must have the predicted class labels for each set of features. Using the labels, the feature space for the training data is divided into two classes and the algorithm learns the general rules to build up the model. Then this model is used to classify the testing data and to calculate the accuracy of the system. There are two kinds of machine learning used in this project: the first is Support Vector Machine (SVM), and the second is Extreme Machine Learning (ELM).

Support Vector Machine

Support Vector Machine (SVM) is a traditional algorithm operation to separate two or more kinds of data. It relies on a mathematically defined function that represents a division, known as a hyper plane in higher dimensions, between the two classes of data. Data lying on one side of this hyper plane is classified as belonging to one class, while data on the other side belongs to the other class. By applying SVM, a classification model in the form of linear function $f(x) = wx+b$ can be generated, which can separate data to

the classes. The location of the hyper plane is defined as f(x) = 0 = wx+b. The example in Figure 3 shows some blue circles on the left side, which represent data belonging to Class 1, and some red circles on the right side, which represent data belonging to Class 2. The blue circles are on the side of the hyper plane, which corresponds to the case when f(x) > 0 and the red circles are on the side of the hyper plane, which correspond to the case where f(x) < 0. By looking at the location of the data with respect to the hyper plane, the class that the data belongs to can be predicated by the model.

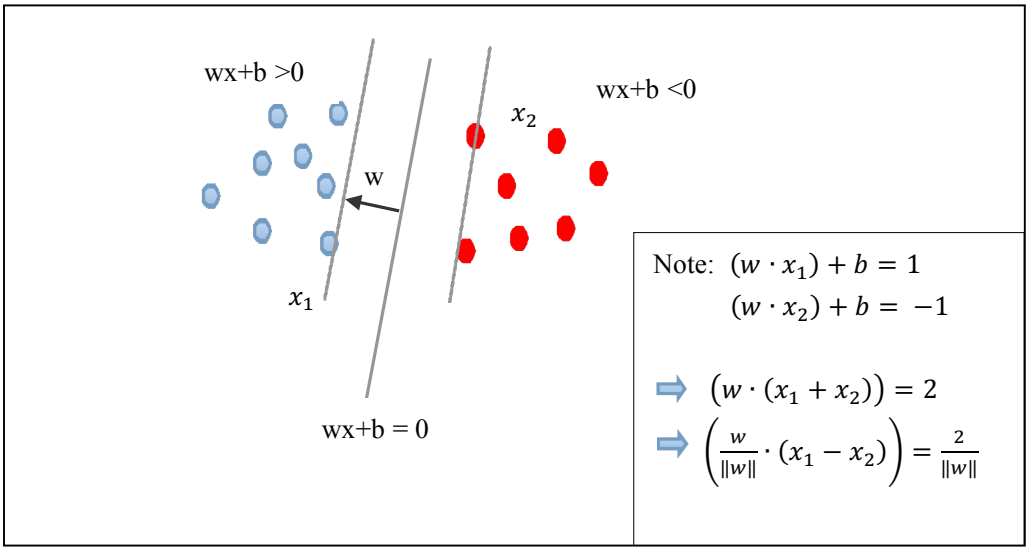

Figure 4: How do the linear SVM separate two different kinds of data

From Figure 4, there are two parallel lines beside the line at wx+b = 0. The line is used to separate between two classes (red circles vs blue circles). The wider the area between the red circles and blue circles, the easier it is to separate the circles and therefore, the easier it is to separate between the two classes. Putting any line in between

the two classes will work for the example shown in the figure, but since the goal is to make the model work for data not belonging to the training dataset, it is best to provide the widest margin between the two classes. The best condition is when the maximum distance between two hyper-planes at the edges occurs. The distance is 2/‖w‖, so max value of ‖w‖ is needed to minimize the distance. When the w is fixed, the three lines are also fixed. As the norm of w uses the square root, instead of using ‖w‖, $\frac{\|w\|^2}{2}$ will be used instead. Therefore, it can formulate a quadratic optimization problem and solve for w and b:

$$min: \frac{\|w\|^2}{2} \tag{3.1}$$

$$subject\ to: y_i(wx_i + b) \geq 1, \forall i$$

$$Primal\ classification\ equation: f(x) = sign(w^T x + b)$$

where the goal is to find w and b such that $\Phi(w) = 1/2w^T w$ is minimized and for all $\{(x_i, y_i)\}$: $y_i(wx_i + b) \geq 1$. Quadratic optimization problems are a famous class of mathematical programming problems and many algorithms exist for solving them [3]. Quadratic programming is a special type of mathematical optimization problem, which minimize or maximize a quadratic function of several variables subject to linear constrains on the variables.

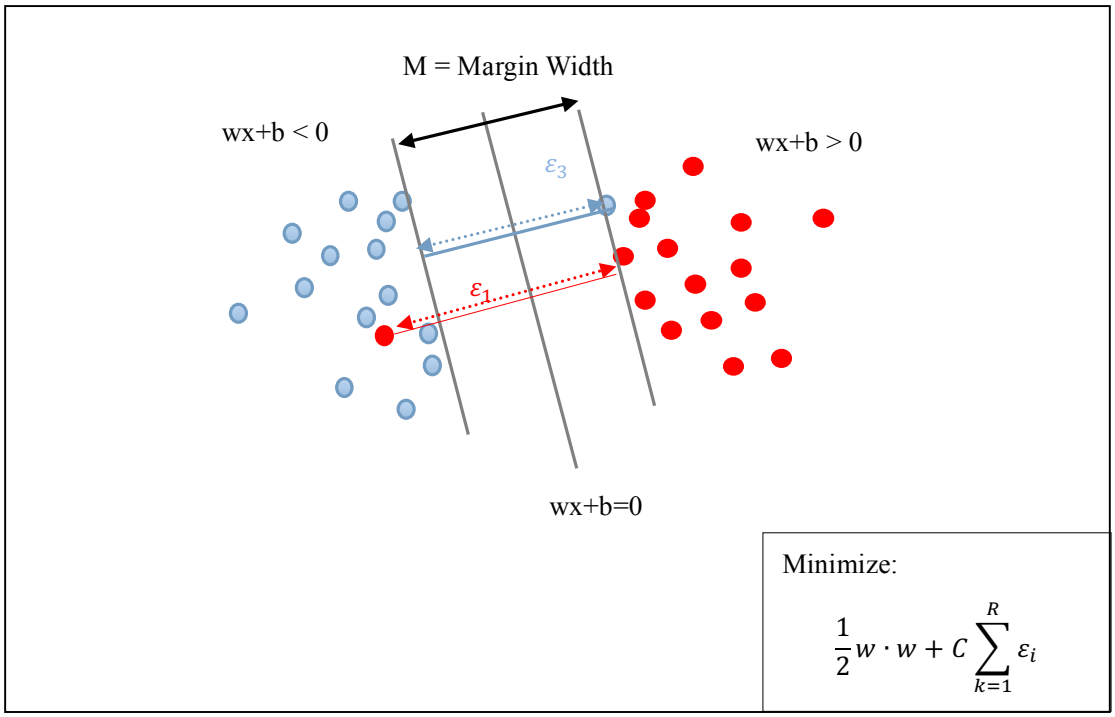

Figure 5: The Soft margin property of SVM

There will be times where there is some data that cannot be separated. From Figure 4, one or two blue circles show up on the side of red circles, which cannot generate a clear boundary to separate the circles by color. The solution to this problem is using a concept known as soft margin. According to the study done by Corinna Cortes and Vladimir N.Vapnik, they found the soft margin in 1995 [5]. In the situation described, the soft margin method can choose a hyper plane that keeps the maximum distance and splits the data as cleanly as possible. In this method, there are non-negative slack variables ε_i, which checks the measurement degree of the misclassified data x_i.

