2010

A Study of recent classification algorithms and a novel approach for biosignal data classification

Eyup Cinar

A STUDY OF RECENT CLASSIFICATION ALGORITHMS AND A NOVEL APPROACH FOR BIOSIGNAL DATA CLASSIFICATION

By

EYUP CINAR

Thesis Submitted to the Faculty of Rochester Institute of Technology in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

Approved by

Thesis Advisor	**Dr. Ferat Sahin**	____________________
Thesis Committee	**Dr. Daniel Phillips**	____________________
Thesis Committee	**Dr. Athimoottil Mathew**	____________________
Department Head	**Dr. Sohail A. Dianat**	____________________

ROCHESTER INSTITUTE OF TECHNOLOGY

KATE GLEASON COLLEGE OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND MICROELECTRONICS ENGINEERING,

ROCHESTER, NEW YORK

JUNE 2010

A STUDY OF RECENT CLASSIFICATION ALGORITHMS AND A NOVEL APPROACH FOR BIOSIGNAL DATA CLASSIFICATION

Acknowledgments

I am very grateful to my advisor Dr. Ferat Sahin for his generous support in my research and academic life as well as the friendship developed over the past two years at RIT. I am thankful to Dr. Daniel Phillips and Dr. Athimoottil Mathew for accepting to be a member of my thesis committee and for their significant reviews and ideas for my thesis. I owe many thanks to my family members for their endless motivation and encouragement throughout my studies abroad. I am also very thankful to my colleagues Ryan Bowen, Ticiano Torres-Peralta, and Vince Baier for their support for my research and their close friendships throughout my student life at RIT. Last but not least I acknowledge the Fulbright commission of Turkey for providing me all the necessary support to pursue my Master's degree in the USA.

List of Publications

Below is the list of publications resulted from this thesis work.

[1] E. Cinar and F. Sahin, “A Study of Recent Classification Algorithms and a Novel Approach for EEG Data Classification,” in *IEEE 2010 International Conference on Systems, Man and Cybernetics*, October 2010.

[2] E. Cinar and F. Sahin, “EOG Controlled Mobile Robot Using Radial Basis Function Networks,” in *IEEE Fifth International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control (ICSCCW 2009)*, September 2009.

[3] R. Bowen, E. Cinar, and F. Sahin, “System of Systems Approach to a Human Tracking Problem with Mobile Robots using a Single Security Camera,” in *IEEE Fifth International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control (ICSCCW 2009)*, September 2009.

Abstract

Analyzing and understanding human biosignals have been important research areas that have many practical applications in everyday life. For example, Brain Computer Interface is a research area that studies the connection between the human brain and external systems by processing and learning the brain signals called *Electroenceplograhpy* (EEG) signals. Similarly, various assistive robotics applications are being developed to interpret eye or muscle signals in humans in order to provide control inputs for external devices. The efficiency for all of these applications depends heavily on being able to process and classify human biosignals. Therefore many techniques from Signal Processing and Machine Learning fields are applied in order to understand human biosignals better and increase the efficiency and success of these applications.

This thesis proposes a new classifier for biosignal data classification utilizing Particle Swarm Optimization Clustering and Radial Basis Function Networks (RBFN). The performance of the proposed classifier together with several variations in the technique is analyzed by utilizing comparisons with the state of the art classifiers such as Fuzzy Functions Support Vector Machines (FFSVM), Improved Fuzzy Functions Support Vector Machines (IFFSVM). These classifiers are implemented on the classification of same biological signals in order to evaluate the proposed technique. Several clustering algorithms, which are used in these classifiers, such as *K*-means, Fuzzy *c*-means, and Particle Swarm Optimization (PSO), are studied and compared with each other based on clustering abilities. The effects of the analyzed clustering algorithms in the performance of Radial Basis Functions Networks classifier are investigated. Strengths and weaknesses are analyzed on various standard and EEG datasets. Results show that the proposed classifier that combines PSO clustering with RBFN classifier can reach or exceed the performance of these state of the art classifiers. Finally, the proposed classification technique is applied to a real-time system application where a mobile robot is controlled based on person's EEG signal.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Motivation

The human body generates several electrical signals that can be studied and analyzed to infer some meaningful information for various applications. For example, human eyes generate a signal called Electroocculogram (EOG) that could be used to determine the position of the eyeball. The firing of many neurons due to neurological activities in the brain generates Electroencephalogram (EEG) signals that might be helpful to decipher human thoughts or intents. Similarly, the contraction of body muscles generates Electromyogram (EMG) signals that could be used for the development of assistive devices.

With the advances in biomedical engineering and sensor technology, the analysis of human bio-potential signals became a crucial research area. Researchers have been studying to understand and classify human biosignals in order to provide various opportunities to people as in developing assistive technologies for disabled people or obtaining more accurate diagnosis of diseases.

For example, Greene et al. [1] uses the EOG signal measurements of schizophrenic patients and tries to detect the disease utilizing saccade motions of the patients against a visual stimulus. According to [1], these saccade motions of the eye can help detect the Schizophrenia disease in humans. Suetsugu et al. [2] utilize the EOG signals in order to control a disabled person's forearm by applying electrical stimulus to the arm muscles. Another study discusses generating control inputs for a wheelchair by analyzing the human EOG signals and extracting the directional information from the eyes so that the mobility of the severely disabled people might be increased [3].

In addition to EOG signals, examining the EEG signals might also help people by developing assistive technologies for disabled people or obtaining more accurate diagnosis of brain related abnormal activities. Lotte et al. surveys over 80 papers in terms of EEG classification techniques for Brain Computer Interface applications [4]. In addition, Yuge et al. [5] studies the effects of alcoholism on the EEG signals by extracting the power spectrum characteristics of the EEG signal collected from the alcoholics in order to detect alcoholism on the patients.

As these studies [1-5] imply, the biological signals in human body might be effectively used to make the human life easier through several applications. These types of studies have contributed to new and rapidly developing scientific areas such as Cybernetics and Biorobotics [6].

According to Wiener, “Cybernetics is the science of control and communication in the animal and the machine.” It merges humans and machines by utilizing intelligent tools in order to build new systems that might make the human life easier [6]. On the other hand, Biorobotics is a field that studies biological beings and how to mimic them to design new mechanical devices [7]. The term is also defined as a subfield of robotics that studies biological beings to be a part of robots. Considering the latter definition of Biorobotics and the definition of Cybernetics, we can conclude that the main focus of these two fields is to develop a technology that might produce inputs for the control of external devices and provide the interface between machines and the human body.

In order to understand biological signals so that they could be used as inputs for the external devices, one should go through several processes and make important considerations on several components as shown in Figure 1. These components can be grouped into three main categories: Processing, Feature Extraction, and Classification. The preprocessing is the phase of preparing the data in order to remove undesired components in the signal that might be considered as artifacts such as the eye blinks for the EEG signal. The second component is the Feature Extraction that is extracting the features from the signal that might be discrimitive enough for the human or machine to differentiate the signal into the different classes. The third component is classifying the extracted features and determining the behavior, action, or thought which causes the generation of the biopotential signal.

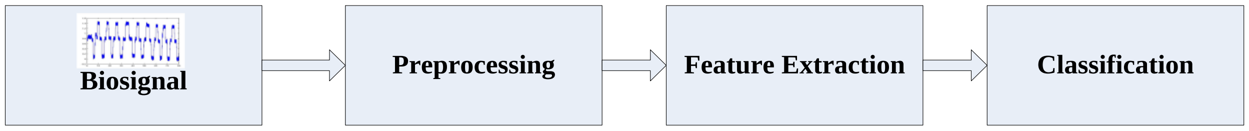


Figure 1 Components of Processing Biosignals

The main focus of this study is to contribute to the fields of Biorobotics and Cybernetics by analyzing the strengths and weaknesses of the current state of the art classification and feature extraction techniques, developing new and efficient ones that might increase the performance of robust decision making, and applying these into the real world applications.

The following section includes a brief introduction on theoretical topics. It explains the nature of the main biological signals examined in this study such as EOG and EEG and the important characteristics of these signals that could be used for the real world applications.

1.2 Biological Signals and Their Characteristics:

Various electrical activities occurring in human body generate several signals called human biopotentials. These signals can be recorded by utilizing special data acquisition devices and interpreted by various processing and classification techniques to infer meaningful information.

In a very small scale, a biological ionic current is created by different polarization of specific ions in a nerve cell such as sodium, potassium, and chloride. Considering the resistance of cell membranes, the ionic current creates the electrical potential called a biopotential. There are four types of human biopotential signals that have been studied by researchers. These are Electrooculogram (EOG), Electromygram (EMG), Electrocardiogram (EKG), and Electroencephalogram (EEG). The next two sections describe the EOG and EEG signals as they are used in the applications described in this thesis.

1.2.1 Electroocculogram (EOG) Signals

The EOG signals are the biopotential signals generated by human eyes. Although there are many theories about how the EOG signal is generated, the highly accepted theory is cornea-retinal dipole theory [8]. According to this theory, the eye ball is polarized like a dipole as illustrated in Figure 2.

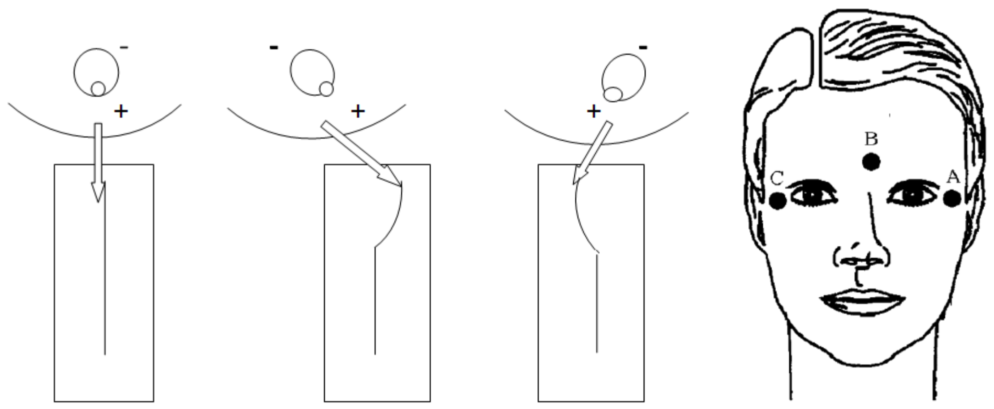


Figure 2 Polarization of the Eye Ball and an Example Electrode Placement for EOG Signal Collection [8]

Movement of the eye ball by the eye muscles changes the orientation of this dipole and generates the EOG signal. This signal can be measured by the special electrodes placed on the specific locations shown in Figure 2. While measuring the EOG signal, one electrode should act as reference for the rest of the electrodes. This is usually selected away from the eyes and on the forehead such as the location B in Figure 2.

In this study we use a data acquisition instrument called Bioradio 150 by Clevemed (Cleveland, Ohio) ® for EOG signal collection. Figure 3 shows the raw EOG signal collected by this device and plotted in Matlab. The electrodes are placed based on the configuration in Figure 2.

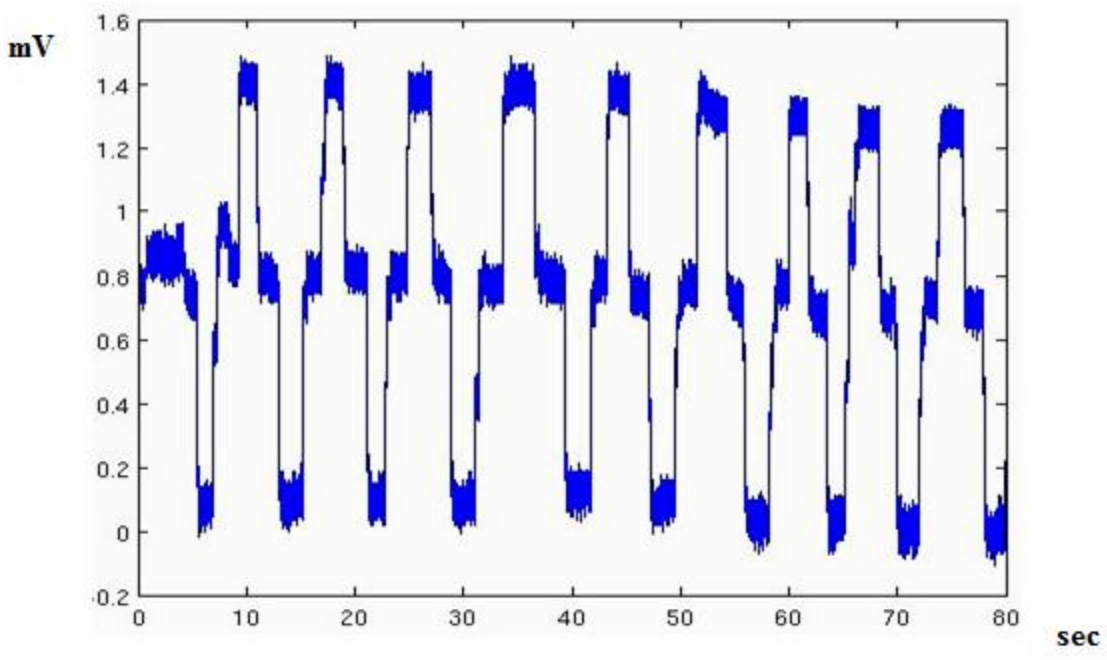


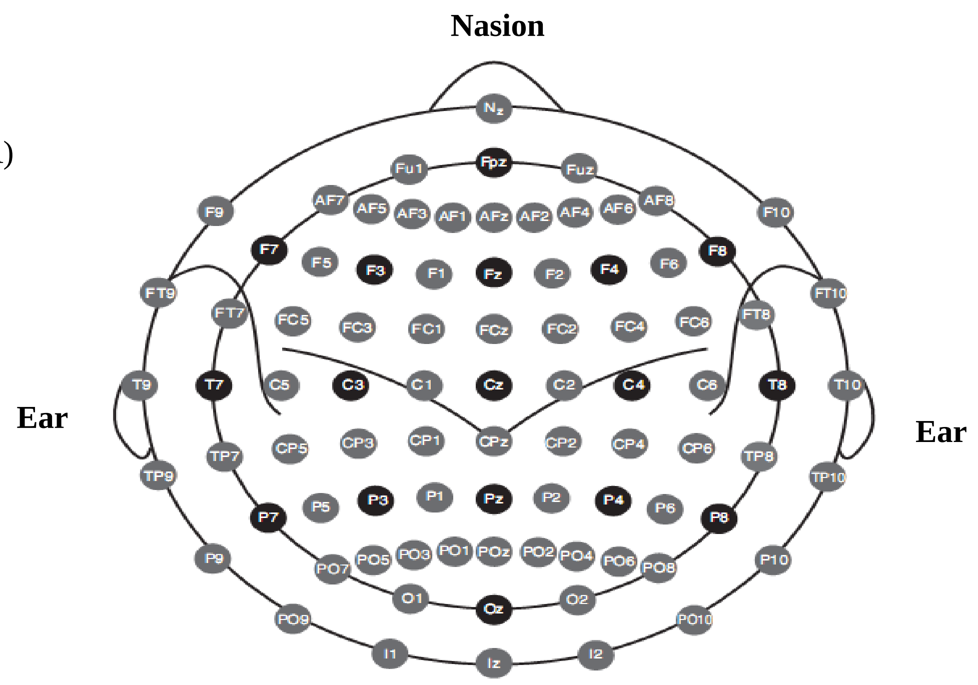
Figure 3 Raw EOG data collected by Clevemed BioRadio 150 Data Acquisition Device as a Result of Repeated Rye Movement in a Left/Right Fashion by a Volunteer

1.2.2 Electroencephalogram (EEG) Signals:

EEG signals are biopotentials recorded on the scalp, generated by the firing of the neurons in the brain. It has been known that specific tasks in the human body are controlled by specific parts of the brain and generated neurologic electrical activity is enough to be measured by the electrodes placed on these specific parts of the brain [9], [10]. The recorded EEG signal by this method is within 1-100 µV amplitude range and may contain useful information to decipher human thoughts or intents. This information can be converted into control inputs for various systems such as BCI-based assistive devices or detection systems for brain-related abnormal activities [9]-[14].

In order to collect EEG signals, electrodes are placed according to the standard International 10-20 system shown in Figure 4 (A). Each electrode is named with a letter to identify the brain region and a number to identify the hemispheric location. For example, the letter *F* indicates that the electrode corresponds to the *Front* region of the brain. The odd numbers refer to the left hemisphere and even numbers refer to the right hemisphere of the brain [9].

(A)



(B)

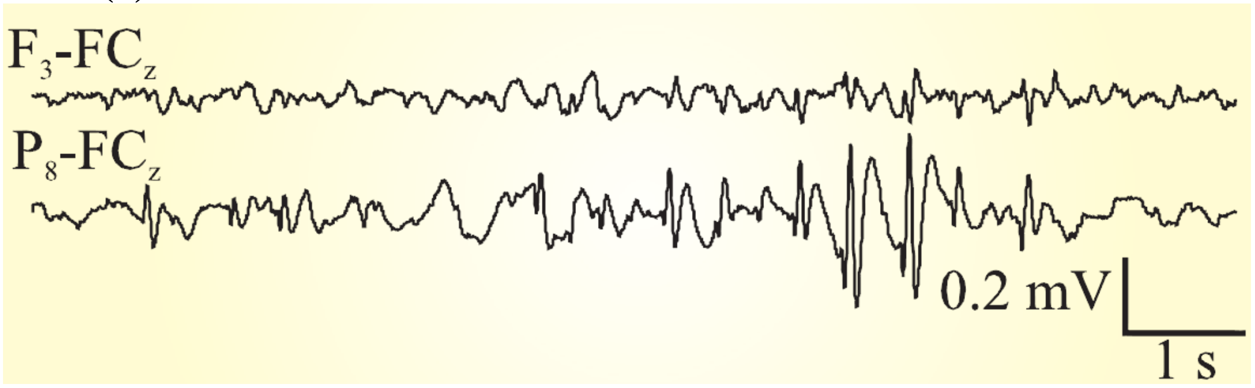


Figure 4 (A) International 10-20 EEG Electrode Placement System. (B) Two EEG Traces Example with a Burst of Epileptiform Activity on the Posterior Right Side. P_8-FC_z represents subtraction of the signal FC_z from electrode P_8 [9] .

Figure 4 (B) represents an example of two EEG traces recorded from the human scalp for a 10 second epoch. It includes a burst of epileptiform activity at the end of the trace within the signal recorded between P_8 and FC_z.

Certain frequency ranges of EEG signals have specific biological significance [10]. These typical frequency ranges are named with Greek letter band names such as *Alpha*, *Beta*, *Delta*, and *Theta*. It is known that some of the tasks controlled by the brain are more evident within specific frequency bands [10]-[12]. Table 1 includes these standard frequency ranges of EEG signals, their names, and examples of the bands in which specific actions are known to be more evident.

Table 1 Specific EEG Signal Frequency Ranges and Associated Real World Actions

Frequency Bands	**Evident Actions**
Delta **[0.5-4 Hz]**	Sleep waves in adults
Theta **[4-8 Hz]**	consciousness slips towards drowsiness, deep mediation
Alpha **[8-13Hz]**	Relaxed awareness without any attention and concentration
Beta **[13-30 Hz]**	Active thinking, attention

Another important phenomenon with the EEG signals is known to be *Mu rhythm* or *Event Related Desynchronization* (ERD) as shown in Figure 5.