Now, using the soft margin, the equation changes to

$$minimize: \frac{1}{2} w \cdot w + C \sum_{k=1}^{R} \varepsilon_i \tag{3.2}$$

$$Subject\ to: y_i(wx_i + b) \geq 1 - \varepsilon_i, and\ \varepsilon_i \geq 0, \forall i$$

For dual problem [5]:

$$max: Q(\alpha) = \sum \alpha_i - 1/2 \sum \sum \alpha_i \alpha_j y_i y_j x_i^T x_j \tag{3.3}$$

$$Subject\ to: \sum \alpha_i y_i = 0 \quad \alpha_i \geq 0 \text{ for all } \alpha_i$$

$$classifying\ function: f(x) = \sum \alpha_i y_i x_i{}^T x_j + b$$

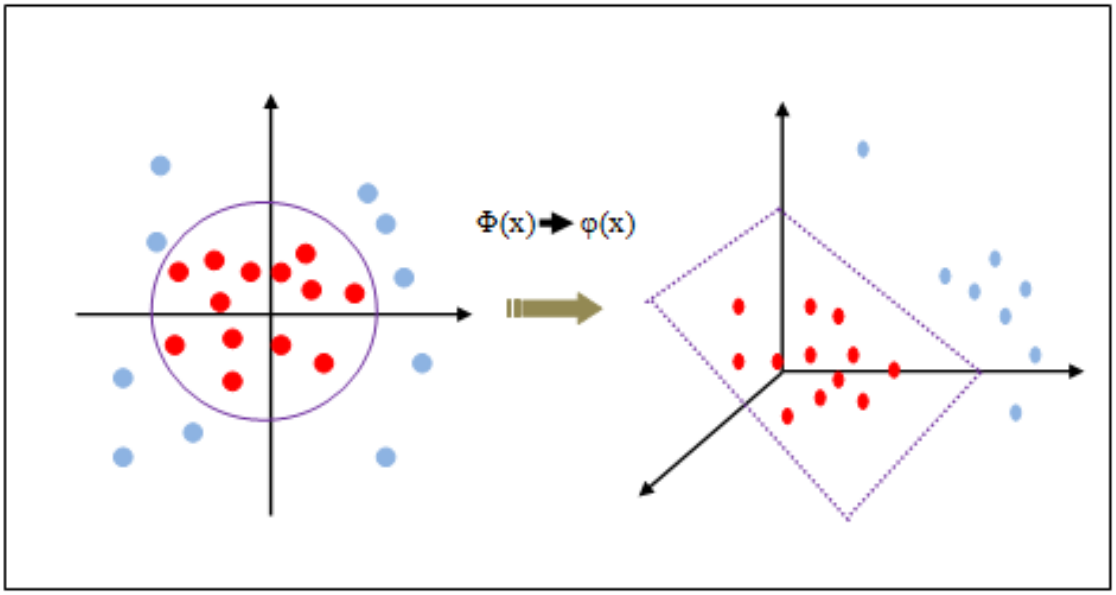

Figure 6: map the original input space to the higher dimensional feature space

Since the data is not always linearly separable in the initial feature space, there are occasions where increasing the dimensionality of the feature space may actually make the separation of classes much easier. To perform this process, the kernel function is used to transform the initial feature space into a higher order feature space, which can make the data be linearly separated by a hyper-plane and achieve the maximum margin between the two classes [6]. As shown in Figure 5, the initial data is transformed from Φ(x) to φ(x) so that the dot product becomes:

$$K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j) \tag{3.4}$$

And the equation (3.1) changes to:

$$Learning: \quad min\|w\|^2 + C\sum_i^N \max(0, 1 - y_i f(x_i)) \tag{3.5}$$

$$classifying\ function: f(x) = sign(w^T \varphi(x) + b)$$

Instead of $x_i^T x_j$, which shows in Eq.6, the non-linear SVM mathematical classification uses $K(x_i, x_j)$ and for dual classifier in transformed feature space:

$$\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2}\sum_{jk} \alpha_j \alpha_k y_j y_k k(x_j, x_k) \tag{3.6}$$

$$Subject\ to: 0 \leq \alpha_i \leq C\ for\ \forall i\ and \sum_i \alpha_i y_i = 0$$

and the classifying function is shown as following:

$$Classifying\ function: f(x) = \sum a_i y_i K(x_i, y_i) + b \tag{3.7}$$

Extreme Learning Machine

Huang Guang-Bin presented Extreme Learning Machine in 2004 [7]. At first, ELM was designed for the single hidden layer feed-forward neural networks (SLFNs). Then, it is developed and extended to the generalized single hidden layer feed-forward networks [8] [9] [10] [11].

The main idea of ELM is to work for the SLFNs, and extends to the multi-hidden layer feed-forward neural networks. The ELM theory assumes that the hidden nodes or the neurons do not need to be tuned and can be generated randomly as shown in Figure 7. The parameters of the hidden nodes are independent from the training data set. The ELM theories state that the randomness may be the real method for how the brain learns.

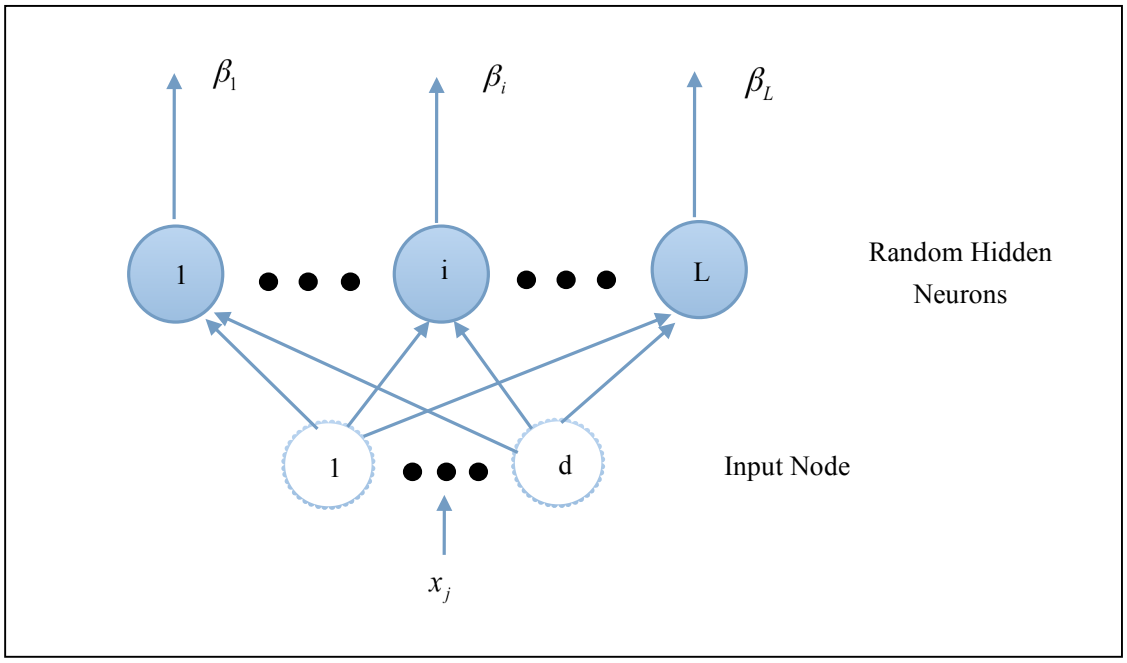

Figure 7:The workflow picture of Extreme Learning Machine

In order to describe the ELM model further, the following parameters need to be defined:

- N: the total number of the training data
- $\tilde{N}$: the total number of hidden nodes
- n, m: the dimensions of the input layer and output layer
- $(x_j + t_j), j = 1,2,....., N$: the testing data

$$x_j = (x_{j1}, x_{j2},....., x_{jn})^T \in R^n, t_j = (t_{j1}, t_{j2},....., t_{jm})^T \in R^m.$$

Put the output vectors together to get the output matrix:

$$T = \begin{bmatrix} t_1^T \\ t_2^T \\ . \\ . \\ . \\ t_N^T \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & . & . & . & t_{1m} \\ t_{21} & t_{22} & . & . & . & t_{2m} \\ & & . & & & \\ & & & . & & \\ & & & & . & \\ t_{1m} & t_{2m} & . & . & . & t_{Nm} \end{bmatrix} \tag{3.8}$$

- $o_j, j = 1,2,..., N$: the particular output vector according to the t_j
- $W = (w_{ij})_{\tilde{N} \times n}$: the related weight matrix of input layer and hidden layer

$$w_i = (w_{i1}, w_{i2},..., w_{in})^T$$

- $b = (b_1, b_2,..., b_{\tilde{N}})^T$: the offset vector, b_i is the threshold of the i^{th} hidden node
- $\beta = (\beta_{ij})_{\tilde{N} \times m}$: the related weight matrix of output layer and hidden layer

$$\beta_i = (\beta_{i1}, \beta_{i2}, ..., \beta_{in})^T$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \beta_2^T \\ \cdot \\ \cdot \\ \cdot \\ \beta_{\tilde{N}}^T \end{bmatrix} = \begin{bmatrix} \beta_{11} & \beta_{12} & \cdot & \cdot & \cdot & \beta_{1m} \\ \beta_{21} & \beta_{22} & \cdot & \cdot & \cdot & \beta_{2m} \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ \beta_{\tilde{N}m} & \beta_{\tilde{N}m} & \cdot & \cdot & \cdot & \beta_{\tilde{N}m} \end{bmatrix} \tag{3.9}$$

- $g(x)$: active function

Single-Hidden Layer Feed-Forward Network (SLFNs)

Mathematically, the standard SLFNs model is the following:

$$\sum_{i=1}^{\tilde{N}} g(w_i \bullet x_i + b_i)\beta_i = o_j, j = 1,2,...,N, \tag{3.10}$$

In order to have zero error and be closer to the N samples,

$$\sum_{j=1}^{N} \left\| o_j - t_j \right\| = 0 \tag{3.11}$$

So the Eq. 5 change to

$$\sum_{i=1}^{\tilde{N}} g(w_i \bullet x_i + b_i)\beta_i = o_j, j = 1,2,...,N, \tag{3.12}$$

Use the matrix to describe Eq.16

$$H\beta = T \tag{3.13}$$

H is the hidden layer output matrix of neural network. $H = H(W,b) = (h_{ij})_{N\times\tilde{N}}$ and $h_{ij} = g(w_j \bullet x_i + b_j)$.

When the number of the hidden units is odd and $\tilde{N} = N$, and the matrix is revertible, the Eq. 14 contains the only solution.

However, in most cases, the number of hidden-layer units is much smaller than the number of training data, which means $\tilde{N} << N$. To solve the difficulties, a new effective learning machine appears. At first, the values of parameters 'W' and 'b' can be given randomly and used to calculate the matrix 'H' (although keep changing the values of 'W' and 'b' does not result in any gain change.) 'β' is the only parameter is used as a decision factor.

When 'W' and 'b' is fixed, Eq. 8 is used to solve for the least square solution $\hat{\beta}$ of the linear system.

$$\left\|H\hat{\beta} - T\right\| = \min_{\beta}\left\|H\beta - T\right\| \tag{3.14}$$

$$\hat{\beta} = HT$$

Because $\hat{\beta}$ is the least square solution of Eq. 8, Eq. 9 also works. On the other hand, $\hat{\beta}$ is also the minimum norm, which means $\hat{\beta}$ is the minimum norm of all the least square solutions. For the feed-forward neural networks, the magnitude of $\hat{\beta}$ is very important: $\hat{\beta}$ is smaller, while the generalization ability of the system is better.

The Extreme Learning Machine build up process:

Given the training data set $R = \{(x_i, t_i) \mid x_i \in R^m, t_i \in R^m\}_{i=1}^{N}$, the active function $g(x)$ and the number of hidden units $\tilde{N}$.

Step 1: Assign input weights w_i and threshold $b_i, i = 1,2,..., \tilde{N}$.

Step 2: Calculate the hidden layer output matrix of neural 'H'.

Step 3: Calculate the related weight matrix of output layer and hidden layer $\beta = HT$.

Chapter 4: NUMERICAL EXPERIMENT

The data used in the project are EEG activity signals, which download from the website: http://bnci-horizon-202.eu [9]. The data includes the training data, the testing data, trials and predicted class labels. Trials are used to separate the data for each sample and the predicted class labels assist the training and the calculation of the accuracy for the classification processes. The training data and the testing data had 15 channels and were recorded using a sampling frequency of 512 Hz, when the hands and foot are moving. There are 160 samples: 100 training sample data and 60 testing sample data, where each one has over 11000 points. Because the frequency of the signals is lower than 100 Hz, which include the main useful information, the EEG signals are band-pass filtered from 0.5Hz to 100Hz. There is also a four-tap 50 Hz notch filter in the system to clean the power noise. The data used in the project only contains the relevant portions. After loading the data to MATLAB, the mean of each class of 15 channels in time domain is plotted, which are shown in the following Figure 7. Then the data is transformed from time domain to power density spectrum through the non-parametric model Welch, which has been introduced in Chapter 2. In MATLAB, the function ‘pwelch’ is used to perform the operation and the result is showed as Figure 8. The last step of the data pre-process is to let the data run through PCA to get the features extracted.

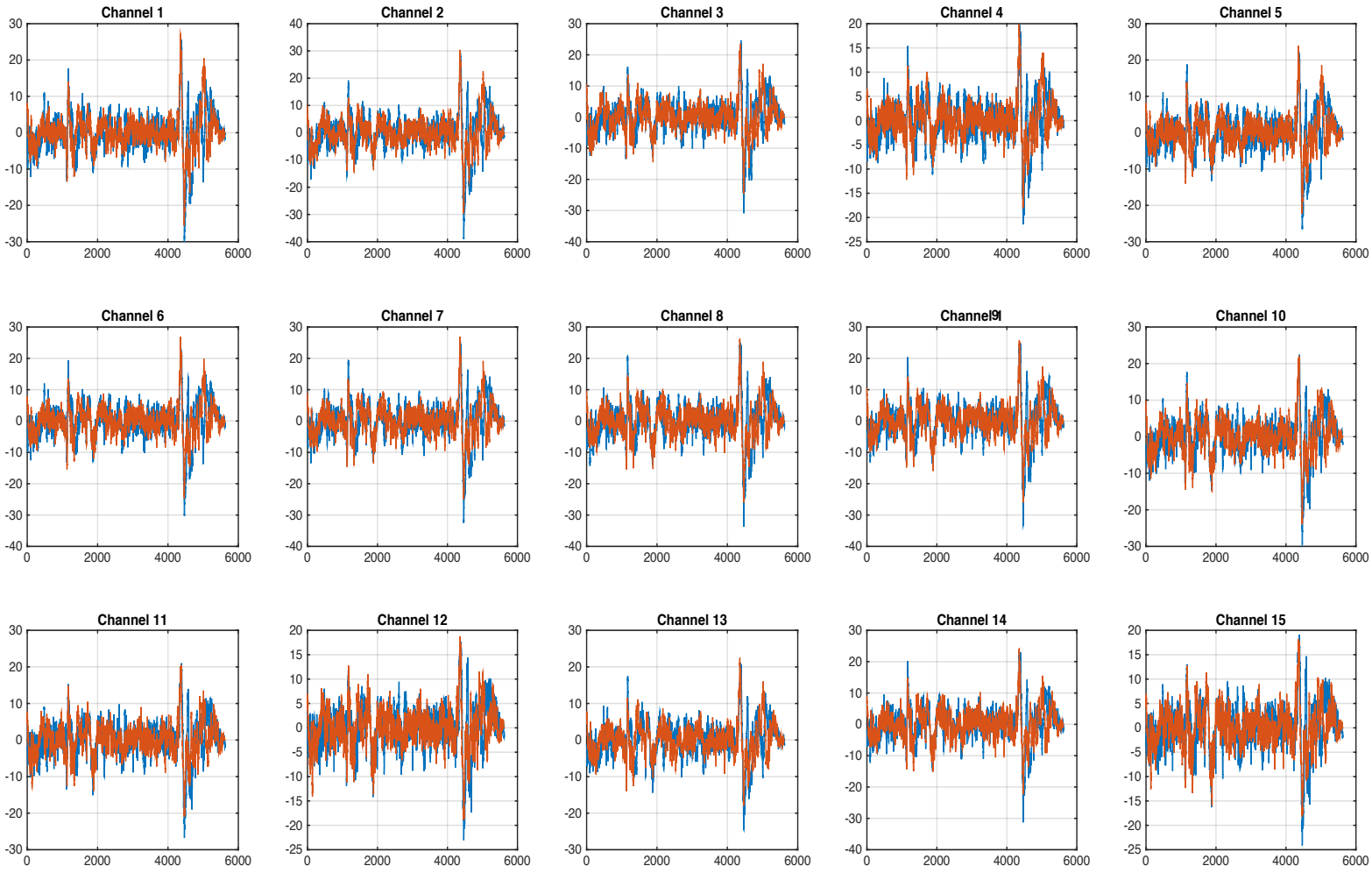

Figure 8: mean of each channel signal in time domain

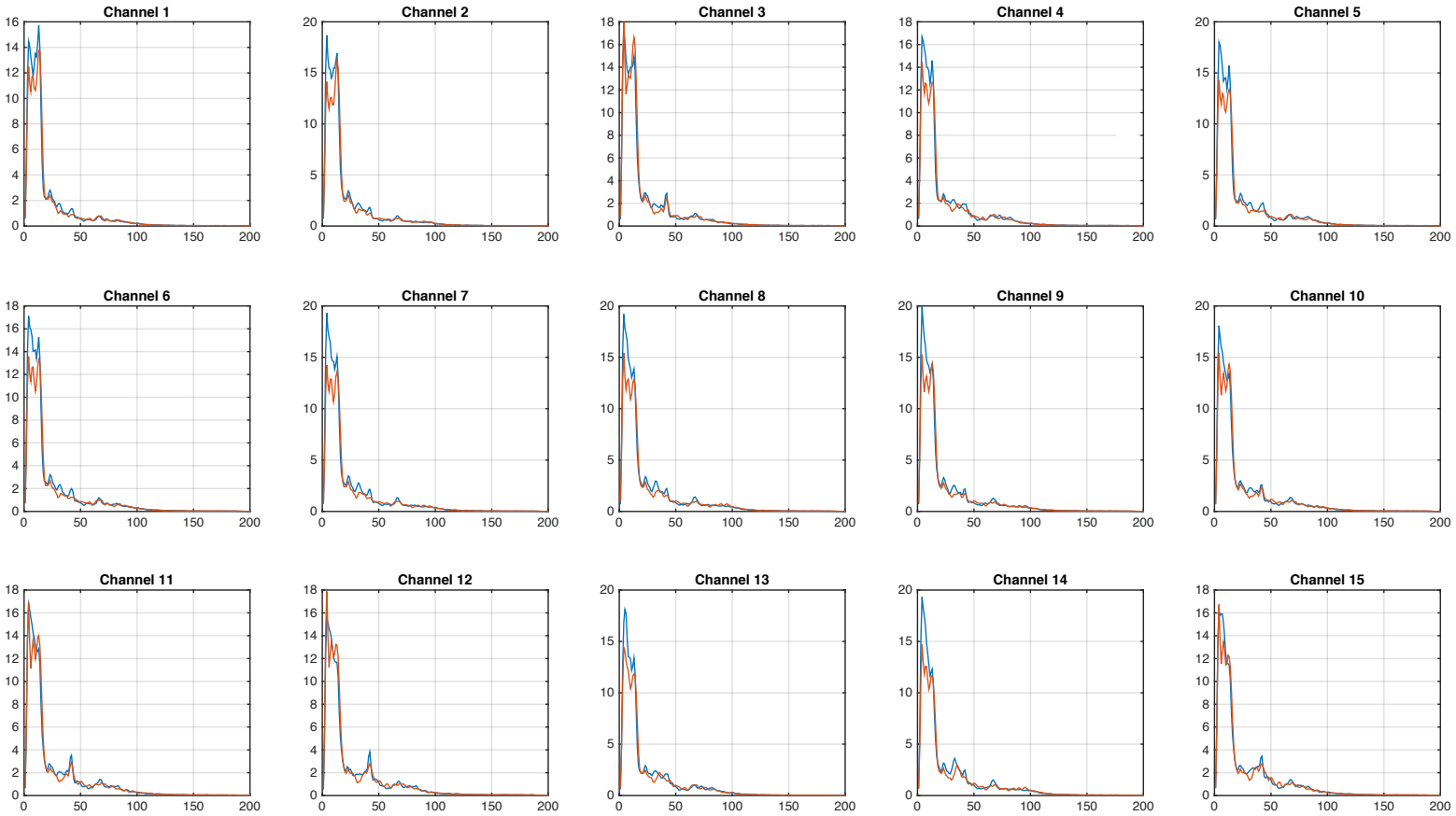

Figure 9:Power density spectrum of Class 1 and Class 2

website: http://www.ntu.edu.sg/home/egbhuang/elm_kernel.html [10]. This model was built by Prof. Huang Guang-Bin study team, which sets up the theories and models of ELM [11]. For SVM model, it is build up by the MATLAB functions: svmtrain.

For Table 1 and Table 2, they are the classification results of ELM and SVM kernel function with all features, which are pass the PCA method. There are two kind modes of kernel function doing the comparison. One is Linear Kernel Function, and the other is Polynomial.

Mode of Kernel Function	Training Accuracy	Testing Accuracy
Linear	100%	68.33% , c = 0.001
Polynomial	100%	60% , c = 0.001

Table 1: The classification results of ELM Kernel Function

Mode of Kernel Function	**Training Accuracy**	**Testing Accuracy**
Linear	100%	68.33% , c = 0.001
Polynomial	100%	68.33% , c = 0.001

Table 2: The classification results of SVM

According to the tables, it is very clear that for the linear kernel mode, the performance of ELM and SVM are same. However, for polynomial, there is a noticeable difference between them, where the ELM performs better.