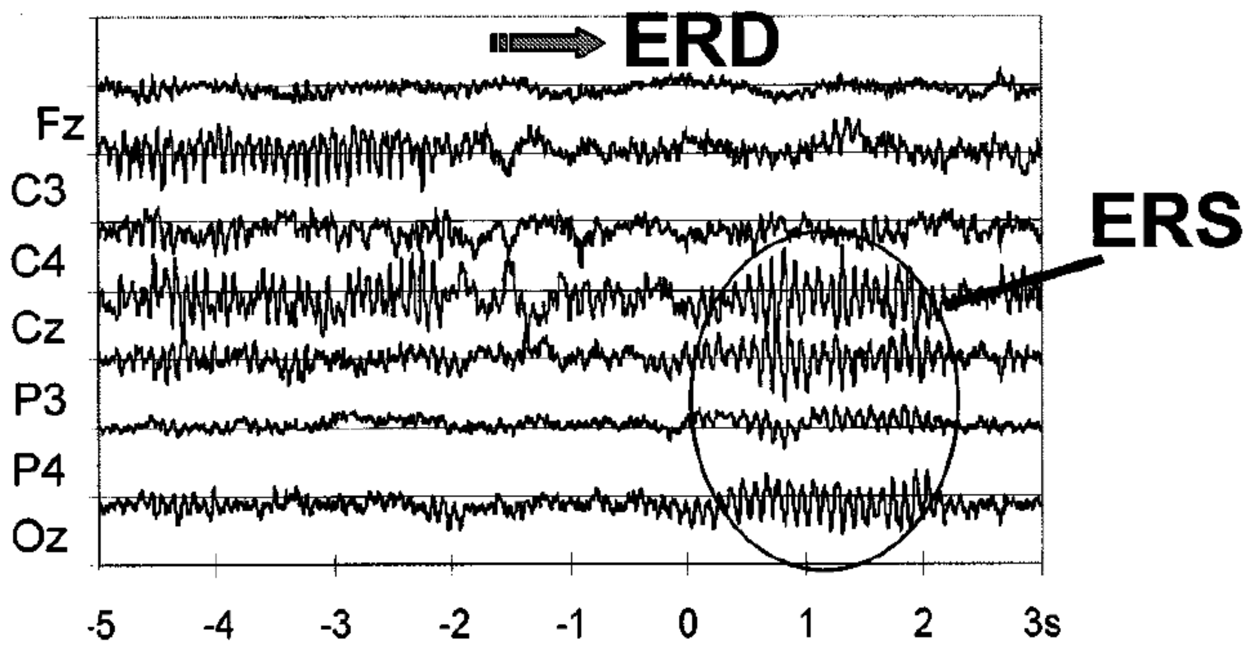


Figure 5 Mu Rhythm, Event Related Desynchronization, and Synchronization [11]

This event is a characteristic attenuation in the power of EEG signal in certain frequency ranges due to motor action preparation by the sensorimotor cortex area of the brain [11]. The Sensorimotor cortex area is located in the central lobe of the brain and manages motor actions of the body such as moving the arms or legs. Although this rhythm is observed in the planning stage of physical movements, it has been discovered by Pfurtscheller [11] that it might also appear when the human is shown a visual stimulus as a mental preparation of physical actions. This visual stimulus is called *Motor Imaginary*. After an instant power decrease, the EEG signal power increases again and this event is called *Event Related Synchronization* (ERS) as shown in Figure 5 [9]-[12].

Considering the fact that particular parts of the sensorimotor cortex area in the brain control different parts of the body, this phenomenon might help to detect and decipher human thoughts regarding motor action intent in the brain.

As mentioned before, although there are other major types of human biopotential signals such as EKG and EMG, their detailed explanations are not provided because their origin is not considered as a relevant aspect of this study.

1.3 Preprocessing

The raw biosignal data collected from human body is usually contaminated with various noise sources and artifacts. These undesired components might impact the efficiency of biological signal processing techniques. Therefore various techniques are applied to the raw signal in order to get rid of them and increase the efficiency of the Feature Extraction and Classification steps. For example, the EOG signals created by eye blinks, Electrocardiogram, or Electromyography signals from the facial muscles might all interfere with EEG signals. In addition to these artifacts, various noise sources coming from the nearby electronic devices might also affect the EEG signal.

The simplest but efficient technique in order to remove these undesired components is using basic filtering. For instance, the most informative frequency range of EEG signal is within 0 Hz and 30 Hz [10] thus the frequencies above 30Hz can simply be removed by a low pass filter. In addition to these basic filtering, it is also possible to eliminate the noise from the original signal using several different computational techniques such as *Independent Competent Analysis* (ICA) or *Principal Component Analysis* (PCA) [9], [10].

In the Independent Component Analysis technique, the multisource signal is separated into independent sub-components by considering the individual signals are statistically independent [14]. In this way, assuming the noise source and the original data are independent, we can separate the original signal from the noise. Vorobyov and Cichocki [15] apply ICA to separate the EEG signal into independent components, and after filtering the noise they reconstruct the clean EEG signal again.

Principal Component Analysis is one of the other computational techniques that could be used to eliminate the noise and artifacts from the biosignal. PCA tries to reduce the dimensionality of the signal into a smaller subspace which consists of orthogonal components that might enable the separation of the original signal from its noise components [10]. Jung et al. [16] studied the removal of artifacts from EEG signals by using and comparing both ICA and PCA techniques.

Depending on the nature of the signal and possible noise sources, one should select the best preprocessing technique and apply it before the feature extraction phase. The following section explains the feature extraction process and several techniques that are widely used in the literature.

1.4 Feature Extraction

The second component of biological signal processing is to extract distinctive features that could be a representation of the signal, discriminative enough to generate some meaningful information. One of the common feature extraction techniques is employed by transforming the original signal into frequency domain using Fourier Transform of the signal. For example, due to the varying time domain characteristics of both EOG and EEG signals, such as shifting along the time axis [17] and certain EEG frequency bands that have critical importance as mentioned, the signal is transformed into the frequency domain. The *Fast Fourier Transform* (FFT) is employed in order to generate the frequency domain representation of the signal and several features are extracted after this transform. Nakayama et al. [17] uses the amplitude of FFT taken EEG signals in order to generate the features. Felzer and Freisleben [18] uses FFT coefficients between 5-15 Hz in order to obtain features for classification of EEG signals.

Principle Component Analysis is also used for feature extraction as well as it is used for noise removing [19]. Since PCA downsamples the signal into principle components by utilizing its eigenvectors, the eigenvalues associated with these eigenvectors can give useful information about the signal. Lee et al. [19] uses PCA generated eigenvalues in order to train two different classifiers for EEG signals.

In addition to FFT and PCA, the power of the signal in certain frequency ranges which is called *bandpower* (BP) might also be used as features in the classification of biosignal. As mentioned before for the EEG signals, certain events such as ERD cause the attenuation of the biosignal in certain frequency ranges. This causes decreasing of the signal power in certain frequency ranges [11] and could effectively be used in feature extraction as demonstrated by the researchers [4], [20]-[23]. Bandpowers might also be used in obtaining features for the EOG signals. Estrada et al. [24] studies classification of sleep stages and use bandpower of the EOG signal as features since the *Rapid Eye Movement* (REM) activity is heavily concentrated on the frequency range between 0.1 Hz and 0.3 Hz.

Depending on the requirements of the applications, it is possible to increase the number of feature extraction methods for the biosignals. One should decide on the best feature extraction method depending on the characteristics of the signal and the specific tasks or signals that are studied. In this study, we have used the mean bandpower features within *alpha* and *beta* bands for the EEG signals. These values are obtained by band-pass filtering the signal within specific frequency ranges, squaring them and then taking the average. The reason to select BP values as features for this study is the successful applications in the literature in classification of the EEG signals [10-14] and the relevance of the technique for the events such as ERD that could be detected by looking at the bandpower.

The next section gives detailed information about the third component in biological signal processing called *Classification*. The theory and the literature review related to the classification and clustering methods covered in this study are presented in section 2. Then, we discuss results and comparisons in section 3. Finally, the applications and experiments are presented in section 4.

2. Clustering and Classification

According to Alpaydin, “Machine learning is programming the computers to optimize a performance criterion using example data or past experience” [24]. The software (or more generally called the *agent*) analyzes the available data and tries to predict meaningful information for the unseen data or extracts a description for the analyzed data. Machine learning has been applied to the problems where there is no human expertise needed or unable to obtain but still intelligent decisions can be made. For example, today’s computers can recognize spoken speech with a high success rate by utilizing Machine Learning techniques [26], [27]. Recognizing spoken speech is a difficult problem due to the highly varying nature of the signal itself due to different accents, gender, and age. In this case, we cannot program the computers directly to solve this problem. Some intelligent techniques are needed in order to understand these signals and map them to the specific outputs. In summary, Machine Learning techniques are applied to the most problems in order to develop systems that could make intelligent decisions [24].

Machine learning techniques can be grouped into *Supervised Learning*, *Unsupervised Learning* and *Reinforcement Learning* techniques [24]. Supervised Learning is learning a pattern from its positive and negative examples and creating a mapping between the characteristic features of the data into classes they belong. Unsupervised learning is used to model the data itself based on only the input data without any supervisor. As a result of unsupervised learning, the obtained model can give some idea about the organization of the data. For example, clustering algorithms are considered within the group of unsupervised learning. They group the similar type of data into clusters so that a model representing the structure of the data can be obtained. Reinforcement learning is the type of learning that the agent takes some actions in an environment and receives reward or penalty for its actions. First, all actions are equally important for the agent and these actions are performed with the same priority in random selections. After an action is performed, according to the type of the action, whether it is a desired or an undesired, a reward or penalty is assigned to each action. Thus, the agent is reinforced to perform a better action so that it may learn to behave in a more desirable fashion.

Throughout this study, several supervised and unsupervised learning methods have been analyzed. In some cases these two learning methods have been combined to create *hybrid learning* structures. Regarding the unsupervised techniques, the detailed analysis of clustering algorithms has been made. The most frequently used clustering algorithms in the literature such as Fuzzy *C*-Means, *K*-means, and Particle Swarm Optimization algorithms have been compared based on their clustering abilities. After analyzing the strengths and weaknesses of these learning techniques, they are used in the classification step and their effects on the classification accuracy have been investigated.

The following sections explain the machine learning techniques (both clustering and classification) analyzed or designed throughout this study.

2.1. Clustering Algorithms

In this section, we will study several clustering algorithms as methods of unsupervised learning. The studied algorithms are *K*-means, Fuzzy *C*-means, and Particle Swarm Optimization Clustering algorithms. The theory and literature related to these algorithms are presented, their strengths and weaknesses are discussed.

2.1.1 *K*-means Clustering Algorithm

K-means is one of the most common unsupervised learning methods that clusters the data according to the centroids calculated as means of clusters. Each data member is assigned to the nearest clusters by utilizing the Euclidean distance between the data and the cluster centroid. The algorithm iteratively calculates the centroids of the clusters by minimizing the following objective function

$$J_{min} = \sum_{i=1}^{c} \sum_{k=1}^{n} \|x_k - v_i\|^2 \tag{1}$$

which is a minimization of the sum of squared error distance from each data point to its cluster centroid. In equation (1), x_k represents the k^{th} data point, v_i represents the i^{th} cluster centroid, *n* is the total number of data points in the set, and *c* is the number of clusters that the data is desired to be partitioned. The algorithm terminates when there is no change in the centroid locations or equivalently in the objective function value.

The *K*-means algorithm pseudo code is as follows [24]

```
Assign c number of initial centroids for the clusters
While the change of centorid locations is greater than some epsilon value
    Assign each data into the clusters using the minimum distance measures
    For i=1 to c
    Calculate the new centroid locations with the mean of all of the samples in the cluster i
    end For
    end While
```

Although *K*-means algorithm is simple and rapidly converging algorithm, it has a major drawback. The algorithm might be trapped in local minima depending on the initial cluster centroids which might avoid the algorithm to cluster the data well enough [28], [29], [30].

The authors in [28] propose a stochastic approach for *K*-means in order to alleviate the local minima problem for the algorithm. The study in [29] similarly investigates the local minima problem and proposes a technique by combining *K*-means clustering with *Particle Swarm Optimization* (PSO) Clustering. They initially run the PSO clustering over the data and stop the algorithm at some point according to the objective function value and let the *K*-means algorithm run. However, deciding the specific value that the PSO algorithm should stop might be another problem since it might highly vary according to the data being clustered. The authors in [30] study the same drawback of *K*-means algorithm by initializing the centroids utilizing *Genetic Algorithm*s and apply this modification of the algorithm to an online shopping market application.

2.1.2 Fuzzy *C*-means Clustering Algorithm

The aforementioned *K*-means clustering algorithm is known as a crisp clustering since portioning of the dataset is performed according to the minimal distance calculation and each data is classified into single cluster. However in Fuzzy *C*-means (FCM) clustering algorithm which is first proposed by Dunn [31] and later developed by Bezdek [32], the data is partitioned into the clusters according to their membership values. These values change between zero and one and represent the degree of how close the data to each cluster center. In FCM algorithm, the data can belong to other clusters up to some degree which is determined by these membership values.

The Fuzzy *C*-means objective function is very similar to the *K*-means but it includes an additional fuzzy term. The equation (2) represents the objective function of FCM algorithm,

$$J_{min} = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik}) \| x_k - v_i \| \tag{2}$$

where x_k represents the *k*th data entry in the dataset, u_{ik} represents the membership matrix, *c* is the number of clusters, *n* is the number of data in the dataset, and v_i is the *i*th cluster center. According to [32], the following u_{ik} and v_i formulas will minimize the above objective function.

$$u_{ik} = (\sum_{j=1}^{c} ({}^{D_{ik}}\!/_{D_{jk}})^{\frac{2}{m-1}})^{-1} \qquad 1 \leq i \leq c\,, 1 \leq k \leq n \tag{3}$$

$$v_i = \frac{\sum_{k=1}^{n} u_{ik}^{m} x_k}{\sum_{k=1}^{n} {u_{ik}}^{m}} \qquad \sum u_{ik} = 1 \tag{4}$$

where $D_{ik} = \| x_k - v_i \|$ and represents the distance between the *k*th data entry and the *i*th cluster center. The algorithm updates the cluster centers according to equation (4) and calculates the membership matrix based on the new cluster centers. It terminates when $\| v_t - v_{t-1} \| \leq \varepsilon$ that is the cluster centroids do not change any more. The parameter *m* is called the *fuzzification constant*. It adjusts the overlapping of the clusters [33] and is always greater than one. More crisp clustering is obtained when *m* gets closer to one. As *m* gets larger, the overlapping of the clusters is also increased. The effect of the fuzzification constant *m* on clustering is illustrated in Figure 6.

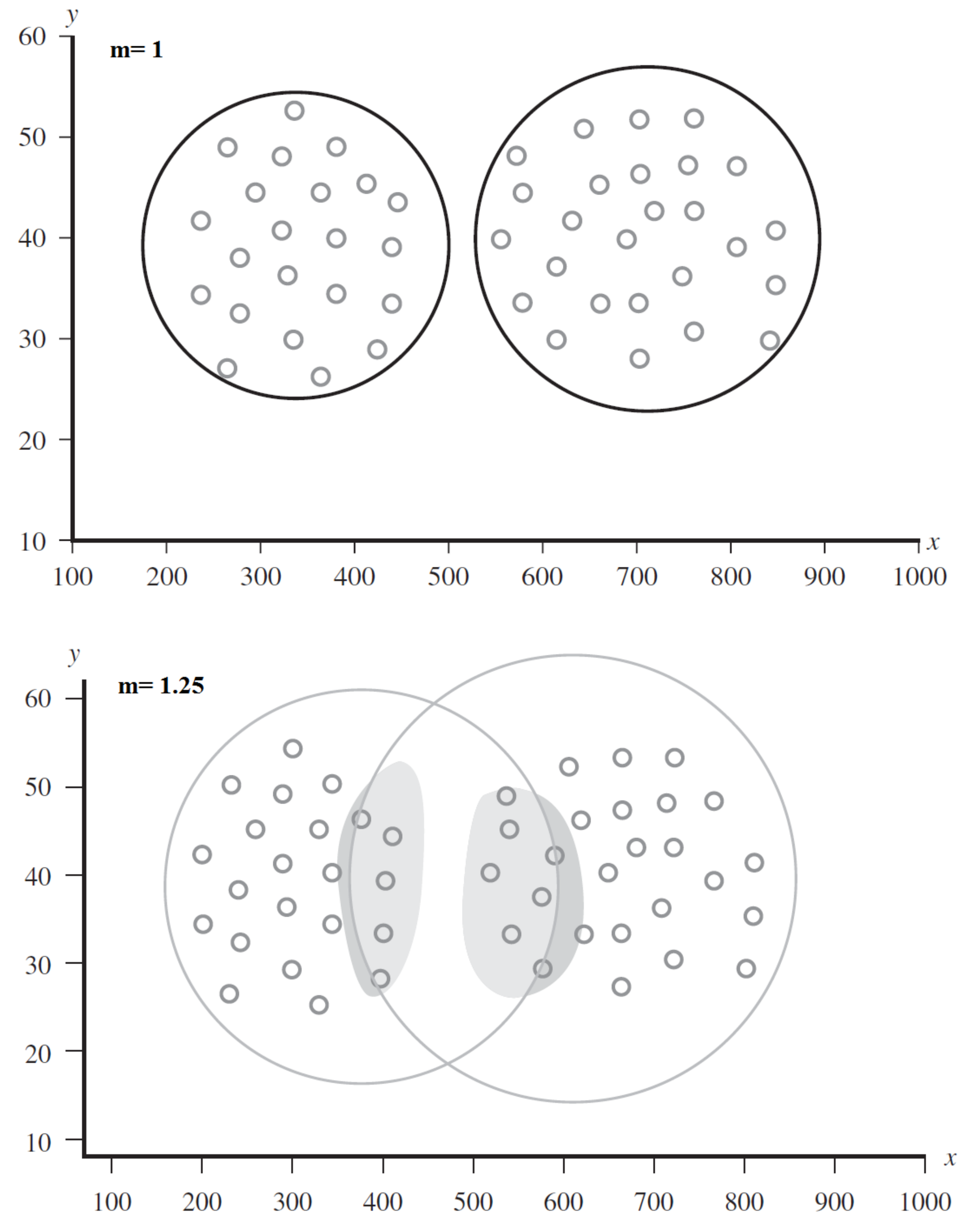


Figure 6 The Effect of Overlapping in Clustering Depending on the Fuzzification Constant *m*, *x*-*y* Axis represents the Cartesian Coordinates of Randomly Generated Data Points [33]

Although FCM has advantages in clustering, such as the fast convergence and degree of memberships which makes the clustering more realistic rather than crisp clustering, it also suffers from various limitations. Cox [33] and Thomas et al. [34] explain several problems with the algorithm. One of the limitations is that since the membership values depend on the other cluster data points (partial membership), the cluster centers tend to move towards the center of *all* the data points in order to increase the fuzziness. This is not a desired effect for the clustering as explained by Wang et al. [35]. They explain that in order to obtain good partitioning of the dataset, the fuzziness should be minimized as much as possible and the objective function of the clustering algorithm should converge. The same problem with cluster centers in FCM algorithm is also explained by [36] that proper location of the cluster centers is not the important focus of FCM algorithm since the centers are moved according to the membership values of the data. In addition, similar to the *K*-means algorithm converging to local minima based on different center initializations may also exist with FCM algorithm as studied in [37].

Recently, Particle Swarm Optimization is being used for clustering by researchers based on its performance in finding global solutions of optimization problems. The next section describes PSO in detail and presents the state of the art PSO algorithm.

2.1.3 Particle Swarm Optimization (PSO) and PSO Clustering

Particle Swarm Optimization is an optimization technique inspired by social behaviors of bird flocking and fish schooling developed by Kennedy and Eberhart [38]. Each particle in the swarm is a potential solution to an optimization problem and has an associated fitness value calculated by the fitness function to be evaluated. In every iteration of the algorithm, each particle is allowed to update its position in the search space evaluating its own fitness and the fitnesses of the neighboring particles. The algorithm is terminated when the specified maximum number of iterations is reached or there is no improvement in the global best solution of the swarm. The particle which has the best fitness is selected as global solution at the end of the last iteration. For each particle k in dimension d, velocity and position of particles are updated based on equations (5) and (6).

$$v_k(n+1) = v_k(n) + \beta_c . r . \left(p_k - x_k(n)\right) + \beta_s . r . (g(n) - x_k(n)) \tag{5}$$

$$x_k(n+1) = x_k(n) + v_k(n+1) \tag{6}$$

where;

$v_k(n) =$ current velocity of particle k

$x_k(n) =$ current position of particle k

$p_k =$ best position of particle k

$g(n) =$ global best position of the swarm

$\beta_c =$ cognitive weight

$\beta_s =$ social weight

$r =$ random number (0,1)

Below is a simple pseudo code of PSO algorithm [39]

```
    Initialize Parameters
    Initialize swarm
    While the number iteration is less than the maximum iteration
```

```
        Find best particle
        Find global best
        Update velocity
        Update position
    end While
```

The random number term, *r*, in equation (5) causes the algorithm to explore the search space continuously and thus help prevent the swarm from converging to a local solution. These constant values in equation (5) together with the social and cognitive weights adjust the tension in the swarm. The high values result in fast movements toward to the global solution passing through the local solutions quickly [40]. Having random factors in the velocity update formula might cause the swarm to explode and particles not to be able to converge to the global solution. In order to avoid this situation, a maximum value for the velocity, $v_{max,}$ is usually defined [38], [39].

There have been several improvements and modifications to the standard PSO algorithm such as *Inertia PSO* and *Constriction PSO*. The next two sections introduce these two types of PSO modifications.

2.1.3.1 Inertia Particle Swarm Optimization

In the Inertia PSO, the current velocity of the particle is weighted with a constant Inertia Factor *w* when the velocity is calculated for the next iteration [41]. Adding the *Inertia Factor* limits the velocity of the particles so that an explosion effect can be prevented. Thus, The equation (5) becomes as in equation (7)

$$v_k(n+1) = wv_k(n) + \beta_c.r.\left(p_k - x_k(n)\right) + \beta_s.r.\left(g(n) - x_k(n)\right) \quad (7)$$

This inertia factor might be a constant value throughout the algorithm or it might dynamically decrease as the algorithm continues.