After comparing two methods with total data, the projects also need to find out how the method performances change when the method conditions and parameters are changed. No matter which model is used, SVM or ELM, there is a very important parameter C, which is a way to control over fitting. Higher values of C indicate that the model sets a harder margin on the classification, while smaller values of C indicate that the model sets a softer margin. This project used K-fold Cross-validation to get the suitable C values.

The purpose of using Cross-validation is to get reliable and stable models. The initial training data is divided into K groups, one separated group is kept aside to use as testing data, the other groups are used to do the training; repeat the cross-validation for K times and each group has been verified one time, then pick up the C value, which has the highest testing accuracy. The advantage of this method is that it repeats using randomly generated groups to do the training and validation and the accuracy of each model is evaluated after each run. The method used in this project is 5-fold cross-validation.

As theories described in Chapter 2, there is very important parameter, which can affect the classifying result directly, is the number of PCA components used in the classification process. The columns of the PCA method output matrix has already sorted from low-order to high order by how much important information of EEG signal is included. This project used the MATLAB function ‘pca’ to build up the PCA method

directly and picked up 1500, 3000, 4500, 6000, 7500, 9000 as the numbers of PCA components.

PCA components	ELM with linear kernel		SVM with linear kernel	
	C Value	Testing Accuracy	C Value	Testing Accuracy
1500	0.0534	46.67%	0.0027	45%
3000	0.0132	53.33%	0.0010	56.67%
4500	0.0100	60%	0.0010	65%
6000	0.0010	68.33%	0.0010	68.33%
7500	0.0010	68.33%	0.0010	68.33%
9000	0.0010	68.33%	0.0010	68.33%
PCA components	ELM with polynomial kernel		SVM with polynomial kernel	
	C Value	Testing Accuracy	C Value	Testing Accuracy
1500	0.0010	46.67%	0.0010	45%
3000	0.0010	56.67%	0.0010	56.67%
4500	0.0010	58.33%	0.0010	65%
6000	0.0010	60%	0.0010	68.33%
7500	0.0010	60%	0.0010	68.33%
9000	0.0010	60%	0.0010	68.33%

Table 3: The results of ELM and SVM with different PCA components and c values.

From Table 3, it can see that as the number of PCA components used in the process of classification, the testing accuracy keeps increasing until reach the highest testing accuracy, which are shown in Table 1 and Table 2. From Table 3, it looks like that 6000

features are enough to show the most characteristic of the differences between Class 1 and Class 2. According to the table, when the find the best C =0.001, C value is fixed by the system and do not change again. It can see from table that SVM find the best C value much faster than ELM and for polynomial kernel method, the testing accuracy of SVM is higher than the testing accuracy of ELM.

Chapter 5: Conclusion

The testing of the EEG signal classification of hand and feet movements is successful by using SVM and ELM. It is clear that compared with ELM, SVM has more stable performance and better accuracy on polynomial kernel method.

From the last section, the result is not perfect. The following are possible reasons:

- The amount of the training data is not enough to generate a classification model. For each class, there are only 100 samples. As the brain signal is not a simple and clear signal, only 50 samples cannot show the entire characteristic. Therefore, the accuracy, universality and applicability of the classification model are affected.
- The training data does not have universality so that the model does not work well with the test data. Training data is always limited, but this particular data is unlimited and very changeable. As the limitation of the training data, the model does not work well on the test data in this specific case.
- The dimensions of the classification are too high. During the processes of the model building, the feature space transforms to higher dimensions. As dimensions increase, the universality and applicability of the model will also decrease.

References

[1] Luis Fernando Nicolas-Alonso and Jaime Gomez-Gil, “Brain Computer Interfaces, a Review”

[2] Jason Brownlee, “A Tour of Machine Learning Algrithms” , on November 25,2013 http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

[3] W. J. G. v. d. Wassenberg, "Signal analysis techniques for evoked and event-related potentials," University of Groningen, Groningen, 2008.

[4] Corinna Cortes and Vladimir Vapnik, “Support Vector Networks”, 1995 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

[5] Carlos Guestrin, “SVMs, Duality and the Kernel Trick”, Carnegie Mellon University, http://www.cs.cmu.edu/~guestrin/Class/10701-S07/Slides/kernels.pdf

[6] N. Crstianini, Support Vector Machines and other kernel-based learning methods, Cambridge: Cambridge University Press, 2000.

[7] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks,” *2004 International Joint Conference on Neural Networks (IJCNN'2004)*, (Budapest, Hungary), July 25-29, 2004.

[8] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, “Extreme Learning Machine for Regression and Multiclass Classification,” *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 42, no. 2, pp. 513-529, 2012. (This paper shows that ELM generally outperforms SVM/LS-SVM in various kinds of cases.)

[9] Two class motor imagery (002-2014), http://bnci-horizon-2020.eu/database/data-sets

[10] ELM Official Website: http://www.ntu.edu.sg/home/egbhuang/

[11]G.-B. Huang, “Kernels, ”Cognitive *Computation*, vol. 6, pp. 376-390, 2014. (**Also briefing the differences and relationships between different methods such as SVM, LS-SVM, RVFL, Quick Net, Rosenblatt's Perceptron, etc**)

[12]"Support Vector Machines (SVM)," MATLAB, [Online]. Available: http://www.mathworks.com/help/stats/support-vector-machines-svm.html. [Accessed 13 November 2014].

[13]Batal, "PCA," [Online]. Available: http://people.cs.pitt.edu/~iyad/PCA.pdf [Accessed 13 November 2014].

Stroke Family: The Sensory Trigger Method: Making New Pathways for Speech After Stroke or Brain Injury

G. Huang, G.-B. Huang, S. Song, and K. You, “Trends in Extreme Learning Machines: A Review” *Neural Networks*, vol. 61, no. 1, pp. 32-48, 2015.