2.1.3.2. Constriction Particle Swarm Optimization

The other modification to the original PSO algorithm is the *Constriction PSO* [42] that we have also utilized in this study. The Constriction PSO is a way to guarantee the convergence in the system through the assignment of eigenvalues with a constriction coefficient determined by cognitive and social acceleration coefficients. The velocity and position updates for this type of PSO algorithm are shown in equations (8), (9) and (10)

$$v_k(n+1) = \chi[v_k(n) + \beta_c.r.\left(p_k - x_k(n)\right) + \beta_s.r.\left(g(n) - x_k(n)\right)] \quad (8)$$

$$x_k(n+1) = x_k(n) + v_k(n+1) \quad (9)$$

$$\chi = \frac{2}{\left|2-\phi-\sqrt{\phi^2-4\phi}\right|} \quad (10)$$

where;

$\phi = \beta_c + \beta_s,\ where\ \phi > 4$

χ: Constriction constant

In equation (8), β_c and β_s are the cognitive and social constants as explained earlier. The parameter, ϕ, is the parameter that determines the constriction constant χ. In [42], in order to guarantee the convergence of the PSO algorithm, the above parameters are selected as: $\chi =$

0.729, $\beta_c = U(1,4)$ (uniformly distributed random number between one and four) and $\beta_s = 4.1 - \beta_c$. The greater the constriction factor than 0.729, the faster the algorithm converges however the probability of explosion might also increase.

Eberhart and Shi [42] proved that the Constriction PSO gives better results compared to the Inertia PSO according to the experimental results obtained by examining five different objective functions:. *spherical*, *Rosenbrock*, *Rastrigin*, *Griewank*, and *Schalffer's f6* function on sample datasets. For our clustering implementation in this study, we have decided to use Constriction PSO method. The next section introduces the application of PSO algorithm into the clustering.

2.1.3.3 PSO Clustering

Application of PSO into clustering was first proposed by Merwe and Engelbrecht in [43]. Each particle in the swarm represents a potential solution for the clustering prototypes (centers) and has a fitness value calculated by the objective function. In [43], the objective function is chosen as the *quantization error* given in equation (11)

$$J_e = \frac{\sum_{j=1}^{N_c}[\sum_{\forall Z_p \in C_j} d(Z_p, m_j)/|C_j|]}{N_c} \tag{11}$$

where $|C_j|$ represents the number of data vectors belonging to cluster *j* that is the frequency of that cluster, m_j represents the centroid of cluster *j* and Z_p is the *p*^*th*^ data vector in the dataset and N_c is the number of clusters that the data will be partitioned.

When the quantization error formula in equation (11) is analyzed, it can be seen that the quantization error is a measure of how close the centroid locations are to the data members in each cluster. Since it makes the intra-cluster distances decrease and inter-cluster distances increase, minimizing the quantization error results in more compact clustering as explained in

[43]. Therefore, throughout this study, the clustering abilities of the algorithms are compared by analyzing their abilities in reducing the quantization error in the dataset.

2.2. Classification Algorithms

In this section, the classification algorithms such as Radial Basis Function Networks, Fuzzy Functions Support Vector Machines, and Improved Fuzzy Functions Support Vector Machines that have been utilized throughout this study are presented.

2.2.1 Radial Basis Function Networks

A Radial Basis Function Networks (RBFN) is a type of feed-forward Neural Network which consists of three layers: input layer, hidden layer, and output layer as shown in Figure 7. The input layer contains n dimensional feature vectors entering the network. The hidden layer is composed of radially symmetric Gaussian kernel functions shown in equation (12)

$$\phi_i = e^{\frac{-\|x-x_i\|^2}{\sigma^2}} \tag{12}$$

where $x_i,\ i = 1,2,3 \ldots, m$ and m being the number of kernels, represents the i^{th} kernel centroid in the hidden layer, x represents the feature vector in the dataset and ϕ_i values are calculated for each data vector with kernel centroids determined by any clustering technique [44]. Note that, the closer x is to x_i, the higher the influence it will have in the hidden layer outputs since ϕ_i values will be larger.

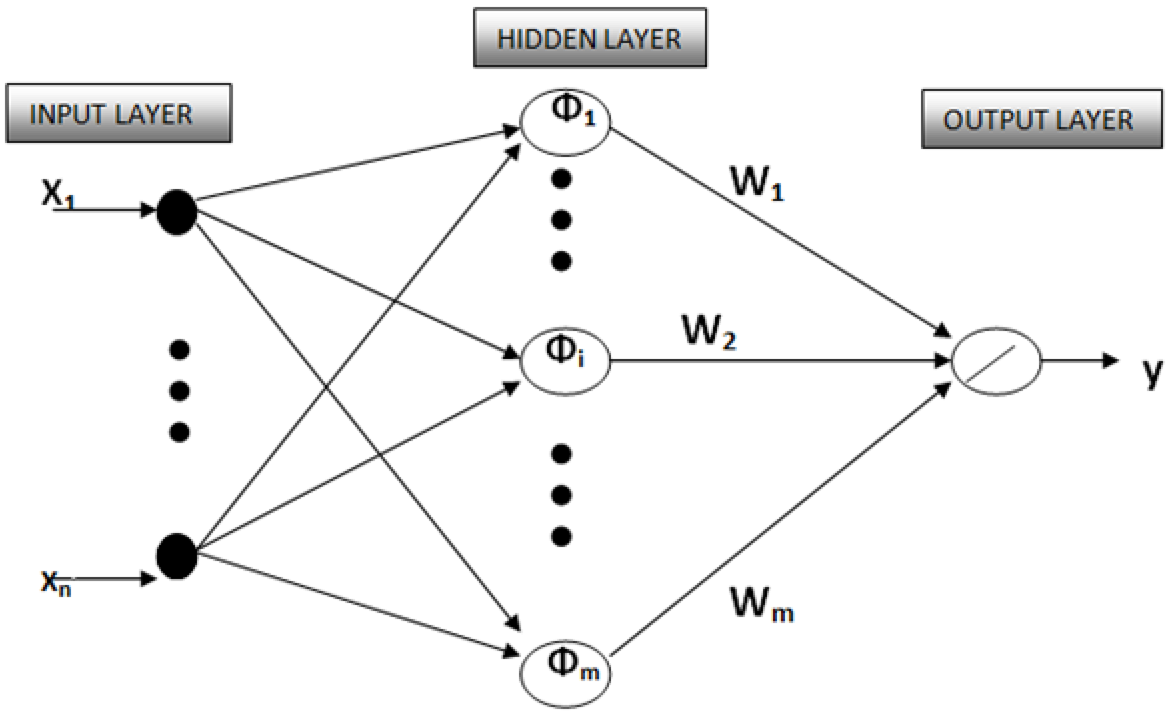
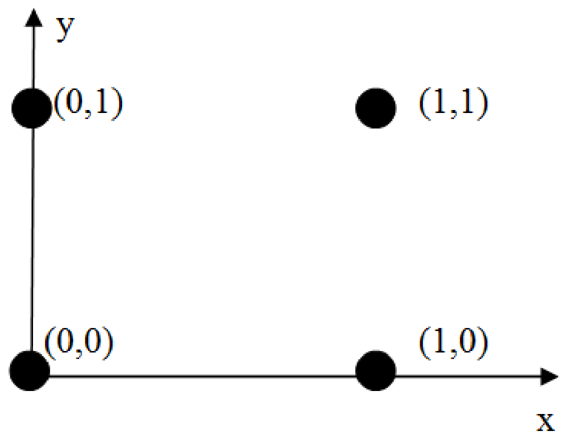


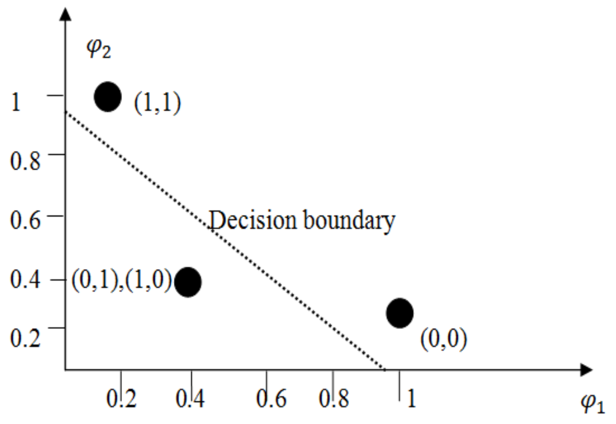
Figure 7 Radial Basis Function Networks Structure, X_n represents the n^{th} Feature of Data, ϕ_i represents the i^{th} Hidden Layer Kernel, W_m represents the weight of the m^{th} link between Hidden and Output Layer, *y* represents the output of RBFN network

By the help of the hidden layer, feature vectors which are in R^n, are mapped to a higher dimensional space, R^m, so that the data can more likely become linearly separable according to Cover's theorem on the separability of random patterns [45].

The most famous example that demonstrates the separability of patterns in higher dimensions is the *XOR problem* [44]. The problem is constructing a classifier with the input patterns (1,1), (0,1), (0,0), and (1,0) so that the classifier will give a binary output 0 when the input patterns are (1,1) or (0,0) and the binary output 1 when the input patterns are (1,0) or (0,1). The input pattern space is depicted in

Figure 8.





(A) (B)

Figure 8 (A) Inputs Patterns in *x-y* coordinate system that form the XOR Problem (B) The Transformed Input Patterns into $\varphi_1 - \varphi_2$ space using two Gaussian Kernels

As it can be seen from

Figure 8 (A), the input patterns (1, 1) and (0, 0) are *not* linearly separable from the other two patterns. If the input patterns are mapped into another dimensional space by using two Gaussian kernels, the input patterns turn out to be linearly separable as depicted in

Figure 8 (B). Table 2 shows the values of mapped input patterns in $\varphi_1 - \varphi_2$ space where $\varphi_1 = e^{-\|x-t_1\|}$, $\varphi_2 = e^{-\|x-t_2\|}$ and $t_1 = [1,1]^T$, $t_2 = [0,0]^T$

Table 2 Specification of the Hidden Functions for XOR Problem

Input Pattern $\boldsymbol{x}$	Hidden Function $\boldsymbol{\varphi_1}(\boldsymbol{x})$	Hidden Function $\boldsymbol{\varphi_2}(\boldsymbol{x})$
(1,1)	1	0.1353
(0,1)	0.3678	0.3678
(0,0)	0.1353	1
(1,0)	0.3678	0.3678

Thus, we may conclude that input patterns that are not linearly separable in their current space might be transformed to a different or a higher input space by using kernel functions so that the input pattern might become linearly separable.

As can be seen from Figure 7, in RBFN the outputs of the hidden layer are connected to the output layer by weighted links. The output node of RBFN is a linear summation described in equation (13)

$$y = \sum_{j=1}^{m} W_j \phi_j \tag{13}$$

where W_j represents the weights of the links between hidden and output layers and y is the output of the mapping.

Let ϕ_{ji} represent the value of the *j*th basis function, ϕ_j, for the *i*th data in the dataset. If there are m inputs and m basis function units, the matrix form can be written in equation (14) to represent Figure 7.

$$\underbrace{\begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1m} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{m1} & \phi_{m2} & \cdots & \phi_{mm} \end{bmatrix}}_{\boldsymbol{\phi}} \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}}_{\mathbf{W}} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}}_{\mathbf{Y}} \tag{14}$$

The weight matrix **W** can be calculated using the inverse of $\boldsymbol{\phi}$ as stated in equation (15).

$$\mathbf{W} = \boldsymbol{\phi}^{-1}.\mathbf{Y} \tag{15}$$

When the number of inputs is greater than the number of hidden units, we can still find the weight matrix utilizing the *pseudo-inverse* of the $\boldsymbol{\phi}$ matrix. The pseudo inverse is calculated by $pinv(\boldsymbol{\phi}) = (\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{\phi})^{-1}\boldsymbol{\phi}^{\mathrm{T}}$.

During the testing phase of the data with RBFN, $\boldsymbol{y}$ outputs are found by using the weights obtained in the training phase and $\boldsymbol{\phi}$ values calculated using only the test data. The $\boldsymbol{y}$ outputs are thresholded at the end in order to generate binary class label outputs.

In employing RBFN for a classification problem, finding the appropriate centers for kernel functions has critical importance on the generalization capability of the classifier [46], [47]. *K*-means clustering algorithms have been widely used to determine the cluster centers for the RBFN [44], [46], [47]. Although selecting these centers has critical importance, surprisingly not many extensive studies do exist in the literature examining the importance of the clustering algorithms on the classification performance [48].

Hongyang et al. [48] only studied the variation of *K*-means clustering using a dynamic *K*-means clustering algorithm and compared the performance with the standard *K*-means algorithm.

According to the experimental results by [48], a better selection of the centers increases the classification performance.

In this thesis document, two clustering algorithms on the classification performance of the RBFN are explored and compared. In the next section, another classifier called Fuzzy Function Support Vector Classifiers is introduced.

2.2.2 Fuzzy Functions Support Vector Classifiers (FFSVC)

The Fuzzy Functions Support Vector Classifier is a new classifier design proposed by Celikyilmaz et al. [49]. It combines the Fuzzy *C*-Means Clustering Algorithm with any classification methods to desig a new efficient classifier. With the conventional classifiers the dataset, which has possible multi-model structure, is classified using a single classifier. This might be a possible drawback for the classification tasks. The novel classifier approach captures the hidden partitions in the dataset using FCM clustering and applies one classifier for each partitions found by the clustering method.

Another important property of the technique is that the membership values found by FCM clustering augment the original training feature set as a new dimension per each data. This helps by increasing the dimensionality of the input space so that the data might more likely become linearly separable. In addition, the data points that stay close to each other with opposite class labels in the input space might move away from each other. Since the features of the data are represented with one more additional feature, the identification of the data is also enhanced in the classification technique.

The general structure of the classifier is represented in Figure 9. Let μ_i represent the membership value of the input data in the training set belonging to the i^{th} cluster and x_j represents the j^{th} feature of the data vector where $j = 1, 2, 3, \dots nv$. Here, nv represents the feature dimension of the dataset. After the FCM clustering is performed on the training data, a one dimensional input matrix $\Phi_i = [x_1, x_2, x_3, x_4, \dots \mu_i]$ is created for each cluster partitioned by FCM. Here, Φ_i represents the augmented input vector that includes the membership value belonging to the i^{th} cluster. This input matrix is created for all clusters that the dataset is partitioned into.

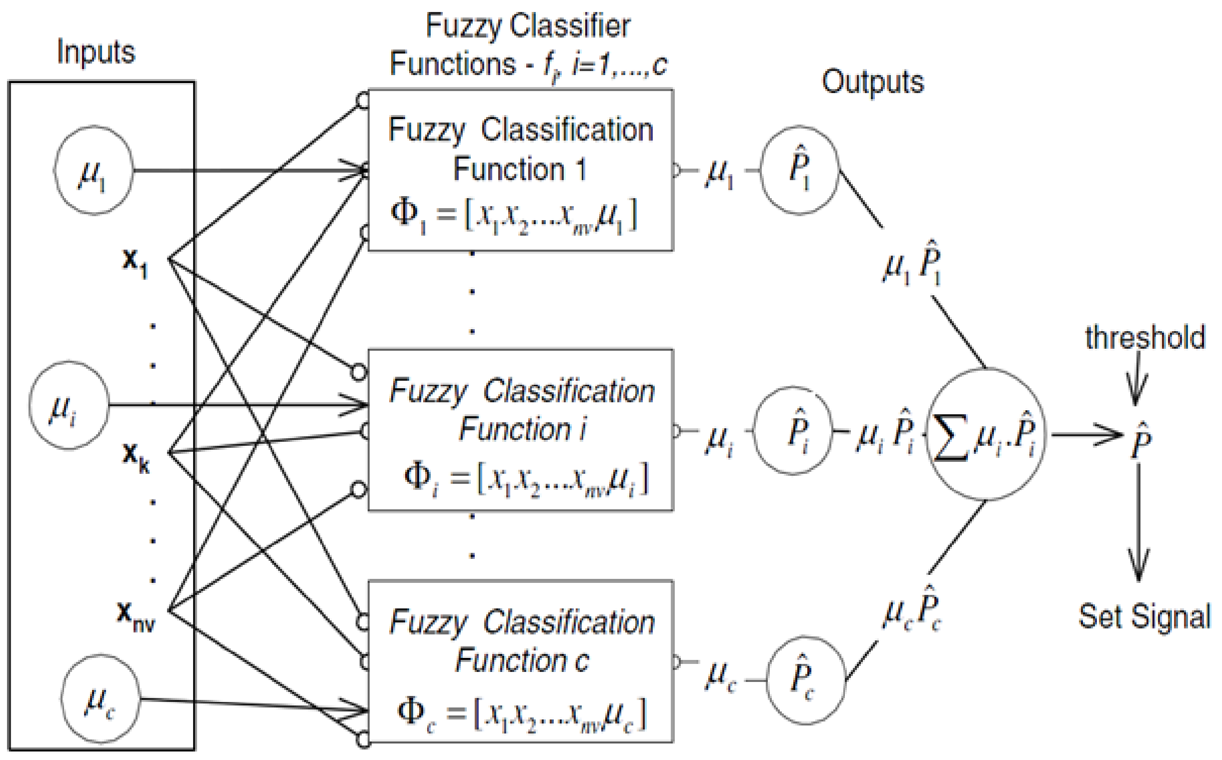

Inputs
Fuzzy Classifier Functions - f_i, $i=1,\dots,c$
Outputs
Fuzzy Classification Function 1
$\Phi_1 = [x_1 x_2 \dots x_{nv} \mu_1]$
Fuzzy Classification Function i
$\Phi_i = [x_1 x_2 \dots x_{nv} \mu_i]$
Fuzzy Classification Function c
$\Phi_c = [x_1 x_2 \dots x_{nv} \mu_c]$
μ_1
μ_i
μ_c
x_1
x_k
x_{nv}
$\hat{P}_1$
$\hat{P}_i$
$\hat{P}_c$
$\mu_1 \hat{P}_1$
$\mu_i \hat{P}_i$
$\mu_c \hat{P}_c$
$\sum \mu_i . \hat{P}_i$
threshold
$\hat{P}$
Set Signal

Figure 9 Fuzzy Functions Support Vector Classifier Schematic [49]

Depending on the dataset, the transformation of these membership values might also be added as additional feature in the input matrix together with the original membership values, e.g. exponential transformation *exp(μ_i)*

After the input matrices are created, one classifier for each cluster in the dataset is built. Depending on the system, this classifier may take the form of a linear classifier such as a *Logistic Regression* or a nonlinear classifier such as *Support Vector Machine* (SVM) [24]. These classifiers that take the input matrix Φ_i and generate a prediction for these input vectors are called *Fuzzy Classifier Functions* (FCF). If a SVM is selected as the FCF, the output of these Fuzzy Classifier Functions is the probability estimate of the class labels generated *by Platt's probability approximation* [51] represented by $\hat{P}_i$ in Figure 9. Since this part of the FFSVC method corresponds to the fuzzy *if-then* rules section in a *Fuzzy Inference System* (FIS) [52], these functions are named Fuzzy Classifier Functions and the novelty of the classifier comes from the property that the classifier can learn the fuzzy if-then rules from the data automatically.

The outputs of the FCFs are multiplied with the membership values μ_i in order to find a crisp output as a result of the classifier. This part also corresponds to the defuzzification phase of standard FIS. The result of the probability output is thresholded by 0.5 in order to generate binary class outputs.

2.2.3 Improved Fuzzy *C*-Means Algorithm and Improved FFSVC (IFFSVC)

As we mentioned earlier, the membership values obtained from FCM clustering algorithm can be used as additional predictors for the data during the classification. Celikyilmaz et al. [50] proposed a modification to the standard FCM objective function by including the difference between the class labels and a probability estimation coming from the Fuzzy Functions Classifier to be minimized. This helps the FCM algorithm to optimize the membership values which could better enhance the prediction for each dataset as discussed in the previous section. The modified objective function is shown in equation (16)

$$J_{min} = \sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})\|x_k - v_j\| + \sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})\|x_k - v_j\| \, (\mathrm{y_k} - \mathrm{P_{ik}}((\mathrm{y_k} = 1 \, |\mathrm{f}(\tau_{\mathrm{ik}}))))^2 \tag{16}$$

where the first term is the same as the objective function of the standard FCM and the second term is the squared error of the difference between the actual class label and the probability output of the *i*th Fuzzy Classifier Function (FCF) on the *k*th data, P_{ik}, as introduced in the previous section. $\mathrm{P_{ik}}((\mathrm{y_k} = 1 \, |\mathrm{f}(\tau_{\mathrm{ik}}))$ represents the probability of the class output being equal to one for the *i*th Fuzzy Classifier Function on the *k*th data in the dataset. The $(\mathrm{y_k} - \mathrm{P_{ik}}((\mathrm{y_k} = 1 \, |\mathrm{f}(\tau_{\mathrm{ik}}))))^2$ term in the objective function is called the *error term*.

It was proven in [50] that the membership update formula in equation (17) is the Lagrangian multiplier of the objective function that can minimize the objective function in equation (16).

$$u_{ik} = (\textstyle\sum_{j=1}^{c}(\frac{D_{ik}+(y_k-P_{ik})^2}{D_{jk}+(y_k-P_{jk})^2})^{\frac{2}{m-1}})^{-1} \tag{17}$$

where D_{ik} represents the Euclidean distance between the *i*th cluster center and the *k*th data vector. Since the error term of the objective function in equation (16) does not include the cluster center term, v_j, the center update formula of the standard FCM stays the same.

Note that, in order to obtain the probability values for the membership update function, only the initial membership values are used as feature vectors excluding the original x_k data features for the FCF. The probability estimates are performed according to the initial membership values. These initial memberships can be obtained by running either the standard FCM or a crisp clustering technique such as *K*-means.

The Improved Fuzzy Functions Support Vector Classifier (IFFSVC) uses this improved FCM clustering algorithm in order to generate improved membership values for the data and classifies them in the same way as FFSVC does.

3. Results and Comparisons

In this section, a comparison of different clustering and classification techniques, their performance analysis, and a novel classification approach based on Particle Swarm Optimization Clustering and Radial Basis Function Networks is presented.

3.1 Standard Datasets and Feature Extraction Method for EEG Datasets

There are two types of datasets used throughout this study for classification. The first type of datasets is the standard datasets obtained from UCI Machine Learning Repository [53] such as *ionosphere, diabetes, liver,* and *cancer*. These are two-class datasets that include various features and a binary class label that refers the class of corresponding feature vector. The other type of datasets is the standard EEG datasets such as *X11, O3V,* and *S4b* that include the EEG data for different subjects. These are obtained from the BCI Competition *IIIb* [54]. The datasets *B0101T* and *B0102T* also include the EEG data from the BCI Competition *IV* dataset *2b* [55]. The collection methods and the feature extraction method used to create these datasets are also explained in detail later in this section. The *Medical* dataset contains statistical data related to patients coming to the clinic provided by Bellevue Hospital in New York. In this dataset, there are 19 features that include demographical attributes of each patient such as ethnicity, sex, current diseases and additional binary class label representing whether the patient came to the scheduled appointment or not.

The EEG data used in this study is collected from different subjects at multiple sessions including several runs each. The electrodes are placed according to International 10-20 system. The positions placed on the scalp of subjects for each datasets are *C3*, *C4* and *Cz*. The EEG signal in datasets named *X11*, *S4b* and *O3VR* is sampled with 125 Hz and filtered between 0.5Hz and 30Hz using a Notch Filter. These dataset names represent different subjects that the EEG signal is collected. On the other hand, the EEG signal in the datasets *B0101T* and *B0102T* has a sampling frequency of 250 Hz and filtered between 0.5 Hz and 100 Hz using a Notch Filter. The data are collected according to motor imaginary pictures shown as cues.

Timing intervals for the collection of datasets is shown in Figure 10. One trial contains eight seconds of recording. Shortly after a fixation cross is displayed on the screen, a short cue beep is generated as a warning to the subject indicating that one of two visual cue images will be displayed.

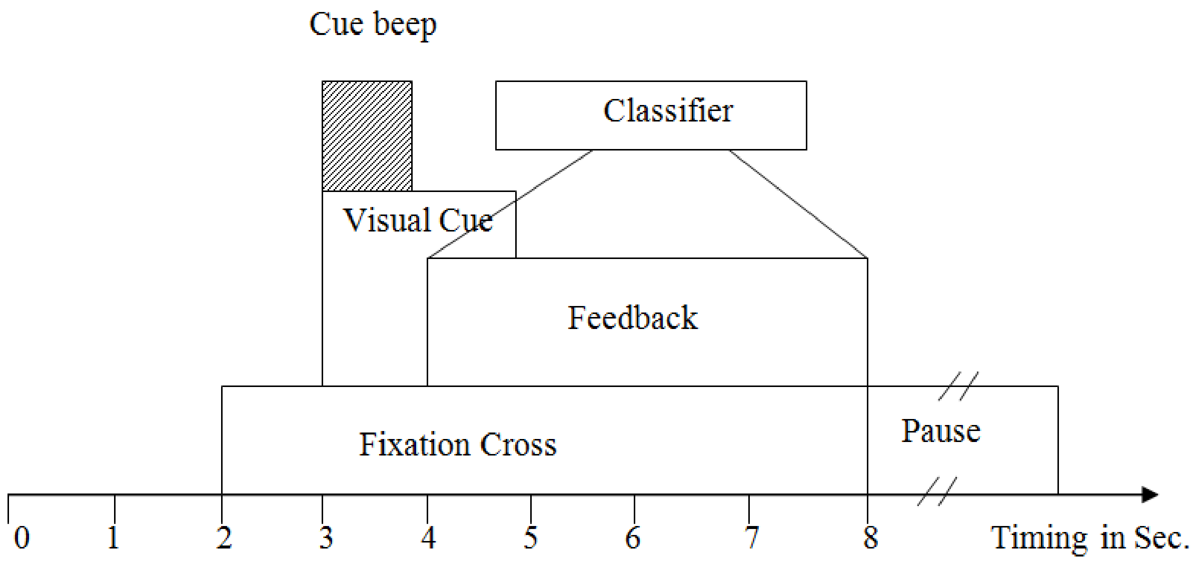


Figure 10 Informative timing schematic for signal collection in the EEG dataset

Cue images displayed to the subjects on the screen are either left arrow or right arrow indicating left thinking or right thinking. After the visual cue is displayed, through a virtual reality experiment, feedback is given to the subject such as moving a ball to the left or right.

In order to extract features from these standard datasets, band power (BP) values of the signal are used. The BP values extracted within *alpha* [8-13] Hz and *beta* [13-30] Hz frequency ranges as suggested in [55] and [56]. The BP values are obtained by band-pass filtering the signal within specific frequency ranges, squaring them and then taking the average. After generating BP values, the mean band power value of the signal is calculated within the time interval of feedback display. As a result of feature extraction, a four dimensional feature vector is obtained for each trial such as $[C3_\alpha, C3_\beta, C4_\alpha, C4_\beta]$, where $C3_\alpha$ represents the mean bandpower value within the *alpha* band from the electrode $C3$. Similarly $C3_\beta$ represents the mean bandpower value within the *beta* band from the electrode $C3$.

While the subject thinks left and right, due to different intensive neurological activity on the left and right side of the brain, the band powers of the EEG signals measured from *C3* and *C4* electrodes also differ [57]. This can be discriminative enough to classify the left and right thinking for the subjects.

3.2 Comparison of Clustering Methods

In the previous section, some drawbacks related to *K*-means and Fuzzy *C*-means algorithms were discussed. In this section, experimental results are presented and algorithms are compared with PSO clustering for their abilities to cluster the data using several standard datasets.

The reason that clustering algorithms are explored in terms of their abilities to do better clustering is the critical importance of finding good cluster centers for the RBFN classifier as proposed by Wettschereck et al. [58]. According to Wettschereck, learning the center locations for RBFN hidden layer can better increase the generalization capability of RBFN classifier therefore the chosen clustering algorithm may have high importance [58].

Three clustering algorithms are evaluated in their abilities to minimize the quantization error given in equation (11). As it is discussed in the first section, minimizing the quantization error can result more compact and better clustering in terms of the final clusters center locations. Therefore the three algorithms are run on different standard datasets and their convergence plots on these datasets are presented.

Figure 10 and Figure 11 show the convergence plots of the three clustering algorithms for their minimization of quantization error. The datasets used in these plots are two EEG datasets obtained from [54]. The *x*-axis is the number of fitness evaluations, namely the evaluation of the objective function.

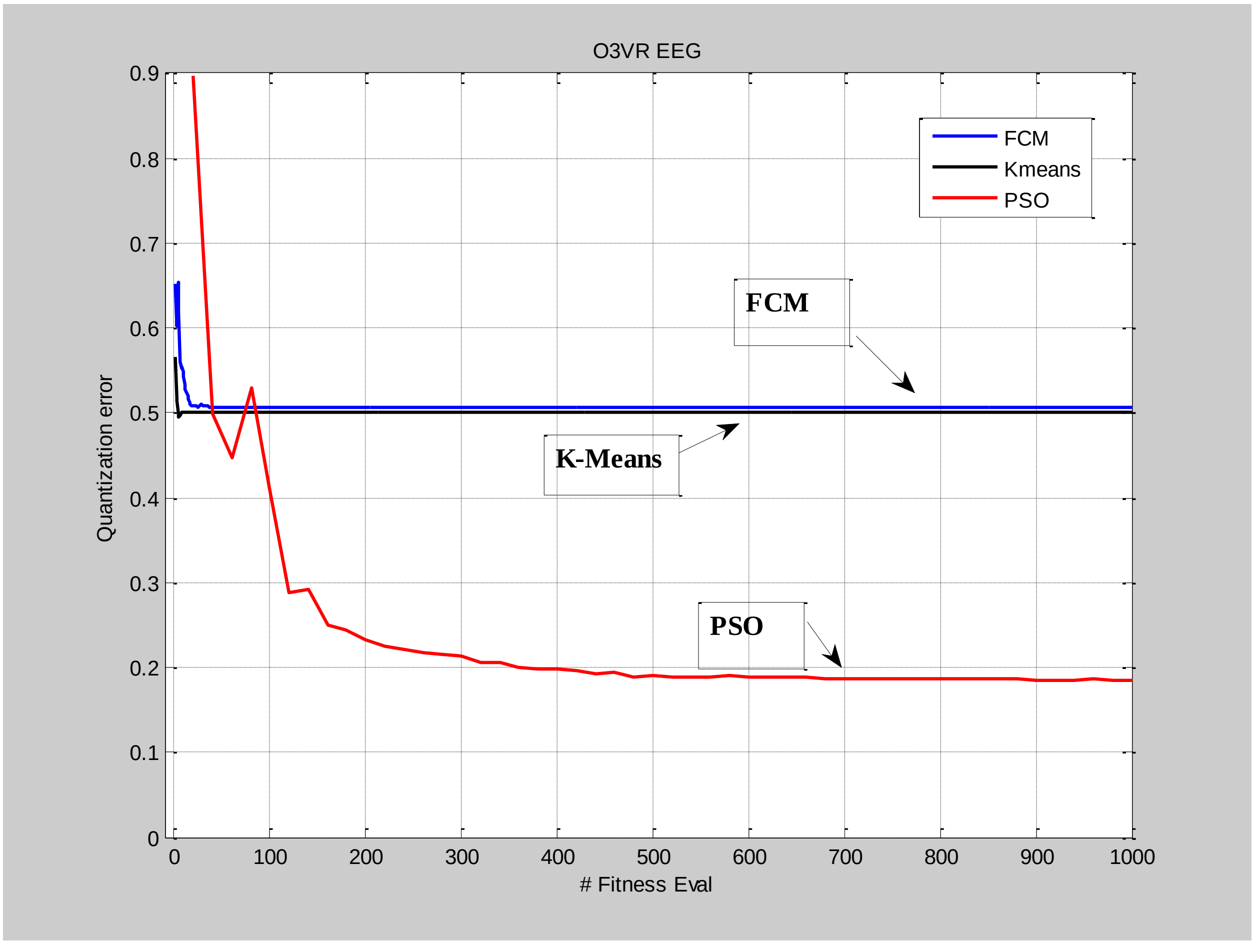


Figure 11 Quantization Error Plot for EEG O3VR Dataset

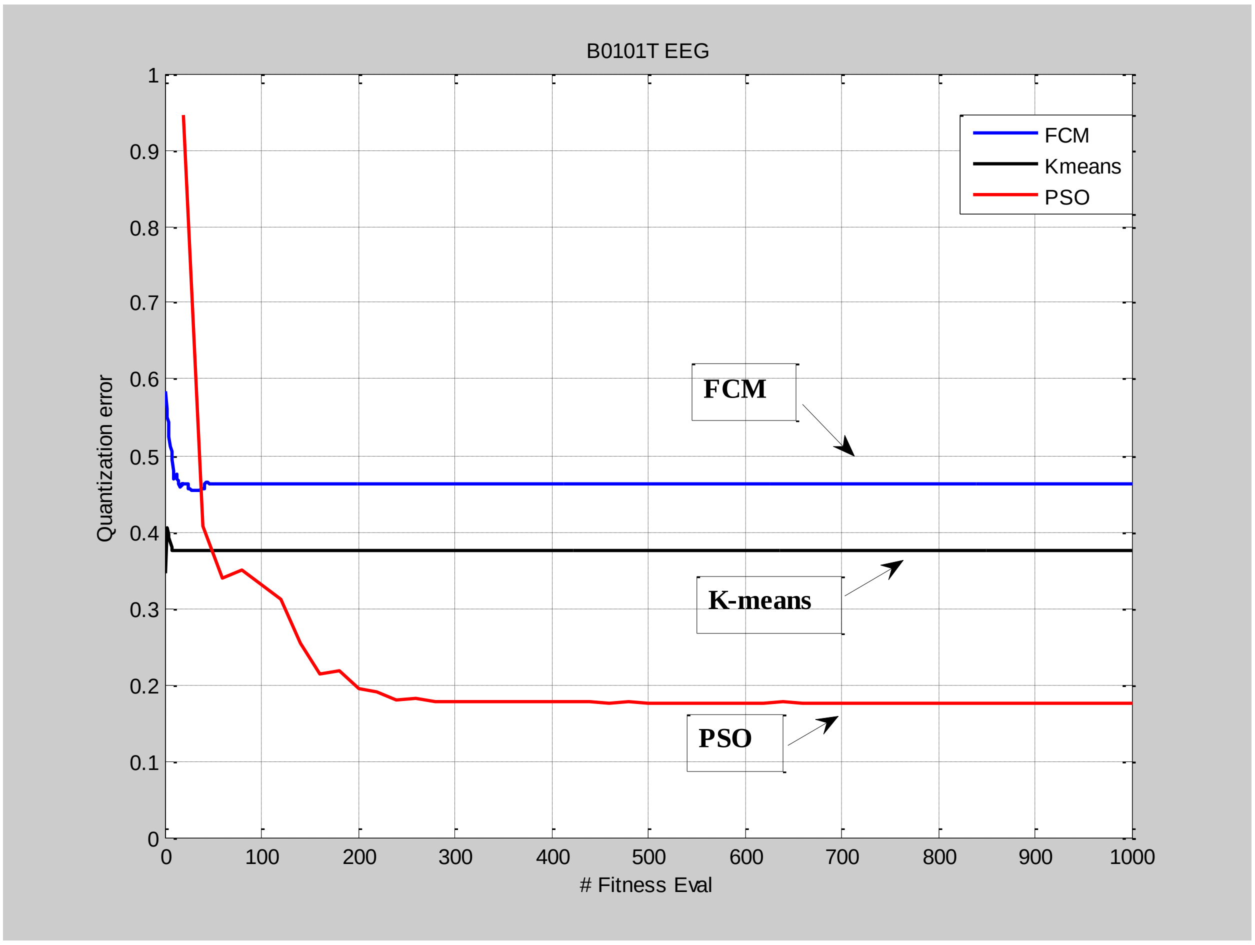


Figure 12 Quantization Error Plot for EEG B0101T dataset

In these plots, the PSO algorithm is comprised of 100 iterations with 10 particles. In order to make an objective comparison, since we know that the each particle evaluates the objective function (quantization error in our case) one time in a single PSO step, the swarm with 10 particles in the PSO algorithm makes 10 fitness evaluations in one iteration. Therefore 100 steps of the PSO algorithm means 100x10=1000 fitness evaluations per run. On the other hand, for the FCM algorithm, since the membership and centroid update formulas are the Lagrangian multipliers of the objective function and always try to minimize it, one iteration of FCM corresponds to one time evaluation of the objective function. Therefore FCM and in the same manner *K*-means algorithm are run 1000 fitness evaluations per run.

Plots in Figure 10 and Figure 11 verify that the PSO Clustering algorithm stops at a significantly better location in terms of quantization error calculation than both the FCM and *K*-means algorithms and converges to better minima for the analyzed datasets.

In addition to these, the performances of FCM and PSO clustering are also studied on the other datasets. It is similarly shown in Figure 13 and Figure 14 that the PSO clustering algorithm performs better clustering on the other datasets considering the *1000* fitness evaluations of both FCM and PSO algorithms.