APPENDIX: CODE

```
% Build up the Training Data
clear;clc
load('S03T.mat')

Data_T= cell2mat(data);

Training_1 = Data_T(1).X;
Training_2 = Data_T(2).X;
Training_3 = Data_T(3).X;
Training_4 = Data_T(4).X;
Training_5 = Data_T(5).X;

Training_Trial1 = Data_T(1).trial;
trial1_length = length(Data_T(1).X);
Training_Trial2 = Data_T(2).trial;
trial2_length = length(Data_T(2).X);
Training_Trial3 = Data_T(3).trial;
trial3_length = length(Data_T(3).X);
Training_Trial4 = Data_T(4).trial;
trial4_length = length(Data_T(4).X);
Training_Trial5 = Data_T(5).trial;
trial5_length = length(Data_T(5).X);
Training_Trial =
[1,Training_Trial1,(Training_Trial2+Training_Trial1(20)),...

(Training_Trial3+Training_Trial1(20)+Training_Trial2(20))...

(Training_Trial4+Training_Trial1(20)+Training_Trial2(20)+Training_Trial
3(20)),...

(Training_Trial5+Training_Trial1(20)+Training_Trial2(20)+Training_Trial
3(20)+Training_Trial4(20))];

Training=[Training_1(1:Training_Trial1(20),:);Training_2(1:Training_Tri
al2(20),:);...

Training_3(1:Training_Trial3(20),:);Training_4(1:Training_Trial4(20),:)
;...
        Training_5(1:Training_Trial5(20),:)];
```

```
Mean_Training = mean(Training);
[row_Training,col_Training]= size(Training);
Mean_Training = repmat(Mean_Training,row_Training,1);
Var_Training = var(Training);
Var_Training = repmat(Var_Training ,row_Training,1);

Label_Training = [Data_T(1).y Data_T(2).y Data_T(3).y Data_T(4).y
Data_T(5).y];

Training = Training- Mean_Training;
Training = Training./sqrt(Var_Training);
T_0=1;

Training_Hand=[];
Training_Feet=[];
Row_T = [];

a_idx=1;
b_idx=1;

Training_Label_C=[];

for i = 1:100

    label = Label_Training(i);

    T = Training_Trial(i+1)-Training_Trial(i);
    Row_T(i) = T;
    if label == 1;
        Training_Data(i,1:(T+1),:)
=Training(Training_Trial(i):Training_Trial(1+i),:);
        T_0 = T+T_0;
     elseif label == 2;
        Training_Data(i,1:(T+1),:) =
Training(Training_Trial(i):Training_Trial(1+i),:);
        T_0 = T+T_0;
    end

    for chanNum=1:15
        PSD_Train(i,:,chanNum)=pwelch(Training_Data(i,:,chanNum));
```

```
    end
   i = i+1;
end


%Build up the Test Data
load('S03E.mat')

Data_E = cell2mat(data);

Testing_1=Data_E(1).X;
Testing_2=Data_E(1).X;
Testing_3=Data_E(1).X;

Testing_Trial1 = Data_E(1).trial;
trial1_length= length(Data_E(1).X);
Testing_Trial2 = Data_E(2).trial;
trial2_length = length(Data_E(2).X);
Testing_Trial3 = Data_E(3).trial;
trial3_length = length(Data_E(3).X);

Testing_Trial = [1,Testing_Trial1,(Testing_Trial2+Testing_Trial1(20)),...

(Testing_Trial3+Testing_Trial1(20)+Testing_Trial2(20))];...

Testing=[Testing_1(1:Testing_Trial1(20),:);Testing_2(1:Testing_Trial2(2
0),:);...
         Testing_3(1:Testing_Trial3(20),:)];

Mean_Testing = mean(Testing);
[row_Testing,col_Testing]= size(Testing);
Mean_Testing= repmat(Mean_Testing,row_Testing,1);
Var_Testing = var(Testing);
Var_Testing = repmat(Var_Testing ,row_Testing,1);

label_testing = [Data_E(1).y,Data_E(2).y,Data_E(3).y];

Testing = Testing- Mean_Testing;
Testing = Testing./sqrt(Var_Testing);

E_0 =1;
```

```
Testing_Data=[];
Row_E = [];

a_idx=1;
b_idx=1;

    for i = 1:1:60

        label = label_testing(i);
        E = Testing_Trial(i+1)-Testing_Trial(i);
        Row_E = E;

        if label == 1;
            Testing_Data(i,1:(E+1),:)
=Testing(Testing_Trial(i):Testing_Trial(1+i),:);
            E_0 = E+E_0;
        elseif label == 2;
            Testing_Data(i,1:(E+1),:) =
Testing(Testing_Trial(i):Testing_Trial(1+i),:);
            E_0 = E+E_0;
        end

        for chanNum=1:15
            PSD_Test(i,:,chanNum)=pwelch(Testing_Data(i,:,chanNum));
        end

        i = i+1;
    end




%% Training_test
Combine_Training_Data = Training_Data;
Combine_classes = Label_Training;

Fs = 512;
Nsec=256;
sigLength = length(Combine_Training_Data);
```

```
% n=0:sigLength-1;t=n/Fs;
% FFT_Combine_Training_data =
abs(fft(Combine_Training_Data,[],2)).^2/Nsec;

PCA_features_Training = [];
PCA_features_Training = [];
New_slice = [];
Q = 15;
for row = 1:1:100
    New_slice(:,:) =
PSD_Train(row,:,:);%FFT_Combine_Training_data(row,:,:);
    PCA_channel_Training = pca(New_slice);
   [row_feature,col_feature] = size(PCA_channel_Training);
    PCA_features_Training(row,:)=PCA_channel_Training(:);
end

Label_Training = Label_Training(:);

for i = 1:1:100

    Class_PCA_Training(i,:) = [Label_Training(i),
PCA_features_Training(i,:)];


end

%% Testing_data
Test_data = Testing_Data;
Test_class = label_testing;

Fs = 512;
Nsec=256;
sigLength = length(Testing_Data);
% n=0:sigLength-1;t=n/Fs;
% FFT_Combine_Test_data = abs(fft(Testing_Data,[],2)).^2/Nsec;
% FFT_Combine_Testing_data_plot = mean(FFT_Combine_Test_data);

New_slice = [];

for row = 1:1:60
    New_slice(:,:) = PSD_Test(row,:,:);%FFT_Combine_Test_data(row,:,:);
```

```
    PCA_channel_Testing = pca(New_slice);
   [row_feature,col_feature] = size(PCA_channel_Testing);
    PCA_features_Testing(row,:)=PCA_channel_Testing(:);
end
Label_Testing= label_testing';
for i = 1:1:60
    Class_PCA_Test(i,:) = [Label_Testing(i), PCA_features_Testing(i,:)];
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:10
    PCA_features_Training=PSD_Train(:,1:(i*100),:);
    PCA_features_Testing=PSD_Test(:,1:(i*100),:);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[c,a,b] = size(PCA_features_Training(1,:,:));
Training_Data = reshape(PCA_features_Training,100,a*b);
Testing_Data = reshape(PCA_features_Testing,60,a*b);
cSweepValues=logspace(-3,0,100);%;linspace(0.012,0.015,1000);

csvwrite('Class_PCA_Training.csv',[Label_Training Training_Data])
csvwrite('Class_PCA_Test.csv',[Label_Testing Testing_Data]);

[ MaxAccuracy_Line_ELM,MaxCVal_Line_ELM,svm_struct_Line_ELM ] =
CrossValidation( [Training_Data Label_Training],5,cSweepValues,'ELM',
'lin_kernel');
[ MaxAccuracy_Poly_ELM,MaxCVal_Poly_ELM,svm_struct_Poly_ELM ] =
CrossValidation( [Training_Data Label_Training],5,cSweepValues,'ELM',
'poly_kernel');
[
MaxAccuracy_Line_ELM,MaxCVal_Line_ELM,i;[MaxAccuracy_Poly_ELM,MaxCVal_P
oly_ELM,i]]
end


% i = 1;
% for c_val= [1:10:1000]./100%1:5:500
%
%
%    c(i) = c_val;
```

```
%    n = 1;
%     for K_pa = 1:10:100
%       [TrainingTime, TestingTime,TrainingAccuracy(n),
TestingAccuracy(n)] =
elm_kernel('Class_PCA_Training.csv','Class_PCA_Test.csv', 1, c_val,
'lin_kernel',Mean_Training);
%    end
%    Accuracy(i) = TestingAccuracy;
%    i = 1+i;
% end
% %
%    figure
%    plot(c,Accuracy)
%
%
% c = [];
% Accuracy = [];
% i = 1