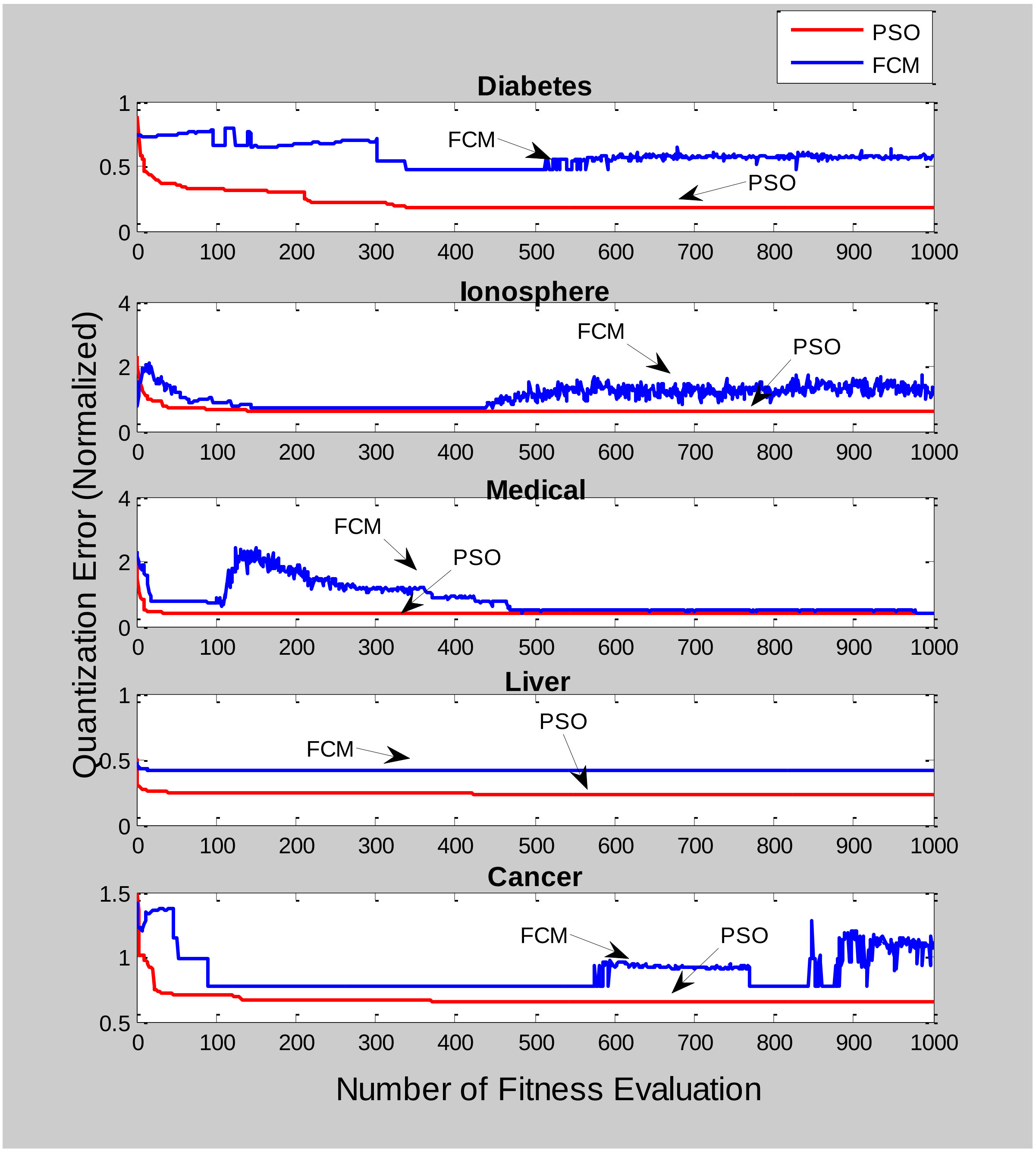


Figure 13 Fuzzy C Means versus PSO Clustering on Standard non-EEG Datasets

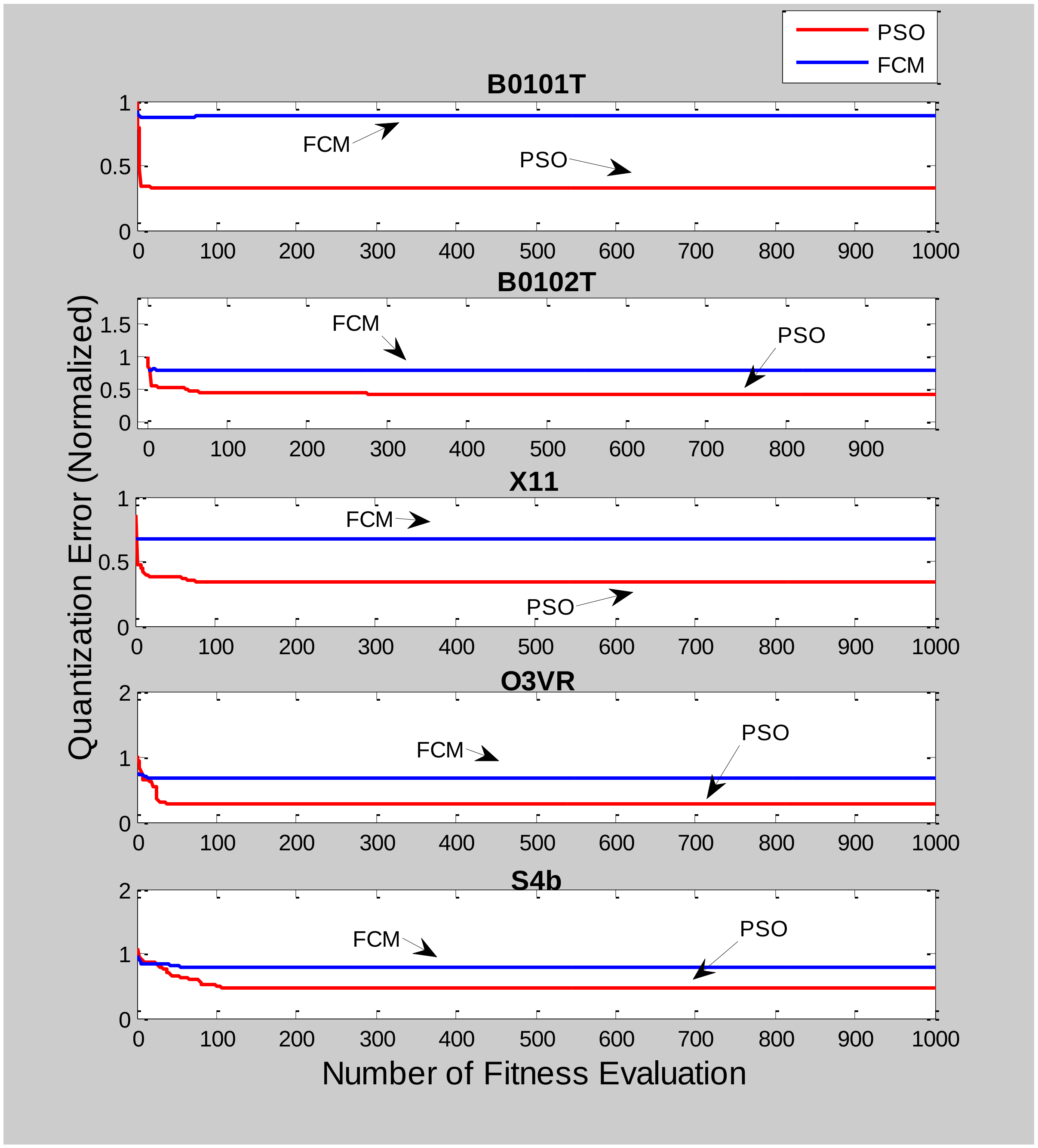


Figure 14 Fuzzy *C*-Means versus PSO Clustering Other Datasets

Figure 13 and Figure 14 present important results on the compared clustering algorithms. The first result that we can observe is due to the local minima problems of the FCM algorithm, no matter how long we keep the algorithm running, there is no improvement in terms of clustering after approximately 100 fitness evaluations. However, since the PSO algorithm works in a more explorative way, it may find the global minimum or near global minimum for the clustering. The second important result that proposed is, also stated by [33], [34], and [58], the FCM algorithm does not deal with proper locations of the cluster centers. That is the algorithm does not try to optimize the cluster centroids during iteration. The centers are recalculated every time according to the membership matrix and due to the influence of partial memberships of the data members, the cluster centers tend to move towards the center of all data points. The improper movement of centroid locations could be observed from the ups and downs of the quantization error value in some of the datasets.

Although the PSO algorithm has superior capabilities in providing better clustering, it comes with a cost. Since the PSO algorithm is computationally more expensive than both *K*-means and FCM, the fast convergence properties of FCM and *K*-means might still be useful for some of the applications where there is not enough computational power.

The final quantization errors obtained at the end of each set of algorithm run are listed in Table 3. It can be seen from the final results that PSO is able to find a better center locations than FCM in every dataset that has been clustered.

Table 3 Data and Quantization Errors

DATA	***Quantization Error***	
	FCM	*PSO*
Diabetes	0.57	0.18
Ionosphere	1.33	0.60
Medical	1.02	0.78
Liver	0.42	0.20
Cancer	1.31	0.88
X11	0.66	0.33
O3VR	0.67	0.27
S4b	0.79	0.47
B0101T	0.88	0.52
B0102T	0.79	0.43

3.2 Comparison of Classification Methods

In this section, the classification results with the RBFN classifier where the hidden layer centers are found both using PSO clustering, named PSO-RBFN, and FCM clustering, FCM-RBFN are presented. In addition to these, FFSVM and IFFSVM classifier results are presented on the same datasets using the same training, validation and testing data. The first part of this section presents the description of the EEG datasets and the feature extraction method utilized. The second part presents the tabulated performance results of classification methods explored.

3.2.2 Results of FFSVC and IFFSVC Algorithms

In this section, the results for Fuzzy Functions Support Vector Classifier and Improved Support Vector Classifiers are presented. Both algorithms are used in classification of multiple standard datasets as discussed in the previous clustering section. In order to implement the Support Vector Machine Algorithm for the FFSVC and IFFSVC, LIBSVM [60] software libraries for Matlab are utilized.

Before running these algorithms, some parameters need to be set. One of these parameters is the *regularization constant* (*Creg*). This parameter comes from the Support Vector Machine classifier and adjusts the position of the hyperplane from the support vectors [24]. Within the training part of the algorithm, this parameter is selected as powers of two within the range $[2^{-4}, 2^{-3}, 2^{-2}, \ldots 2^{8}]$ as suggested by [52]. Another parameter that needs to be selected is the number of clusters, *c*. The number of clusters is searched exhaustively starting from two to the number of features in each dataset. The third parameter is the fuzzification constant *m* for the FCM algorithm. This is selected from the range of $[1.2, 1.3, \ldots, 2]$. All of these parameters are determined by performing a grid search and a combination of the parameters that gives the highest cross validation accuracy is chosen for testing of the classifier. Table 4 includes training, validation and testing data ratios that each dataset is partitioned. The algorithm is run 10 times for each dataset and the average results are presented in Table 5 and Table 6. Note that for most of the datasets, there was not large variation encountered in the performance accuracy.

Table 4 Dataset Names and Corresponding Train, Validation, and Test Data ratios

Name	*# Train Data*	*# Validation Data*	*# Test Data*
Diabetes	384	192	192
Liver	172	86	86
Ionosphere	174	87	87
Medical	500	250	250
Cancer	320	160	160
X11	540	270	270
O3VR	238	118	123
S4b	540	270	270
B0101T	80	20	20
B0102T	80	20	20

Table 5 Fuzzy Function Support Vector Classifier Results

Data	*Percentage*	*FFSVC*
Diabetes	79.30	Creg=1 c=8 m= 1.2
Liver	76.74	Creg=32 c=5 m= 1.7
Ionosphere	97.70	Creg=2 c=7 m= 1.4
Medical	73.60	Creg=16 c=2 m= 1.3
Cancer	99.38	Creg=32 c=5 m= 2
X11	78.80	Creg=16 c=5 m= 1.8
O3V	95.12	Creg=2^-4 c=3 m= 2.1
S4b	75.79	Creg=2^5 c=8 m= 1.2
B0101T	85	Creg=1 c=4 m= 1.7
B0102T	61.25	Creg=16 c=8 m= 1.7
Average	80.36	

Table 6 Improved Fuzzy Functions Support Vector Classifier Results

Data	*Percentage*	*IFFSVC*
Diabetes	80.83	Creg= 1 c= 7 m= 1.2
Liver	76.98	Creg= 4 c= 4 m= 2.1
Ionosphere	98.85	Creg= 50 c= 4 m= 1.4
Medical	74.40	Creg= 0.25 c= 2 m= 2
Cancer	99.50	Creg= 32 c= 5 m= 2
X11	80	Creg= 0.25 c= 2 m= 1.6
O3V	95.12	Creg=32 c= 5 m= 1.5
S4b	78.11	Creg= 2^3 c= 7 m= 1.6
B0101T	90	Creg= 2^0 c= 4 m= 1.2
B0102T	84	Creg= 2^6 c= 8 m= 1.7
Average	84.25	

Table 5 and Table 6 show that there is a significant difference between the IFFSVC and FFSVC. This implies that working on the improvement of the membership values may help increasing the classification accuracy of data. On average, IFFSVC performs approximately 4 % better than FFSVC

3.2.3 Proposed Classification Algorithm Runs (PSO-RBFN)

In the previous section, classification results of FFSVC and IFFSVC algorithms were presented. In this section our results related to Radial Basis Function Networks Classifier where the hidden layer units are found by utilizing both PSO clustering (PSO-RBFN) and FCM Clustering (FCM-RBFN) are analyzed. The reason that the two different clustering algorithms were explored is to show the clustering effects on the classification performance of RBFN classifier. As it was explained earlier, the better clustering with the RBFN should result in higher classification accuracies.

3.2.3.1 Exhaustive PSO-RBFN

The classification results of 1000 iterations with PSO-RBFN and FCM-RBFN algorithms are presented in Table 7. The runs are performed 10 times. The initialization of the particles in PSO is done according to a random selection pattern, considering the upper limits and the lower limits of each of the feature values in the dataset. The initial centers of the FCM clustering is taken the same as one of the particles in the swarm so we guarantee that the FCM clustering starts at the same location as the PSO clustering. The rest of the particles are different from the FCM initial centers.

Table 7 Exhaustive PSO-RBFN versus Exhaustive FCM-RBFN Classification Results

	EFCM-RBFN											
# of Runs	1	2	3	4	5	6	7	8	9	10	**Average**	**Variance**
Diabetes	78.65	77.08	79.17	79.17	78.65	79.69	80.21	80.21	80.21	79.17	**79.22**	**0.93**
Liver	67.44	67.44	67.44	72.09	67.44	76.74	73.26	67.44	67.44	74.42	**70.12**	**13.24**
Ionosphere	95.40	94.25	94.25	93.10	94.25	94.25	95.40	94.25	96.55	94.25	**94.60**	**0.90**
Cancer	100.00	100.00	100.00	99.38	99.38	100.00	100.00	99.38	100.00	100.00	**99.81**	**0.09**
Medical	73.60	73.60	73.60	73.60	73.60	73.60	73.60	73.60	74.00	74.00	**73.68**	**0.03**
X11	78.89	78.89	78.89	78.89	78.89	78.89	78.89	78.89	78.89	78.89	**78.89**	**0.00**
O3V	94.17	97.50	94.17	94.17	97.50	94.17	97.50	97.50	94.17	94.17	**95.50**	**2.96**
S4b	79.63	79.63	79.63	80.37	79.63	80.37	79.63	79.63	79.63	80.37	**79.85**	**0.13**
B0101T	85.00	85.00	85.00	85.00	85.00	85.00	85.00	85.00	85.00	85.00	**85.00**	**0.00**
B0102T	70.00	70.00	70.00	70.00	70.00	70.00	70.00	70.00	70.00	70.00	**70.00**	**0.00**

	EPSO-RBFN											
# of Runs	1	2	3	4	5	6	7	8	9	10	**Average**	**Variance**
Diabetes	79.17	80.21	78.65	76.56	79.17	79.69	81.25	80.21	80.21	77.60	**79.27**	**1.92**
Liver	70.93	68.60	75.58	70.93	69.77	66.28	70.93	74.42	68.60	70.93	**70.70**	**7.45**
Ionosphere	93.10	97.70	95.40	98.85	**100.00**	98.85	97.70	98.85	96.55	96.55	**97.36**	**4.13**
Cancer	98.75	99.38	99.38	100.00	98.75	100.00	99.38	100.00	99.38	98.13	**99.31**	**0.39**
Medical	71.60	71.20	70.00	71.20	70.80	70.00	71.20	72.00	72.00	69.60	**70.96**	**0.72**
X11	78.89	76.30	79.26	78.89	79.26	80.74	75.93	79.26	77.04	76.67	**78.22**	**2.58**
O3V	96.67	97.50	95.00	95.00	97.50	95.00	95.83	95.00	95.83	96.67	**96.00**	**1.05**
S4b	77.41	81.48	79.63	76.67	77.04	80.37	81.85	79.63	78.52	78.89	**79.15**	**3.20**
B0101T	85.00	85.00	85.00	85.00	85.00	90.00	85.00	85.00	85.00	85.00	**85.50**	**2.50**
B0102T	75.00	75.00	80.00	75.00	85.00	70.00	80.00	85.00	80.00	80.00	**78.50**	**22.50**

Parameter optimization of RBFN includes finding hidden layer kernel centers and the optimum number of hidden layer units that might generalize the data well enough. During the parameter optimization the RBFN, the number of hidden layer units is searched *exhaustively*, starting from the same number of feature space dimension of the dataset and increasing until two times of the dataset dimension. Thus, we call this PSO-RBF combined algorithm as Exhaustive PSO-RBFN, EPSO-RBFN. The cross validation is done according to the average result of both training and testing performances. The reason for this is to avoid learning the validation data indirectly through the training process of the RBFN classifier.

After the hidden layer centers are found by the clustering algorithm, the variance values for each Gaussian hidden units are searched within the range [0.1, 0.2, 0.3, …, 10, 20, 30, 40, 50]. The RBFN parameters at the number of hidden layer units and variance that gives the highest average training and validation result is selected for testing of the classifier on the new data. A simple flowchart representing the RBFN classifier parameter selection, training, and testing phases is given in Figure 15.

The Number of Particles for each dataset is adjusted according to a formula in equation (18) as suggested by [40].

$$Npc = 10 + 2\sqrt{D}\ ,\ D = numberOfFeatures * NumberOfClusters \tag{18}$$

The fitness function of PSO clustering is selected as quantization error and fitness values of the particles are evaluated according to this error during PSO runs.

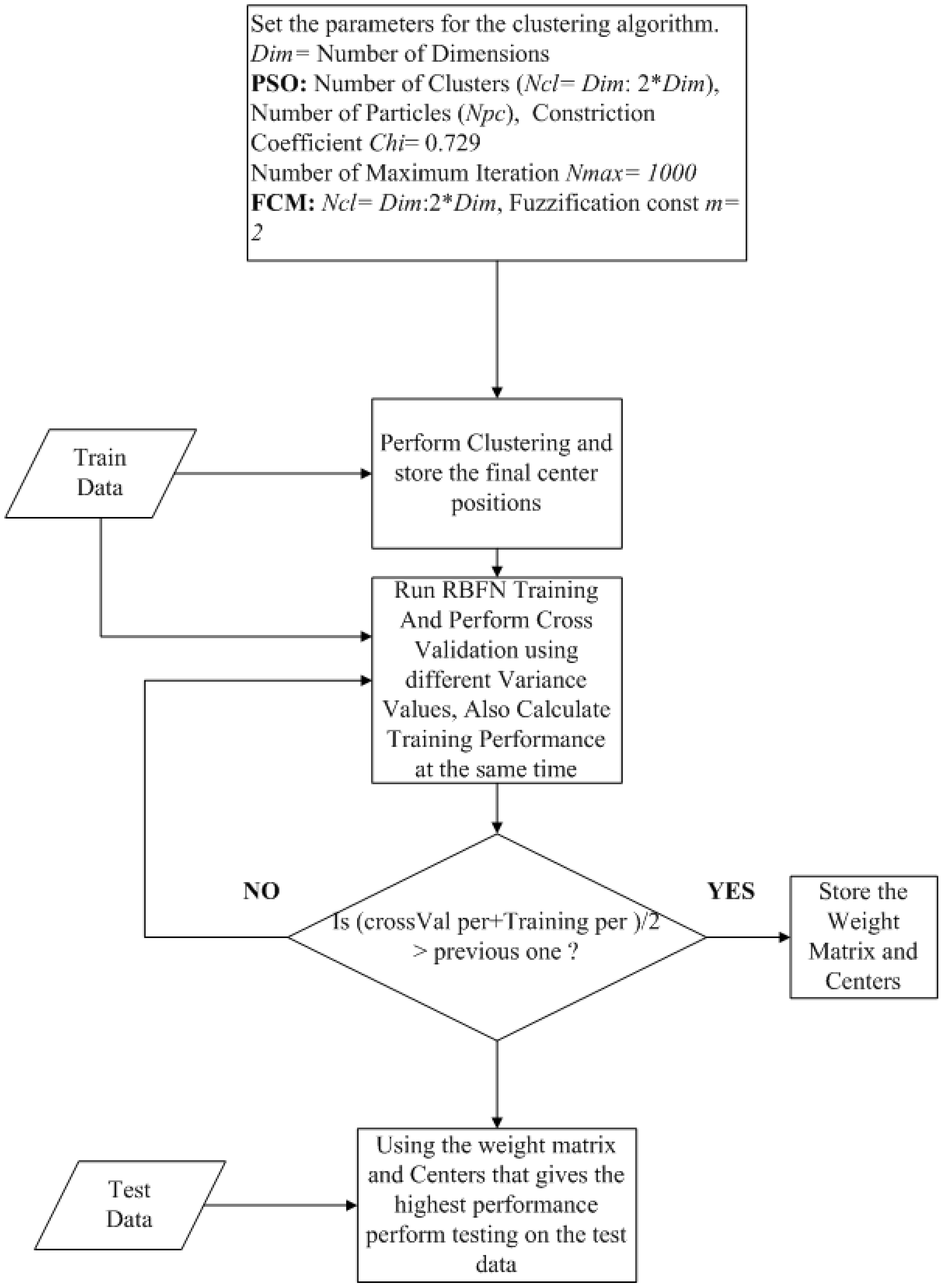

Set the parameters for the clustering algorithm.
Dim= Number of Dimensions
PSO: Number of Clusters (Ncl= Dim: 2*Dim), Number of Particles (Npc), Constriction Coefficient Chi= 0.729
Number of Maximum Iteration Nmax= 1000
FCM: Ncl= Dim:2*Dim, Fuzzification const m= 2
Train Data
Perform Clustering and store the final center positions
Run RBFN Training And Perform Cross Validation using different Variance Values, Also Calculate Training Performance at the same time
NO
YES
Is (crossVal per+Training per)/2 > previous one ?
Store the Weight Matrix and Centers
Test Data
Using the weight matrix and Centers that gives the highest performance perform testing on the test data

Figure 15 Radial Basis Function Networks Parameter Selection and Training Flowchart

When the results are analyzed, it can be seen that the PSO clustering algorithm does significantly better for the *B0102T* dataset. Almost every data run exceeds the EFCM-RBF performance. Another promising result that can be observed is that the PSO clustering helps RBFN to obtain the *highest performances* over the entire data runs for each datasets. For example, in the Ionosphere dataset, it is observed that the performance on the fifth run for PSO-RBFN has reached to 100 percent where it is significantly better than any other FCM-RBFN run.

For some of the datasets, the FCM algorithm produced the same performance for all the runs. The main reason for the behavior is that FCM iterates over the original data and is prone to get stuck in local minima. However, since the PSO clustering works in a more explorative way, the PSO clustering brings more variation to the clustering that can reach the maximum performance in overall.

3.2.3.3 Hybrid PSO-RBFN

During these data runs the PSO algorithm so that in addition to optimizing the quantization error, it might also optimize some of the RBFN parameters and help parameter selection. Therefore an additional dimension is added to the particles that includes different variance values for hidden units of the RBFN. In addition, the fitness function is modified so that the validation and training performances of RBFN are also taken into consideration. This way, clustering is optimized by employing the PSO algorithm and the selection of the RBFN parameters is also optimized. Thus, this PSO-RBFN combined algorithm is called the *Hybrid* PSO-RBFN, HPSO-RBFN.

As particles have both cluster centers and variance for each cluster, we would like to explore parameters of PSO (maximum iterations and constriction constant, χ, *Chi*) and different cost functions to determine RBF performance could be incorporated in the PSO cost function. Equations (19), (20) and (21) show the three cost functions explored regarding the RBF training and validation performance incorporated into the PSO cost function.

Equation (19) shows the fitness function of the modified PSO clustering algorithm.

$$f_1 = 1/quantization\ error + \ pt \tag{19}$$

$$f_2 = 1/quantization\ error + (pt + pv)/2 \tag{20}$$

$$f_3 = 1/quantization\ error + 0.9 * pt + 0.1 * pv \tag{21}$$

In the cost functions, *pt*, is the RBFN training performance and *pv* is the RBFN validation performance. The validation performance is calculated by using the weights obtained in the training phase. We have also explored the PSO parameters: maximum iterations (100 and 1000) and χ coefficient (0.729 and 0.829). By increasing the χ coefficient we would like to increase the exploration ability of the PSO with the hope of achieving global solution or better local solutions. Table 8 presents the results of the HPSO-RBFN runs using the fitness function in equation (19) where only RBF training performance is added to the quantization error. For each dataset, we present average performance, maximum performance, and variance of 10 runs. We have done similar exploration based on the cost function in equation (21) where the validation performance of the RBFN is also incorporated by 10 %. The results of the HPSO-RBFN runs with this cost function are presented in Table 9 .

Table 8 Classification Performance for Hybrid PSO-RBFN with Quantization Error and RBF-Training Performance, μ: mean, *Max*: Maximum Performance Value, σ: Standard Deviation, χ: Constriction Coefficient

	Step: 100 χ: 0.729			Step: 100 χ: 0.829			Step: 1000 χ: 0.729			Step: 1000 χ: 0.829		
Data	μ	Max	σ	μ	Max	σ	μ	Max	σ	μ	Max	σ
Diabetes	78.44	**80.73**	5.38	**79.84**	82.29	1.63	78.13	80.21	2.53	78.33	81.25	**7.49**
Liver	69.07	**73.26**	16.29	**68.95**	74.42	23.75	66.86	73.26	21.41	67.33	74.42	**14.86**
Ionosphere	97.59	**100.00**	3.66	**97.36**	100.00	3.83	98.28	100.00	1.54	98.97	100.00	**0.72**
Medical	71.84	**74.80**	2.00	**71.20**	72.40	1.35	71.52	74.80	2.52	71.52	74.00	**2.06**
Cancer	98.94	**100.00**	1.22	**99.44**	100.00	0.21	99.63	100.00	0.10	99.25	100.00	**0.42**
X11	76.59	**84.07**	17.34	**76.85**	80.37	9.27	73.07	78.52	7.90	77.19	81.48	**8.75**
03VR	95.45	**97.56**	1.94	**94.63**	96.75	0.91	95.04	95.93	0.51	93.82	96.75	**3.85**
S4b	77.63	**81.48**	5.71	**75.26**	80.37	10.69	77.44	81.48	7.91	74.00	76.67	**3.10**
B0101T	72.00	**90.00**	112.22	**76.00**	90.00	65.56	73.00	85.00	73.33	70.50	75.00	**13.61**
B0102T	67.50	**80.00**	56.94	**71.00**	75.00	26.67	70.00	80.00	61.11	66.00	75.00	**32.22**
Average	80.50	**86.19**	22.27	**81.05**	85.16	14.39	80.30	84.92	17.89	79.69	83.46	**8.71**

Table 9 Classification Performance for Hybrid PSO with Quantization Error and RBF Training and Validation Performance ($0.9T + 0.1V$), μ: mean, *Max*: Maximum Performance Value, σ: Standard Deviation, χ: Constriction Coefficient

	Step: 100 χ: 0.729			Step: 100 χ: 0.829			Step: 1000 χ: 0.729			Step: 1000 χ: 0.829		
Data	μ	Max	σ	μ	Max	σ	μ	Max	σ	μ	Max	σ
Diabetes	**78.59**	**81.25**	2.56	78.28	80.73	1.87	79.17	80.73	2.77	78.91	81.77	**3.51**
Liver	**69.88**	**72.09**	4.64	68.14	75.58	9.37	71.63	74.42	2.76	70.23	73.26	**6.97**
Ionosphere	**97.82**	**100.00**	2.77	97.82	100.00	1.60	97.24	100.00	5.05	96.90	98.85	**2.66**
Medical	**71.24**	**72.80**	0.94	72.48	73.60	0.78	72.32	74.00	0.81	70.84	72.80	**1.90**
Cancer	**99.50**	**100.00**	0.24	99.50	100.00	0.07	99.69	100.00	0.11	99.63	100.00	**0.10**
X11	**76.93**	**79.26**	6.28	78.11	80.37	4.46	74.63	78.89	7.60	77.44	80.00	**4.56**
03VR	**94.88**	**97.56**	2.65	95.93	97.56	1.03	94.55	95.12	0.30	94.39	97.56	**2.42**
S4b	**77.26**	**82.22**	6.86	76.44	80.00	4.30	75.78	80.37	14.91	77.93	81.11	**7.32**
B0101T	**80.50**	**95.00**	69.17	78.00	95.00	90.00	70.50	80.00	19.17	73.50	80.00	**11.39**
B0102T	**70.50**	**85.00**	96.94	66.00	80.00	60.00	67.50	80.00	134.72	69.50	75.00	**24.72**
Average	**81.71**	**86.52**	19.31	81.07	86.28	17.35	80.30	84.35	18.82	80.93	84.03	**6.55**

In Table 8 and Table 9, we have marked the best performing parameter sets by making the corresponding column bold. Three columns in each table for the best mean performance, best maximum performance, and the best variance are marked. The maximum performance is the highest performance among 10 runs. The variance is the performance variance of the 10 runs for each dataset. As can be seen from Table 8 and Table 9, better results are achieved when the χ coefficient is 0.729 as described in the PSO literature [41]. However, by making χ parameter larger, we were able to make the PSO algorithm explore the space more as the maximum performance is slightly higher and the variance is smaller. This suggests that the algorithm is hitting better local solutions consistently. When the PSO steps is increased to 1000 iteration, it can be seen that the variance becomes very small as the algorithm has more time to explore the space. It is believed that this increases the chance to obtain better solutions as it does not let PSO settle into a local solution. However, when the algorithm is run longer, the overall performance is lower probably because some over training is experienced. Thus, it is believed that the best PSO parameter pair is 100 iterations and χ coefficient of 0.729. It has also been explored a cost function based on the average of training and validation performances of RBFN as shown in equation (20). Table 10 compares the HPSO-RBFN algorithms based on their cost function with a maximum PSO iteration of 100 and χ coefficient of 0.729. As can be seen from Table 10, the best performances are achieved when 10 % of the validation is factored into the cost function. However, when validation performance and training performance is equally weighted, the robustness of the algorithm is improved as the variance got significantly smaller. So, If the average performance, maximum performance, and the variance are analyzed, it can be concluded that the algorithm is robust and reasonably successful when both training and validation performances of RBFN are equally weighted.

Table 10 Performance Comparison of Different PSO Cost Functions

	Step: 100 χ: 0.729								
Cost Type	Training Only			0.9Training + 0.1Validation			0.5Training + 0.5Validation		
	μ	Max	σ	μ	Max	σ	μ	Max	σ
Diabetes	78.44	80.73	5.38	**78.59**	**81.25**	2.56	79.22	80.73	**0.87**
Liver	69.07	73.26	16.29	**69.88**	**72.09**	4.64	71.98	76.74	**11.25**
Ionosphere	97.59	100.00	3.66	**97.82**	**100.00**	2.77	97.70	100.00	**2.06**
Medical	71.84	74.80	2.00	**71.24**	**72.80**	0.94	71.32	72.80	**0.75**
Cancer	98.94	100.00	1.22	**99.50**	**100.00**	0.24	99.19	100.00	**0.35**
X11	76.59	84.07	17.34	**76.93**	**79.26**	6.28	76.30	78.89	**2.87**
03VR	95.45	97.56	1.94	**94.88**	**97.56**	2.65	94.47	96.75	**2.76**
S4b	77.63	81.48	5.71	**77.26**	**82.22**	6.86	77.63	81.11	**6.53**
B0101T	72.00	90.00	112.22	**80.50**	**95.00**	69.17	84.50	90.00	**19.17**
B0102T	67.50	80.00	56.94	**70.50**	**85.00**	96.94	60.50	70.00	**69.17**
Average	80.50	86.19	22.27	**81.71**	**86.52**	19.31	81.28	84.70	**11.58**

3.2.3.3 Pure PSO-RBFN

After exploring a Hybrid PSO-RBFN where both clustering and RBFN variance exploration are optimized at the same time, we would like to explore yet another PSO-RBFN algorithm where both clustering and RBFN variance optimization are done by two PSO algorithms sequentially. That is, first PSO algorithm does clustering to minimize the quantization error. Then, the second PSO algorithm optimizes corresponding hidden layer variances in the RBFN to maximize the performance. Thus, we call this combined PSO-RBFN algorithm as *Pure* PSO-RBFN, PPSO-RBFN. We have chosen the number of clusters being equal to the number of features in each dataset since we experienced best clustering results when the number of clusters is equal to the number of features in the dataset.

Table 11 summarizes the results of the runs for maximum iteration of 100 and 1000 and χ coefficients of 0.729 and 0.829.

The cost function of the first PSO is the quantization error while the cost function of the second PSO is mostly based on the training performance and higher validation performances are preferred as can be seen in equation (22).

$$f_4 = pt + \frac{1}{abs(pt - pv) + 1} \tag{22}$$

Table 11 Classification Results for Pure PSO-RBFN

	Step: 100 Chi: 0.729			Step: 100 Chi: 0.829			Step: 1000 Chi: 0.729			Step: 1000 Chi: 0.829		
	μ	Max	σ	μ	Max	σ	μ	Max	σ	μ	Max	σ
Diabetes	**80.16**	81.77	0.81	78.91	80.73	**1.10**	79.27	**81.25**	1.37	79.58	80.73	0.89
Liver	**72.33**	74.42	1.74	72.21	73.26	**1.94**	71.40	**74.42**	3.67	72.09	74.42	2.10
Ionosphere	**97.82**	98.85	0.72	97.93	98.85	**0.82**	98.05	**100.00**	1.19	96.09	100.00	4.46
Medical	**71.20**	72.00	0.43	71.04	72.00	**0.65**	70.96	**71.60**	0.29	71.28	72.40	0.46
Cancer	**99.63**	100.00	0.10	99.56	100.00	**0.18**	99.75	**100.00**	0.10	99.56	100.00	0.09
X11	**76.70**	77.78	0.32	76.30	77.78	**0.67**	77.11	**79.63**	1.88	77.59	78.89	1.14
O3V	**85.12**	90.24	8.82	83.66	86.18	**2.12**	85.53	**89.43**	8.05	84.07	86.18	2.82
S4b	**75.15**	76.67	0.84	75.04	76.30	**1.35**	75.22	**75.93**	0.32	75.44	75.93	0.15
B101	**90.00**	90.00	0.00	89.50	90.00	**2.50**	89.00	**90.00**	4.44	89.00	90.00	4.44
B102	**62.00**	65.00	6.67	60.50	65.00	**2.50**	62.50	**65.00**	6.94	59.00	60.00	4.44
Average	**81.01**	82.67	2.04	80.46	82.01	**1.38**	80.88	**82.73**	2.83	80.37	81.85	2.10

As can be seen from Table 11, the Pure PSO-RBFN algorithm is the most robust algorithm among all the proposed PSO based classification algorithms. The main reason for this is that the algorithm first clusters the data as crispy as possible. Then it optimizes the variances of the hidden layers to maximize the classification performance based on the crisp clustering. As the first PSO pushes the algorithm to a crispier clustering, adjusting the variances of the hidden layers can only maximize the classification performance based on the crisp clustering. Thus, the algorithm ends up around the same local classification performance based on the crisp clustering. Even though the maximum performance of this algorithm is lower than the Hybrid PSO-RBFN, the average performance is comparable or better than the Hybrid PSO-RBFN when the algorithm is run 1000 steps. For some applications, this would be preferable as it presents very robust results with an average variance of 1.38 which is very small compared to the best previously achieved variance of 6.55

3.2.3.4 PSO Fuzzy Functions Support Vector Classifier

As we have observed from the classification performances between FFSVC and IFFSVC, improving the membership values might also increase the classification performance. This prompted the idea searching for a way of improving the membership values utilizing the PSO clustering algorithm. As explained in the previous section, FCM Clustering is not good at finding good cluster centers. Therefore it has been explored whether this drawback of the FCM can be alleviated by the help of PSO so that the membership values might be improved.

Initially the PSO clustering algorithm is run on the datasets, the modified membership values based on the final cluster centers are obtained using the standard membership update formula of the FCM. After the modified membership values are calculated, the rest of the standard FFSVM technique is applied. Therefore this technique is called PFFSVC (PSO Fuzzy Functions Support Vector Classifier) as the clustering is done by PSO instead of FCM. The results for the PFFSVC approach with 10 data runs are presented in Table 12.

Table 12 PSO Fuzzy Functions Support Vector Classifier Results

	P-FFSVC Results											
# of Runs	1	2	3	4	5	6	7	8	9	10	**Average**	**Variance**
Diabetes	79.68	**80.23**	79.68	80.23	80.23	79.68	80.23	79.68	80.23	80.23	80.01	0.08
Liver	**77.90**	77.90	76.74	77.90	75.58	77.90	77.90	77.90	77.90	77.90	77.55	0.61
Ionosphere	**98.85**	98.85	98.85	97.70	97.70	98.85	98.85	97.70	98.85	98.85	98.51	0.31
Medical	**74.40**	74.40	74.40	74.40	74.40	74.40	74.40	74.40	74.40	74.40	74.40	0.00
Cancer	**100.00**	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	0.00
X11	**82.20**	80.00	81.85	82.20	80.00	79.62	80.37	79.25	80.37	81.11	80.70	1.16
O3V	**95.12**	93.49	95.12	94.31	95.12	95.12	95.12	95.12	95.12	95.12	94.88	0.30
S4b	76.66	76.29	**77.04**	75.56	76.30	76.30	75.19	74.81	75.56	75.56	75.92	0.49
B101	95.00	90.00	90.00	**95.00**	95.00	95.00	95.00	95.00	95.00	95.00	94.00	4.44
B102	**80.00**	75.00	75.00	80.00	80.00	75.00	80.00	80.00	80.00	80.00	78.50	5.83
Average:											**85.44**	**1.32**

Comparing the results in Table 12 with Table 5 and Table 6, the classification performances for almost every run are greater than FFSVM and also there are cases that most of the classification results reach or pass the results of the IFFSVM classifier. This shows that the drawback of FCM algorithm in finding good cluster centers could be alleviated using an initial seed algorithm such as PSO that does better clustering.

Finally, Table 13 presents the best performances of all the classification algorithms implemented in this thesis.

Table 13 Best Performances of All the Classification Algorithms

	FFSVC			IFSVC			EPSO-RBFN			EFCM-RBFN		
Datasets	μ	Max	σ	μ	Max	σ	μ	Max	σ	μ	Max	σ
Diabetes	79.3	79.69	0.06	80.83	81.77	1.37	79.27	81.25	1.92	79.22	80.21	0.93
Liver	76.74	76.74	0	76.98	79.07	0.54	70.7	75.58	7.45	70.12	76.74	13.24
Ionosphere	97.7	97.70	0	98.85	98.85	0	97.36	100	70.12	94.6	96.55	0.9
Medical	73.60	73.60	0	74.4	74.4	0	70.96	72	0.72	73.68	74	0.03
Cancer	99.38	99.38	0	99.5	100	0	99.31	100	0.39	99.81	100	0.09
X11	78.80	80.00	0.26	80	80	0	78.22	80.74	2.58	78.89	78.89	0
O3V	95.12	95.12	0	95.12	95.12	0	96	97.5	1.05	95.5	97.5	2.96
S4b	75.79	77.04	0.7	78.11	78.15	0.01	79.15	81.85	3.2	79.85	80.37	0.13
B101	85	85	0	90	90	0	85.5	90	2.5	85	85	0
B102	61.25	65	5.36	84	95	60	78.5	85	22.5	70	70	0
Average	82.268	82.9268	0.638	85.779	87.236	6.192	83.497	86.392	11.243	82.667	83.926	1.828

	HPSO-RBFN			PPSO-RBFN			PFFSVC		
Datasets	μ	Max	σ	μ	Max	σ	μ	Max	σ
Diabetes	78.59	81.25	2.56	78.44	80.73	5.38	80.01	80.23	0.08
Liver	69.88	72.09	4.64	69.07	73.26	16.29	77.55	77.9	0.61
Ionosphere	97.82	100	2.77	97.59	100	3.66	98.51	98.85	0.31
Medical	71.24	72.8	0.94	71.84	74.8	2	74.4	74.4	0
Cancer	99.5	100	0.24	98.94	100	1.22	100	100	0
X11	76.93	79.26	6.28	76.59	84.07	17.34	80.7	82.2	1.16
O3V	94.88	97.56	2.65	95.45	97.56	1.94	94.88	95.12	0.3
S4b	77.26	82.22	6.86	77.63	81.48	5.71	75.92	77.04	0.49
B101	80.5	95	69.17	72	90	112.22	94	95	4.44
B102	70.5	85	96.94	67.5	80	56.94	78.5	80	5.83
Average	81.71	86.52	19.31	80.5	86.19	22.27	85.44	86.074	1.322

4. Applications and Experiments

In this section, the applications of previously analyzed classification and clustering techniques are presented. First, an EOG controlled mobile robot system design [61] with Radial Basis Function Networks Classifier is presented. Then, an application that includes design of patient tracking system created utilizing RBFN networks is presented. Finally, a real-time Brain Computer Interface Design and a robot control experiment is presented.

4.1 Electroocculogram Controlled Mobile Robot

In this thesis work, the classification of EOG signal was studied first. A better understanding of the EOG signals can help improving the quality of important applications in the fields such as assistive robotics or human computer interaction. For example, controlling of assistive machines without any joystick mechanism is essential for elderly people who may have lost their muscle control due to the injuries to the spinal cord. For people who are not able to control their muscles below their neck, researchers have been trying to develop new efficient technologies in order to help them to control their wheelchairs using their biosignals taken from the eyes [62], [63], [64].

EOG signals have advantages over the EEG because of its high signal to noise ratio and simple collectability from the eyes. However, it also suffers from several issues such as artifacts and noises added to the signal. The common problems in processing EOG signals include the eye blink artifacts, shifting resting potential artifacts, the effect of fatigue, and environmental conditions. These types of effects change the nature of the signal and make them harder to process. Figure 16 shows the EOG signal taken from the same person but in different periods of times. Since the classification of these signals plays the most important role for the control mechanism, the decision techniques chosen should be considerably robust for the general use. Therefore there is a need for employing machine learning techniques for the correct classification of EOG signals

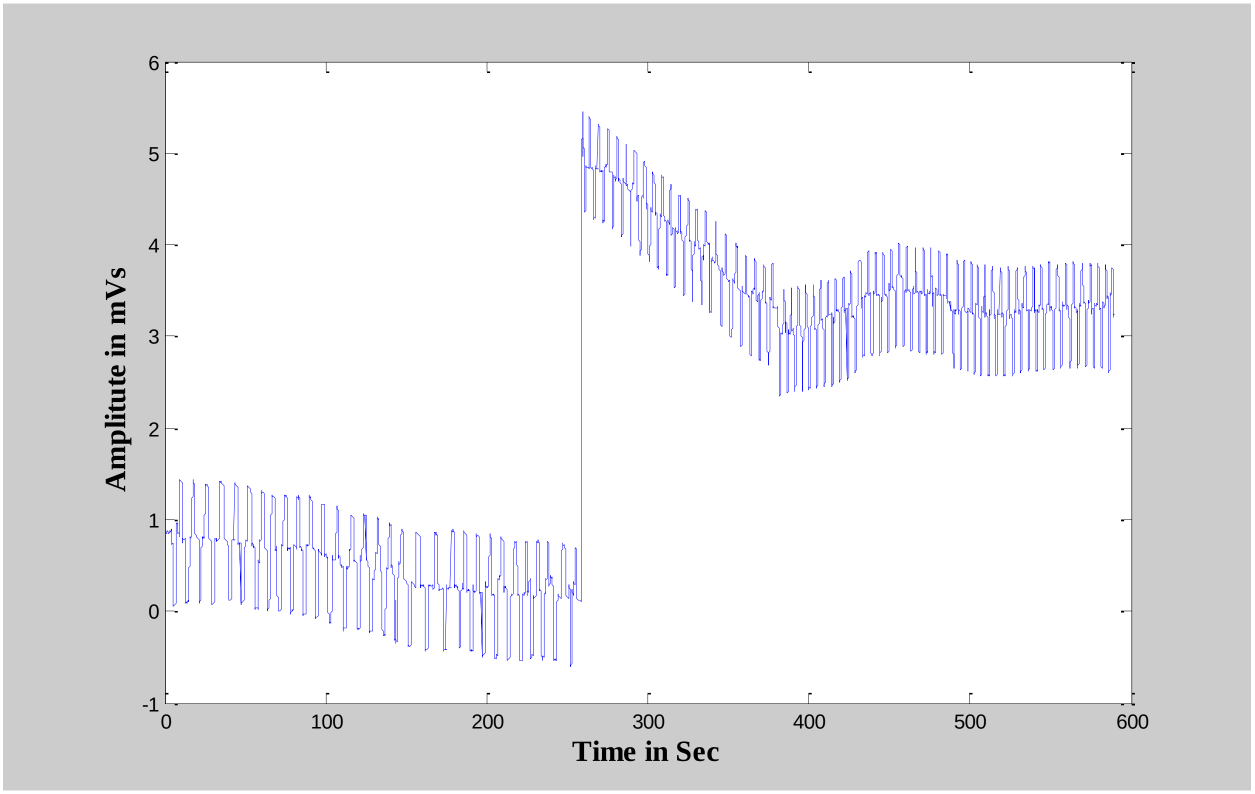


Figure 16 EOG signal data collected from the same person at different times. The movements tested by the subject includes looking at center, left, center, right and center in turn [61]

Considering the previous related work in the literature [62]-[66], most of the work contains an application of pure signal processing techniques without any learning mechanism such as thresholding, differentiation, and Fast Fourier Transforms.

Takashahi et al. [62] uses threshold values for gesture analyses after applying a digital filter to the EOG signal in order to classify eye movements. Wijesome et al. [66] applies time domain analyzes to EOG signal considering the 180 degrees phase shift can be distinguishable enough to decide Left and Right looking of a person and controls the robot accordingly.