% for c_val=logspace(-5,-3,0)
%
%
%    c(i) = c_val
%    n = 1;
%    %for K_pa = 1:10:100
%       [TrainingTime, TestingTime,TrainingAccuracy, TestingAccuracy] =
elm_kernel('Class_PCA_Training.csv','Class_PCA_Test.csv', 1, c_val,
'poly_kernel',100);
%    %end
%    Accuracy = max(TestingAccuracy);
%    i = 1+i;
% end

%%%%%%########################################
% Using freq components as the features instead of the PCA outputs
% FFT_Combine_Training_data=FFT_Combine_Training_data(:,1:1000,:);
% FFT_Combine_Test_data=FFT_Combine_Test_data(:,1:1000,:);
% PCA_features_Training=reshape(FFT_Combine_Training_data,100,1000*15);
% PCA_features_Testing=reshape(FFT_Combine_Test_data,60,1000*15);
%%%%%%########################################

%Combine_Training_Data=Combine_Training_Data
```

```
% PCA_features_Training=[];
% PCA_features_Testing=[];
% for i=1:15
%    PCA_features_Training=[PCA_features_Training
Combine_Training_Data(:,:,i)];
%    PCA_features_Testing=[PCA_features_Testing Test_data(:,:,i)];
% end
%%%%%########################################
%  PCA_features_Training=PSD_Train;
%  PCA_features_Testing=PSD_Test;
%%%%%########################################

%
%% SVM
clear C_val Accuracy_Training Accuracy_Testing;
 n = 1;
for i=1:10
   PCA_features_Training=PSD_Train(:,1:(i*100),:);
   PCA_features_Testing=PSD_Test(:,1:(i*100),:);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[c,a,b] = size(PCA_features_Training(1,:,:));
Training_Data = reshape(PCA_features_Training,100,a*b);
Testing_Data = reshape(PCA_features_Testing,60,a*b);
cSweepValues=logspace(-3,0,100);%;linspace(0.012,0.015,1000);

csvwrite('Class_PCA_Training.csv',[Label_Training Training_Data])
csvwrite('Class_PCA_Test.csv',[Label_Testing Testing_Data]);
[ MaxAccuracy_Line,MaxCVal_Line,svm_struct_Line ] = CrossValidation(
[Training_Data Label_Training],5,cSweepValues,'SVM', 'linear');
%[ MaxAccuracy_Quad,MaxCVal_Quad,svm_struct_Quad ] = CrossValidation(
[Training_Data Label_Training],5,cSweepValues,'SVM', 'quadratic');
[ MaxAccuracy_Poly,MaxCVal_Poly,svm_struct_Poly ] = CrossValidation(
[Training_Data Label_Training],5,cSweepValues,'SVM', 'polynomial');

Group_Testing = svmclassify(svm_struct_Line,Testing_Data);%(1:20,:));
Correct_Classification_Testing = (Group_Testing==Label_Testing);%(1:20));
Accuracy_Testing_Line_SVM = (sum(Correct_Classification_Testing))/60;%20;


Group_Testing_2 = svmclassify(svm_struct_Poly,Testing_Data);%(1:20,:));
```

```
Correct_Classification_Testing_2 =
(Group_Testing==Label_Testing);%(1:20));
Accuracy_Testing_Poly_SVM =
(sum(Correct_Classification_Testing_2))/60;%20;

result(i,:) =
[MaxCVal_Line,Accuracy_Testing_Line_SVM,MaxCVal_Poly,Accuracy_Testing_P
oly_SVM]

end

function [TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy,TY]
= elm_kernel(TrainingData_File, TestingData_File, Elm_Type,
Regularization_coefficient, Kernel_type, Kernel_para)

% Usage: elm(TrainingData_File, TestingData_File, Elm_Type,
NumberofHiddenNeurons, ActivationFunction)
% OR:    [TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy] =
elm(TrainingData_File, TestingData_File, Elm_Type, NumberofHiddenNeurons,
ActivationFunction)
%
% Input:
% TrainingData_File          - Filename of training data set
% TestingData_File           - Filename of testing data set
% Elm_Type                  - 0 for regression; 1 for (both binary and
multi-classes) classification
% Regularization_coefficient  - Regularization coefficient C
% Kernel_type               - Type of Kernels:
%                              'RBF_kernel' for RBF Kernel
%                              'lin_kernel' for Linear Kernel
%                              'poly_kernel' for Polynomial Kernel
%                              'wav_kernel' for Wavelet Kernel
%Kernel_para                - A number or vector of Kernel Parameters. eg.
1, [0.1,10]...
% Output:
% TrainingTime              - Time (seconds) spent on training ELM
% TestingTime               - Time (seconds) spent on predicting ALL testing
data
% TrainingAccuracy           - Training accuracy:
%                            RMSE for regression or correct classification
rate for classification
```

```
% TestingAccuracy           - Testing accuracy:
%                          RMSE for regression or correct classification
rate for classification
%
% MULTI-CLASSE CLASSIFICATION: NUMBER OF OUTPUT NEURONS WILL BE AUTOMATICALLY
SET EQUAL TO NUMBER OF CLASSES
% FOR EXAMPLE, if there are 7 classes in all, there will have 7 output
% neurons; neuron 5 has the highest output means input belongs to 5-th class
%
% Sample1 regression: [TrainingTime, TestingTime, TrainingAccuracy,
TestingAccuracy] = elm_kernel('sinc_train', 'sinc_test', 0, 1,
''RBF_kernel',100)
% Sample2 classification: elm_kernel('diabetes_train', 'diabetes_test', 1,
1, 'RBF_kernel',100)
%
    %%%%    Authors:    MR HONG-MING ZHOU AND DR GUANG-BIN HUANG
    %%%%    NANYANG TECHNOLOGICAL UNIVERSITY, SINGAPORE
    %%%%    EMAIL:      EGBHUANG@NTU.EDU.SG; GBHUANG@IEEE.ORG
    %%%%    WEBSITE:    http://www.ntu.edu.sg/eee/icis/cv/egbhuang.htm
    %%%%    DATE:       MARCH 2012

%%%%%%%%%%% Macro definition
REGRESSION=0;
CLASSIFIER=1;

%%%%%%%%%%% Load training dataset
train_data=load(TrainingData_File);
T=train_data(:,1)';
P=train_data(:,2:size(train_data,2))';
clear train_data;                                   %   Release raw training
data array

%%%%%%%%%%% Load testing dataset
test_data=load(TestingData_File);
TV.T=test_data(:,1)';
TV.P=test_data(:,2:size(test_data,2))';
clear test_data;                                    %   Release raw testing data
array

C = Regularization_coefficient;
NumberofTrainingData=size(P,2);
```

```
NumberofTestingData=size(TV.P,2);

if Elm_Type~=REGRESSION
    %%%%%%%%%%%% Preprocessing the data of classification
    sorted_target=sort(cat(2,T,TV.T),2);
    label=zeros(1,1);                               %   Find and save in 'label'
class label from training and testing data sets
    label(1,1)=sorted_target(1,1);
    j=1;
    for i = 2:(NumberofTrainingData+NumberofTestingData)
        if sorted_target(1,i) ~= label(1,j)
            j=j+1;
            label(1,j) = sorted_target(1,i);
        end
    end
    number_class=j;
    NumberofOutputNeurons=number_class;

    %%%%%%%%%% Processing the targets of training
    temp_T=zeros(NumberofOutputNeurons, NumberofTrainingData);
    for i = 1:NumberofTrainingData
        for j = 1:number_class
            if label(1,j) == T(1,i)