Although these techniques give decent performance, it is highly possible that they might suffer due to easily varying characteristics of EOG signals such as shifting resting potential effect as could be observed from Figure 16. Thus, the efficiency of classified signals may be increased by applying various machine learning algorithms that can adopt the change of the signal nature and make the decision robustly.

In our work, the horizontal EOG signal has been collected through one second intervals by utilizing CleveMed BioRadio® [67] with the electrode placement as shown in Figure 17.

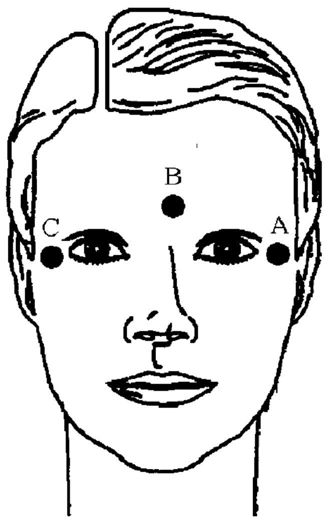
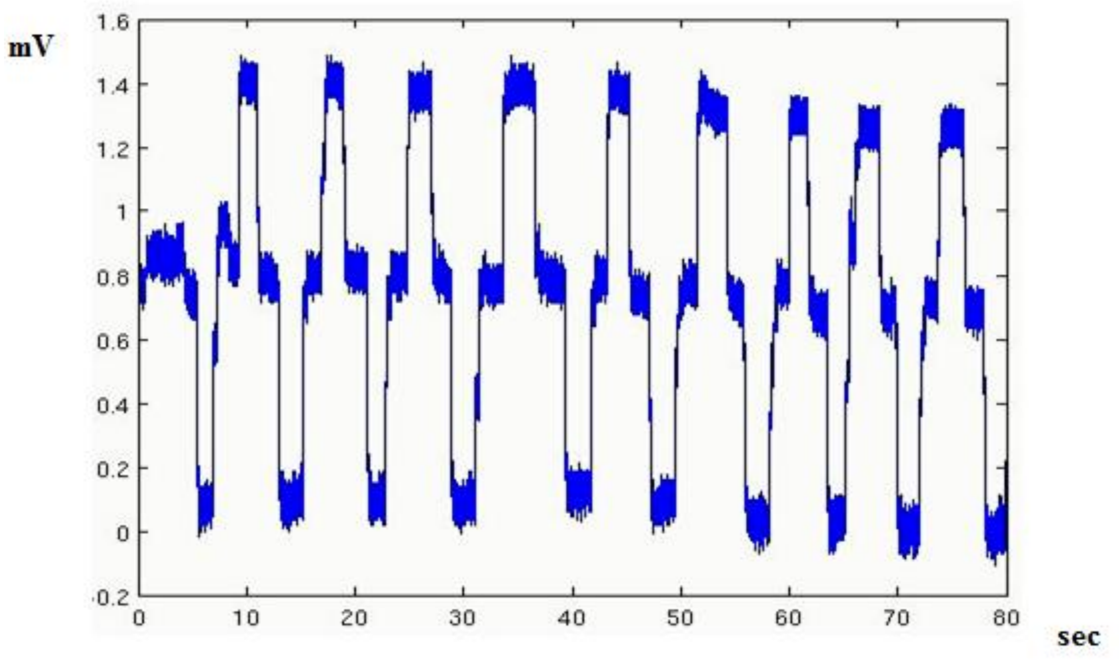


Figure 17 Electrode Placement for EOG Data Signal Collection [8]

The user initially starts looking at the center and moves his or her eyes sequentially to the left first, to the center and to the right in one second of intervals. The timing is supervised by a Power Point program running on the computer screen. During the eye movements, EOG signals are collected and saved into a file to be used in the training phase. The collected signal is labeled according to the sequence the subject is supervised for the training of the system. Since the collected data contains noise and muscle artifacts from by other biopotential signals, a second order Butterworth low pass filter with 100 Hz cutoff frequency is applied in order to get rid of these artifacts. Figure 18 shows the raw on the top data and the low pass filtered data on the bottom.



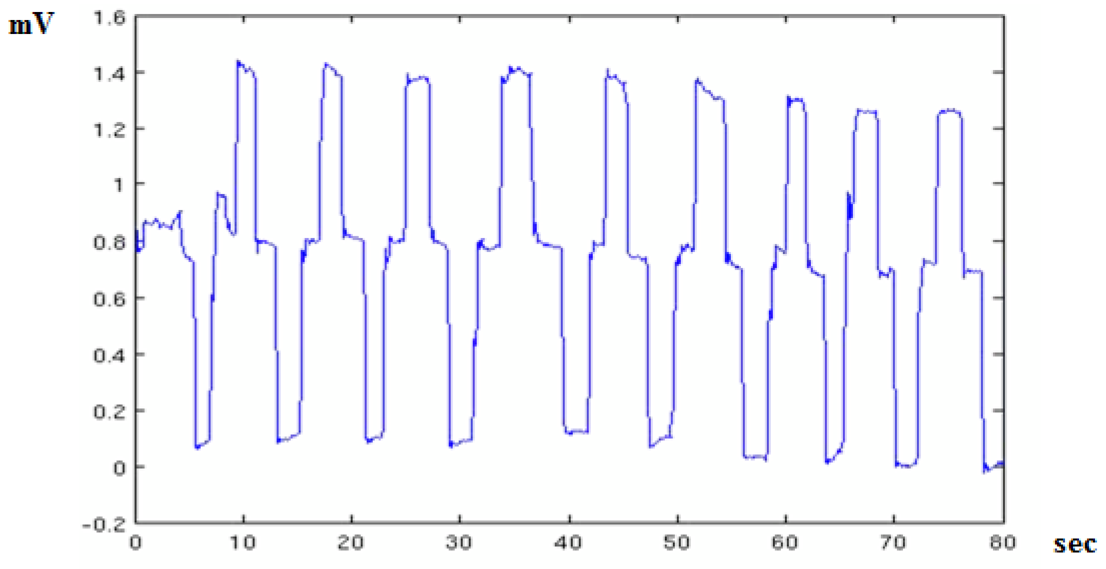


Figure 18 Raw EOG Data and Low Passed Filtered Data

After the collected EOG signal is preprocessed with a low-pass filter, the feature extraction session is started. In humans, EOG signals are linearly proportional to the eye displacements. Since a linear regression of the signal within a specific time interval will result in a slope and an intersection, these could be used as features for the EOG signal and they might give useful information such as the speed and direction of the eye movement. Thus, we have decided to apply linear regression to the signal in one second of intervals as feature extraction method. The linearly regressed EOG signal is shown in Figure 19. The slopes and *y*-intercept of these lines are utilized in the training phase of the system.

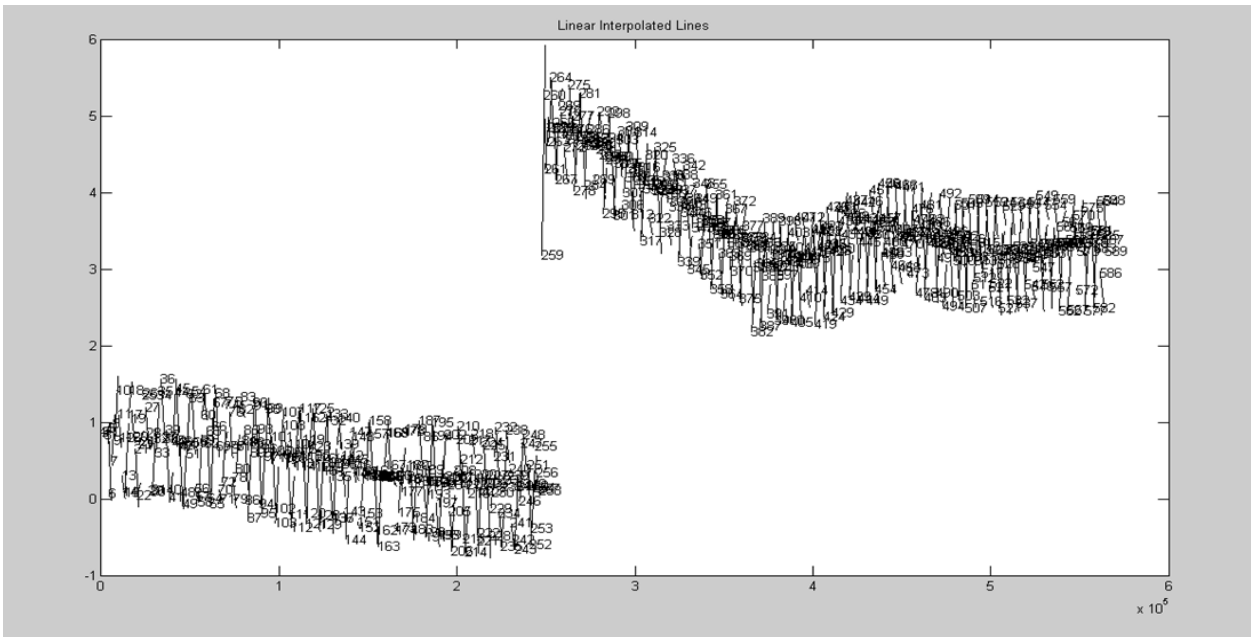


Figure 19 Part of the EOG Signal after Linear Regression

4.1.1 Training of the System and Decision Making

After generating the features for EOG signal, in the classification step we have designed an RBFN classifier as shown in Figure 20. The RBFN in this section, different than the previously designed RBFN in section two, includes multiple output units to be able to classify more than two class labels. The designed RBFN has two inputs and three binary outputs. The two inputs correspond to the features extracted as slope and intercept of the signal (*X_1* and *X_2*). Each of the three output units (*y_1*, *y_2*, and *y_3*) generates binary values in order to decode the decision output for the EOG signal according to the corresponding classes as shown in Table 14.

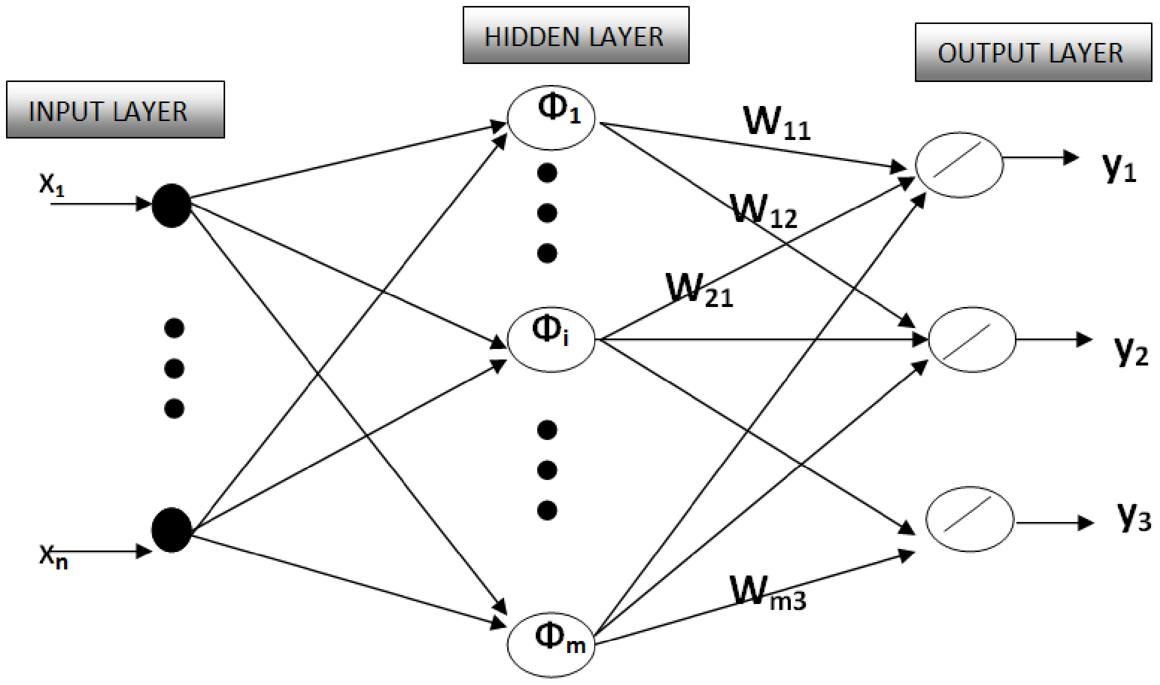


Figure 20 Structural Schematic for EOG RBFN Classifier

Table 14 Classification of Network Outputs

Network Output	**Class**
000	Center
001	Center-Left
010	Left-Left
011	Left-Center
100	Center-Right
101	Right-Right
110	Right-Center
111	Undefined

In order to determine the RBFN hidden layer basis function centers, conventional *K*-Means clustering algorithm is applied. Figure 21 shows the recorded training data features and the results of clustering centers after *K*-means algorithm.

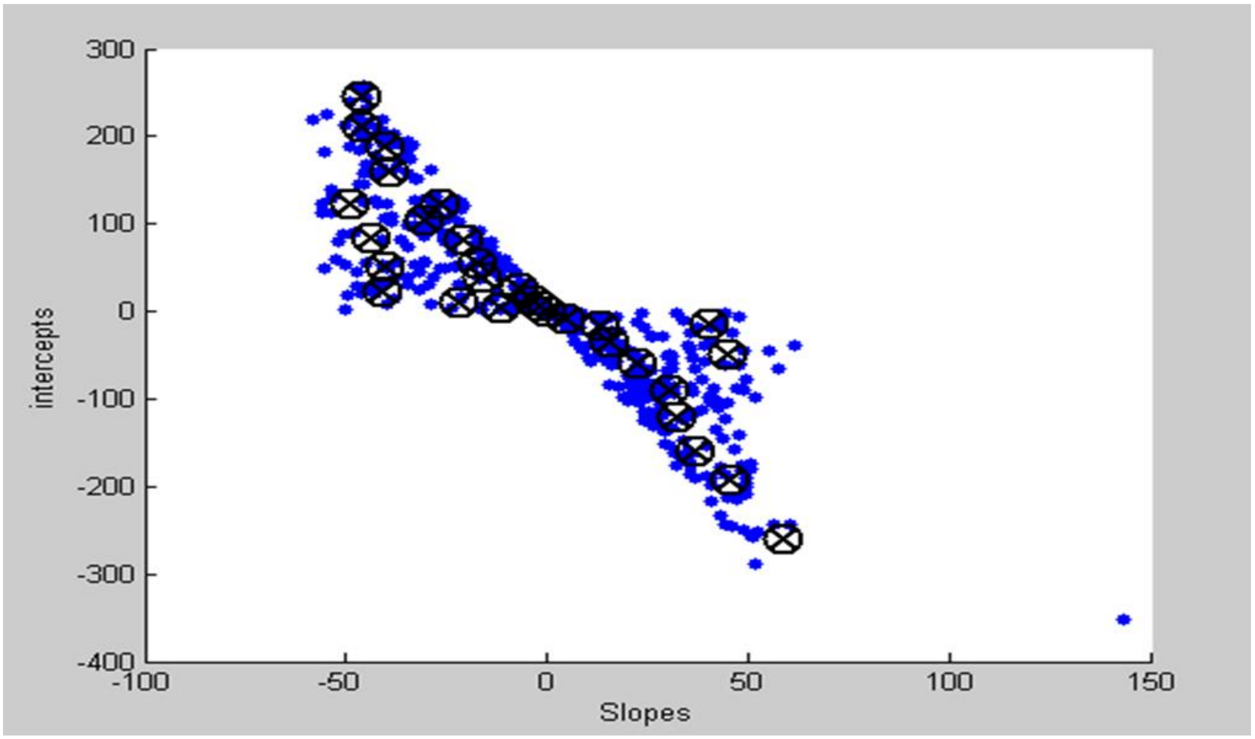


Figure 21 Determining Centers for the Hidden Layer Kernels by *K*-Means Clustering, dots represent the data points in the dataset and circles represent the cluster centers generated by K-Means Clustering

After finding the number of cluster centers that give the highest training performance for the RBFN, the network parameters are stored in order to be used for the test data which is the data that is not used in the training session of the classifier. For this application the variances of each hidden layer units are calculated from the data itself according to the cluster centers determined by the clustering algorithm. The binary outputs of the RBFN classifier are generated by a hard limiter, placed at the output units and thresholds the output layer values by 0.5. During the testing phase of the classifier, it has been observed that the RBFN which is trained with 500 seconds of data is able to classify the 74 seconds of test data in the correct sequence of the eye movements up to 80% accuracy. The generated decisions are sent to a mobile robot in order to control the robot's direction as it is moving forward with a constant speed.

The system GUI created in Matlab to train the RBFN classifier and control the robot is shown in Figure 21.

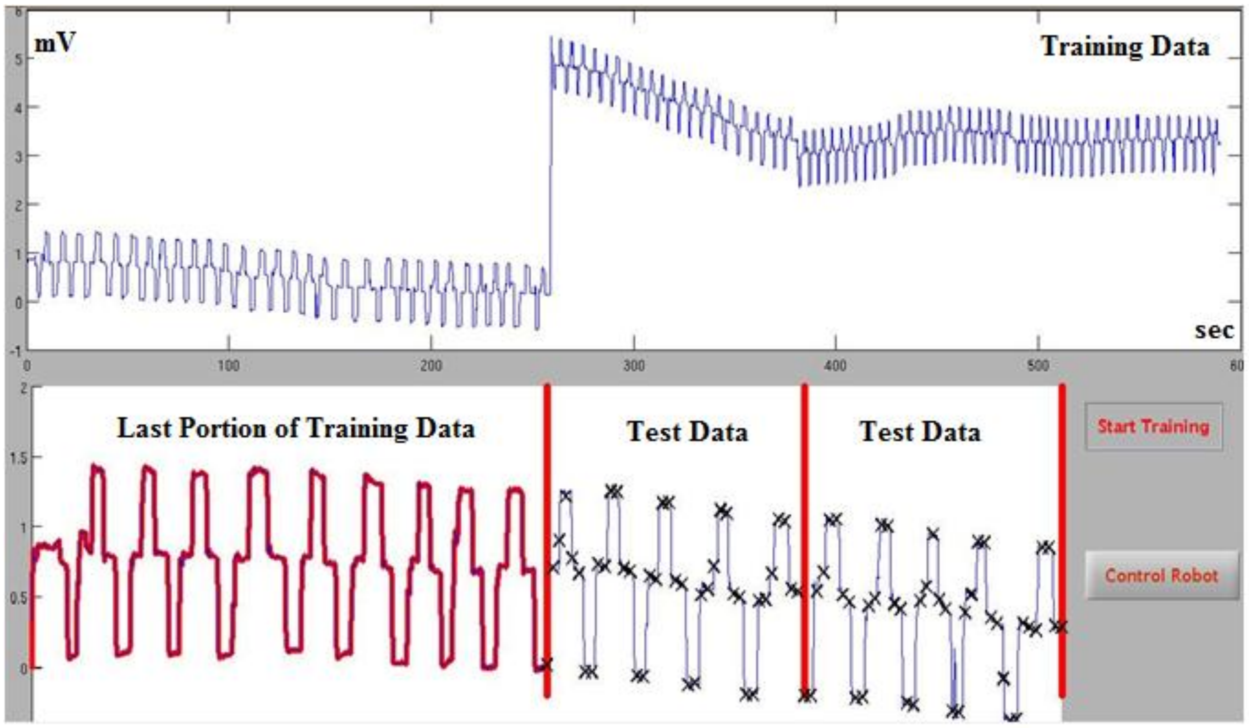


Figure 22 System GUI to Train and Control the Amigobot Robot, the crosses represent the moment that the decision has been made for the EOG signal [61]

4.1.2 The Robot Control

A basic schematic of the mobile robot control system is shown in Figure 23. The software framework is designed as a TCP/IP client-server interaction. A server program is written in Java utilizing *java.net* standard libraries. This program always listens to the connections from any client program on a specified computer host identified by the IP address and a port number of the computer. If any connection occurs from the client software, the message encapsulated in the socket sent from the client program is parsed in order to extract the commands. The message format sent from the client software is as follows:

"<command>,<robotNumber>"

where *command* is a two letter representation of the classification output e.g. *CL* represents Center-Left and means the user looks from the center to the left. Whenever this command is sensed by the server, the robot is given a direction to turn 45 degrees to the left side as it is moving ahead. *robotNumber* is the identification number of the mobile robot used in this study.

The TCP client program is written in Matlab utilizing TCP command functions of Matlab. The *Control Robot* button on the right side of the GUI runs the written server script to connect to the robot server from the Matlab side. It sends the commands of the RBFN Classifier in the format discussed before.