                break;
            end
        end
        temp_T(j,i)=1;
    end
    T=temp_T*2-1;

    %%%%%%%%%% Processing the targets of testing
    temp_TV_T=zeros(NumberofOutputNeurons, NumberofTestingData);
    for i = 1:NumberofTestingData
        for j = 1:number_class
            if label(1,j) == TV.T(1,i)
                break;
            end
        end
        temp_TV_T(j,i)=1;
    end
    TV.T=temp_TV_T*2-1;
```

```
                                        %   end if of Elm_Type
end

%%%%%%%%%%% Training Phase %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic;
n = size(T,2);
%Omega_train = kernel_matrix(P',Kernel_type, Kernel_para);
Omega_train = kernel_matrix(P',Kernel_type, Kernel_para);
OutputWeight=((Omega_train+speye(n)/C)\(T'));
TrainingTime=toc;

%%%%%%%%%%% Calculate the training output
Y=(Omega_train * OutputWeight)';                            %   Y: the actual
output of the training data

%%%%%%%%%%% Calculate the output of testing input
tic;
Omega_test = kernel_matrix(P',Kernel_type, Kernel_para,TV.P');
TY=(Omega_test' * OutputWeight)';                           %   TY: the actual
output of the testing data
TestingTime=toc;
clear C
clear TV.P
%%%%%%%%%% Calculate training & testing classification accuracy

if Elm_Type == REGRESSION
%%%%%%%%%% Calculate training & testing accuracy (RMSE) for regression case
    TrainingAccuracy=sqrt(mse(T - Y));
    TestingAccuracy=sqrt(mse(TV.T - TY))  ;
end

if Elm_Type == CLASSIFIER
%%%%%%%%%% Calculate training & testing classification accuracy
    MissClassificationRate_Training=0;
    MissClassificationRate_Testing=0;

    for i = 1 : size(T, 2)
        [x, label_index_expected]=max(T(:,i));
        [x, label_index_actual]=max(Y(:,i));
        if label_index_actual~=label_index_expected
```

```
MissClassificationRate_Training=MissClassificationRate_Training+1;
        end
    end
    TrainingAccuracy=1-MissClassificationRate_Training/size(T,2)  ;
    for i = 1 : size(TV.T, 2)
        [x, label_index_expected]=max(TV.T(:,i));
        [x, label_index_actual]=max(TY(:,i));
        if label_index_actual~=label_index_expected

MissClassificationRate_Testing=MissClassificationRate_Testing+1;
        end
    end
    TestingAccuracy=1-MissClassificationRate_Testing/size(TV.T,2) ;
end


%%%%%%%%%%%%%%%%%% Kernel Matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function omega = kernel_matrix(Xtrain,kernel_type, kernel_pars,Xt)

nb_data = size(Xtrain,1);


if strcmp(kernel_type,'RBF_kernel'),
    if nargin<4,
        XXh = sum(Xtrain.^2,2)*ones(1,nb_data);
        omega = XXh+XXh'-2*(Xtrain*Xtrain');
        omega = exp(-omega./kernel_pars(1));
    else
        XXh1 = sum(Xtrain.^2,2)*ones(1,size(Xt,1));
        XXh2 = sum(Xt.^2,2)*ones(1,nb_data);
        omega = XXh1+XXh2' - 2*Xtrain*Xt';
        omega = exp(-omega./kernel_pars(1));
    end

elseif strcmp(kernel_type,'lin_kernel')
    if nargin<4,
        omega = Xtrain*Xtrain';
    else
        omega = Xtrain*Xt';
```

```
    end

elseif strcmp(kernel_type,'poly_kernel')
    if nargin<4,
        omega = (Xtrain*Xtrain'+kernel_pars(1)).^kernel_pars(2);
    else
        omega = (Xtrain*Xt'+kernel_pars(1)).^kernel_pars(2);
    end

elseif strcmp(kernel_type,'wav_kernel')
    if nargin<4,
        XXh = sum(Xtrain.^2,2)*ones(1,nb_data);
        omega = XXh+XXh'-2*(Xtrain*Xtrain');

        XXh1 = sum(Xtrain,2)*ones(1,nb_data);
        omega1 = XXh1-XXh1';
        omega =
cos(kernel_pars(3)*omega1./kernel_pars(2)).*exp(-omega./kernel_pars(1))
;

    else
        XXh1 = sum(Xtrain.^2,2)*ones(1,size(Xt,1));
        XXh2 = sum(Xt.^2,2)*ones(1,nb_data);
        omega = XXh1+XXh2' - 2*(Xtrain*Xt');

        XXh11 = sum(Xtrain,2)*ones(1,size(Xt,1));
        XXh22 = sum(Xt,2)*ones(1,nb_data);
        omega1 = XXh11-XXh22';

        omega =
cos(kernel_pars(3)*omega1./kernel_pars(2)).*exp(-omega./kernel_pars(1))
;
    end
    clear omega
end

function [ MaxAccuracy,MaxCVal,MachineLearningsModel ] = CrossValidation(
TotalData,NumberOfSections,cSweepValues,MachineLearningAlgorithm,
KernelFunction)
%UNTITLED Summary of this function goes here
%   TotalData - [TrainingData, TrainingLabels]
```

```
%   NumberOfSections - Number of sections to break TotalData into to
%                   perform Leave-One-Out cross- validation
%   cSweepValues - The vector of values that c will sweep when searching
%                for the best c value
%   MachineLearningAlgorithm - Either 'SVM' to perform the learning using
%                           SVM or 'ELM' to perform the learning using
%                           ELM.
%   KernelFunction -
%                    For SVM:
%                    For ELM:    'lin_kernel'
    [rw, cm]=size(TotalData);

    for DataIdx=1:NumberOfSections
        StartIdx=(rw/NumberOfSections)*(DataIdx-1)+1;
        StopIdx=(rw/NumberOfSections)*(DataIdx);

        TestDatasetIdx=StartIdx:StopIdx;
        TrainDatasetIdx=setdiff(1:rw,TestDatasetIdx);
        TrainDataset=TotalData(TrainDatasetIdx,:);
        TestDataset=TotalData(TestDatasetIdx,:);
        [rw_train, cm_test]=size(TrainDataset);
        [rw_test,cm_train]=size(TestDataset);

        tmpTrainFile='Class_PCA_Training.csv';
        tmpTestFile='Class_PCA_Test.csv';

        Mean_Training = mean(TrainDataset);
        [row_Training,col_Training]= size(TrainDataset);
        Mean_Training = repmat(Mean_Training,row_Training,1);

        %Perform c sweep here
        for c= cSweepValues
            n=find(cSweepValues==c);
            if strcmp(MachineLearningAlgorithm,'SVM')
                MachineLearningsModel =
svmtrain(TrainDataset(:,1:cm_train-1),TrainDataset(:,cm_train),'kernel_
function',KernelFunction,'boxconstraint',c,'method','LS');
                Group_Testing =
svmclassify(MachineLearningsModel,TestDataset(:,1:cm_test-1));
                Correct_Classification_Testing =
(Group_Testing==TestDataset(:,cm_test));
```

```
            Accuracy(n) =
(sum(Correct_Classification_Testing))/rw_test;
          elseif strcmp(MachineLearningAlgorithm,'ELM')
            [TrainingTime, TestingTime,TestingAccuracy(n), Accuracy(n)]
= elm_kernel(tmpTrainFile,tmpTestFile, 1, c,
KernelFunction,Mean_Training);
            MachineLearningsModel=-1;
          end
       end
       PeakAccuracy(DataIdx)=max(Accuracy);

PeakCVal(DataIdx)=min(cSweepValues(Accuracy==PeakAccuracy(DataIdx)));
    end
    MaxAccuracy=max(PeakAccuracy);
    MaxCVal=min(PeakCVal(MaxAccuracy==PeakAccuracy));

end
```