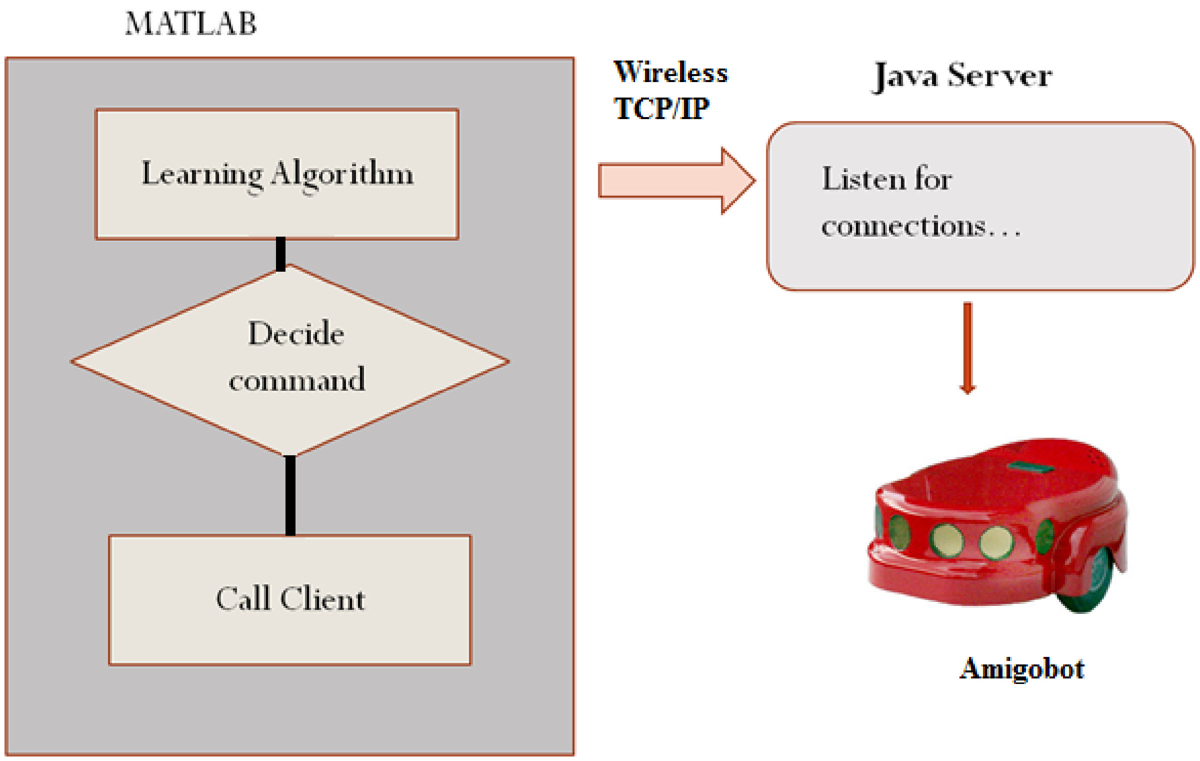


Figure 23 Control Setup of the Mobile Robot

The Amigobot mobile robot used in this study is controlled by *ARIA Java* libraries which is an open source library for controlling the ActivMedia robots [68].

4.2 MediTrack Software

Another application that we have used the RBFN classifier is for a patient tracking system. The objective of this system is to provide the user with an idea if a patient who has been discharged after some period of hospitalization will show up at the scheduled outpatient appointment based on the medical history of the patient. This prediction is done by analyzing several real cases included in a dataset and training the RBFN classifier in order to generate the predictions.

The dataset that contains the experimental data includes 19 variables related to the patient's medical history and demographical information. These data are collected from 1000 patients who are discharged throughout 2008 from all inpatient units at Bellevue Hospital in New York. The variables and explanations are listed in Table 15.

Table 15 Patient Attributes

Variable	**Explanation**
Length of Stay	The duration that the inpatient has stayed in the hospital The options are less than and greater than two months, less than one month and less than two weeks
Sex	Male/Female
Psychiatric diagnoses	Options include: Schizophrenia, Schizoaffective, Major Depressive Disorder, Bipolar and other
Ethnicity	Hispanic, African American, Asian, South East Asian, other
Use of Medsap program	A medication teaching intervention while on the unit Options: True/False
Discharge placement	Which type are they being discharged to, Private Residence, SRO, Shelter, Inpatient Rehab, Inpatient, and Other

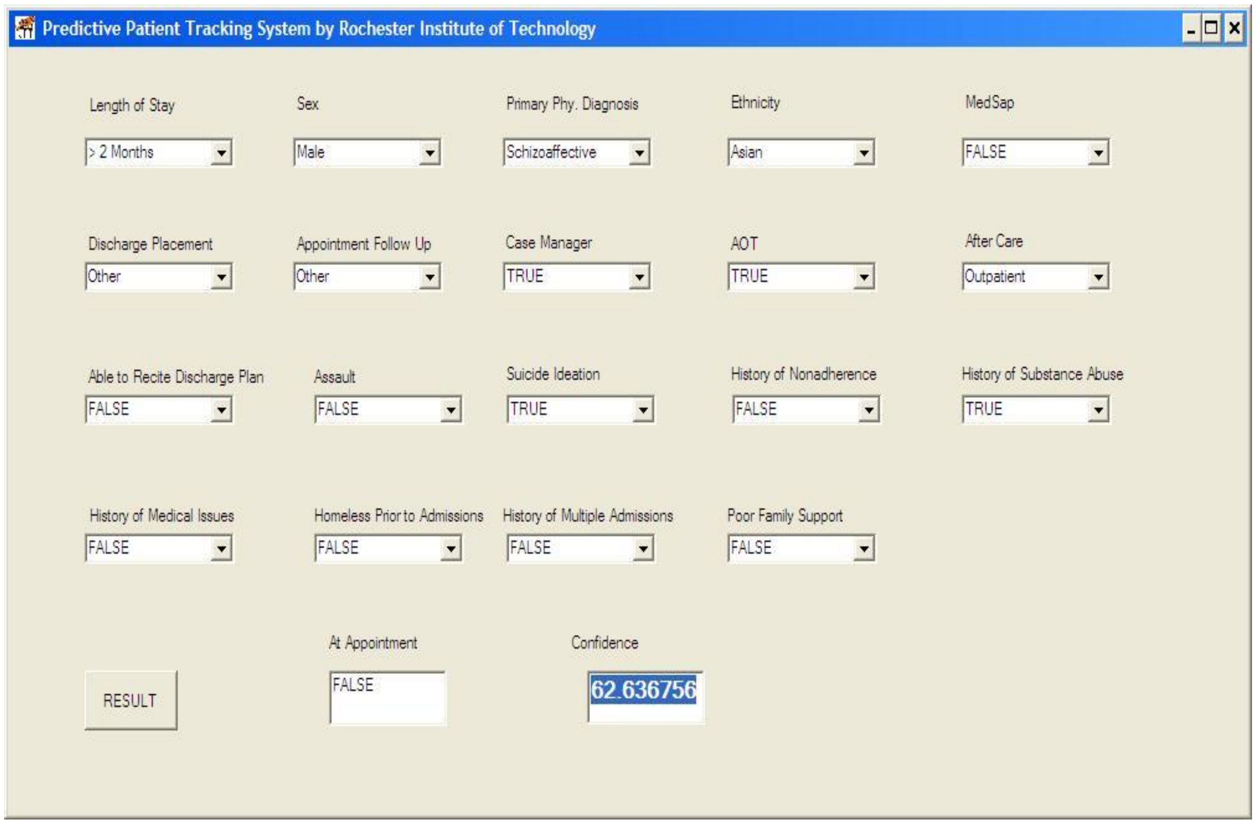
Presence of a case manager	Options: True/False
Presence of AOT	Assisted Outpatient Treatment Options: True/False
History of Assault	True/False
History of Suicidality	True/False
History of Non-adherence	True/False
History of Substance Abuse	True/False
History of Medical issues	True/False
Homeless just prior to admission	True/False
History of multiple admissions	True/False
Presence of Poor family support	True/False
Follow up appointment date less or more than one week	True/False

After Care	Outpatient/Inpatient
Able to Recite Discharge Plan	To be able to recite their discharge plan to a member of the treatment team Options: True/False

After obtaining one of the highest classification performance results with the Medical dataset during data runs using the PSO-RBFN classifier, the resulting center locations and the classifier parameters are used for the hidden layer of RBFN to test the remaining data out of the dataset by the inputs entered from the GUI. The result is a prediction of whether the patient will come to the appointment or not. Two textboxes at the bottom of the GUI indicates if the patient will come to the appointment and the confident level of the decision. This confidence level is produced by an exponential mapping of the output of RBFN around the threshold value 0.5 as in equation (20)

$$confidence = 100.\, e^{(|result-0.5|-0.5)} \tag{20}$$

where *result* represents the output of RBFN output layer. Figure 24 shows the software GUI written in C++ utilizing OpenCV 2.1 Matrix libraries [69].



Predictive Patient Tracking System by Rochester Institute of Technology

Length of Stay	Sex	Primary Phy. Diagnosis	Ethnicity	MedSap
> 2 Months	Male	Schizoaffective	Asian	FALSE
Discharge Placement	Appointment Follow Up	Case Manager	AOT	After Care
Other	Other	TRUE	TRUE	Outpatient
Able to Recite Discharge Plan	Assault	Suicide Ideation	History of Nonadherence	History of Substance Abuse
FALSE	FALSE	TRUE	FALSE	TRUE
History of Medical Issues	Homeless Prior to Admissions	History of Multiple Admissions	Poor Family Support	
FALSE	FALSE	FALSE	FALSE	

At Appointment: FALSE

Confidence: 62.636756

Figure 24 MediTrack Software GUI

As future work, the same type of the software will be designed as an iPhone ® application for widespread usage and new data collection.

4.3 Real-Time Brain Computer Interface Application

In order to provide more objective comparison with the other researcher's work and measure the performance of our proposed algorithm, we have used the standard EEG datasets that most researchers have been described in the literature [55]. After determining that the PSO-RBFN may compete with the performance of the state of the art classifiers, it was decided to apply this technique for real time EEG classification robot control.

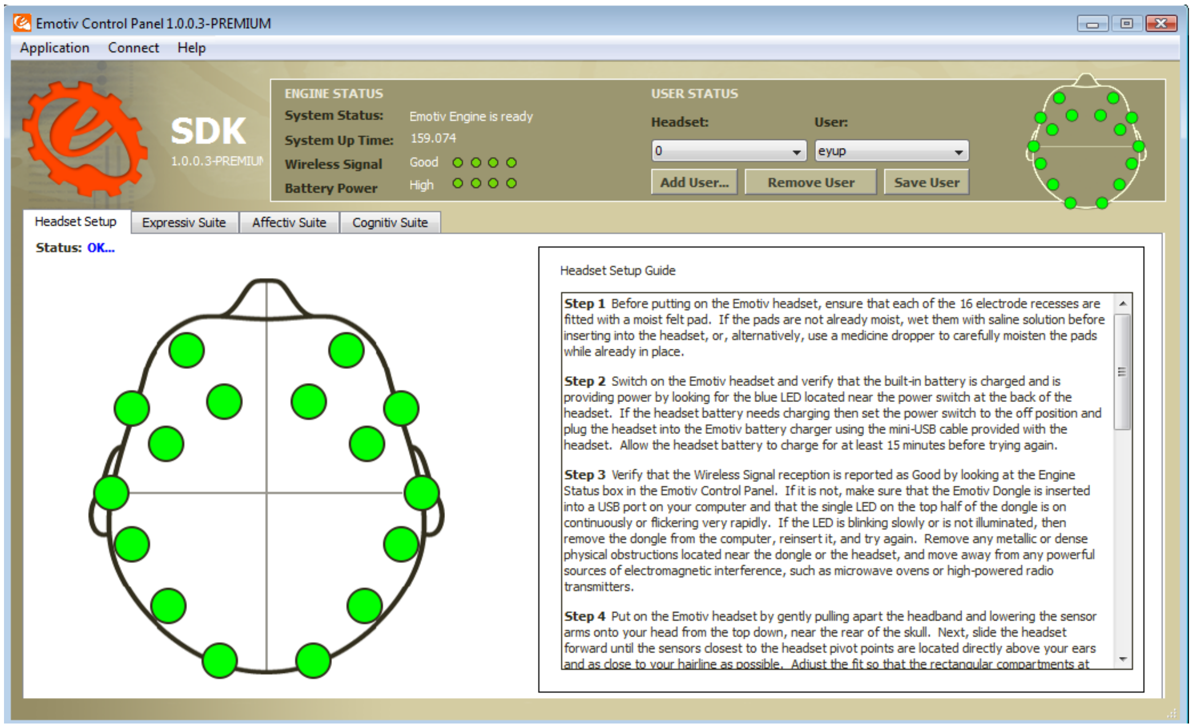
The commercial Emotiv Epoc® EEG data acquisition headsets available in our lab are used for this application. Having the research edition of the headsets in the Multi Agent Biorobotics Lab, we have access to the raw EEG data being sent from the headset to the PC. The headsets have Software Development Kit (SDK) libraries written in C++ and provide the raw EEG data access through the C++ libraries.

Since all of the previous algorithms have been coded in Matlab environment and there are many Matlab built-in functions for Signal Processing, we have tried to find a way of integrating the C++ *dynamic linking libraries* (*dll*) libraries with Matlab. The solution is to load the dll files into the Matlab workspace and call the dll functions from the Matlab m-files. Because of several deficiencies realized in the original header files of the SDK libraries, there were incompatibility problems with loading the dll files into the Matlab's environment. Since the Matlab software can only call C compatible dll libraries and the original header files were written in C++, header files were completely rewritten so that they were C compatible and could be integrated with Matlab.

The following sections provide brief introduction related to the hardware specifications of the headsets, real time data acquisition, SDK basics, training, and real time testing of the designed BCI system.

4.3.1 The Emotiv Epoc Headsets

The Emotiv headsets were originally created in order to provide an innovative way of game control using Brain Computer Interface (BCI) technology [70]. After a basic training session of specific actions using the Emotiv's own EEG recognition engine, the user is able to convert its trained thoughts in the system into control inputs for the computer. This could be a keyboard command such as for a video game input or moving of a cube or an avatar on the screen. The headset comes with three main software packages. The first software package is called the *Control Panel*. It is the main software package that manages the training and testing sessions as well as providing the user information related to the battery level, the status of the electrodes and user profile management. Figure 25 shows the screenshot of the control panel software.



ENGINE STATUS

System Status: Emotiv Engine is ready
System Up Time: 159.074
Wireless Signal: Good ● ● ● ●
Battery Power: High ● ● ● ●

USER STATUS

Headset: 0
User: eyup

Status: OK...

Headset Setup Guide

Step 1 Before putting on the Emotiv headset, ensure that each of the 16 electrode recesses are fitted with a moist felt pad. If the pads are not already moist, wet them with saline solution before inserting into the headset, or, alternatively, use a medicine dropper to carefully moisten the pads while already in place.

Step 2 Switch on the Emotiv headset and verify that the built-in battery is charged and is providing power by looking for the blue LED located near the power switch at the back of the headset. If the headset battery needs charging then set the power switch to the off position and plug the headset into the Emotiv battery charger using the mini-USB cable provided with the headset. Allow the headset battery to charge for at least 15 minutes before trying again.

Step 3 Verify that the Wireless Signal reception is reported as Good by looking at the Engine Status box in the Emotiv Control Panel. If it is not, make sure that the Emotiv Dongle is inserted into a USB port on your computer and that the single LED on the top half of the dongle is on continuously or flickering very rapidly. If the LED is blinking slowly or is not illuminated, then remove the dongle from the computer, reinsert it, and try again. Remove any metallic or dense physical obstructions located near the dongle or the headset, and move away from any powerful sources of electromagnetic interference, such as microwave ovens or high-powered radio transmitters.

Step 4 Put on the Emotiv headset by gently pulling apart the headband and lowering the sensor arms onto your head from the top down, near the rear of the skull. Next, slide the headset forward until the sensors closest to the headset pivot points are located directly above your ears and as close to your hairline as possible. Adjust the fit so that the rectangular compartments at

Figure 25 The Emotiv Headset Control Panel

The *TestBench* software as shown in Figure 26 provides access to the raw EEG data and real time EEG plots for the users who have the research edition license.

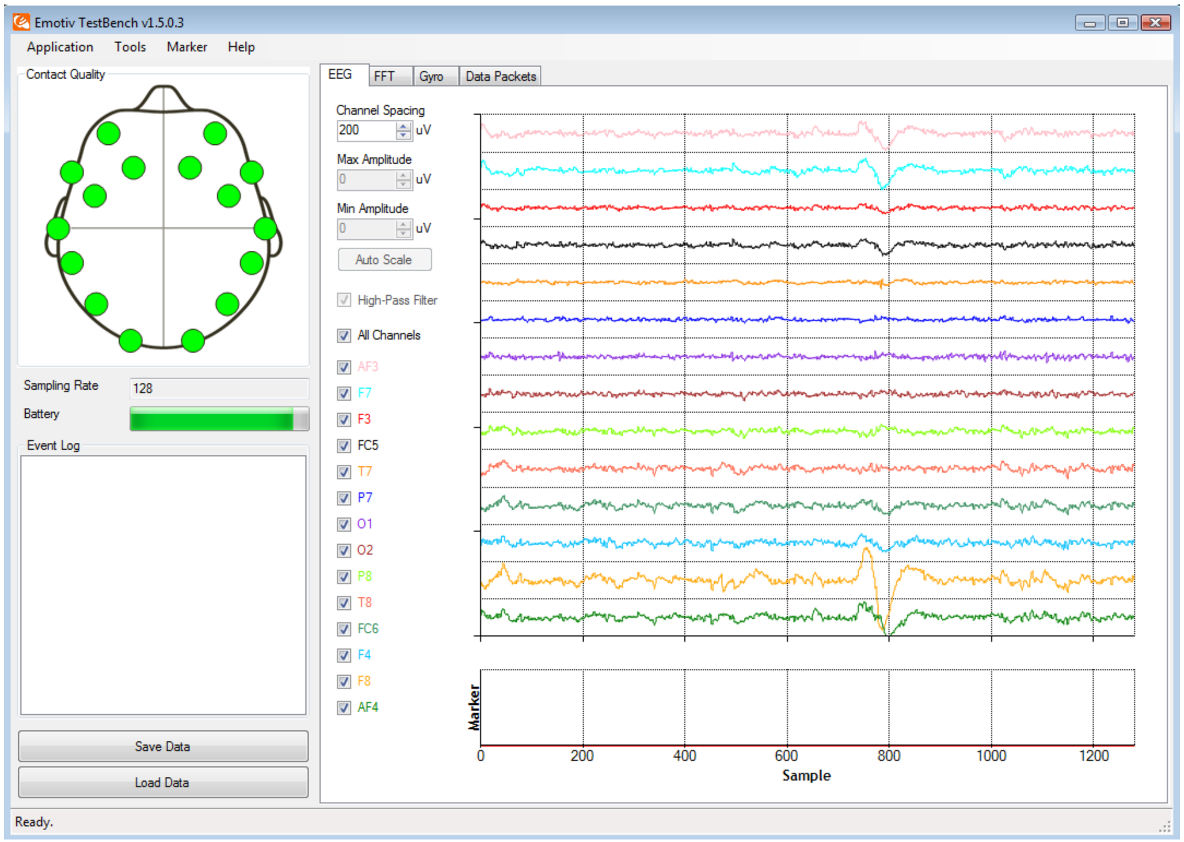


Figure 26 The TestBench Software Screen Capture

In addition to these two main software modules, the *EmoComposer* is used to simulate the headset signals for application developers to test their software without having the real headset. The EEG headset is designed such as the Electrode placements match the International 10-20 system as it was introduced in the first section. Figure 27 shows the sensor layouts for the headset and their labels for the corresponding International 10-20 system.

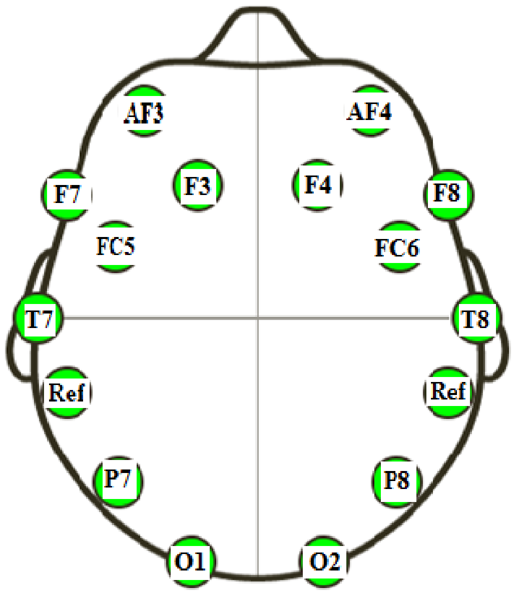


Figure 27 Electrode Positions of the Emotiv Headset

In this study, it has been decided to use the electrodes FC5, F3, F4, FC6, O1, and O2. As they relate to the Sensorimotor Cortex (FC5, F3, F4, FC6) and Visual Cortex areas (O1, O2) of the brain. This section of the brain is known to manage planning, control, and execution of motor actions of the body such as lifting right hand or left hand [10], [11]. As it can be seen from Figure 26, the positions of the electrodes are symmetric according to the central axis of the brain. As it is stated by [55], [56], [57] the motor actions controlled by the brain such as left thinking or right thinking will also cause the same power characteristics in the brain every time the subject thinks about them. This might be distinguished by analyzing the bandpower features of the singals and training of a classifier in order to distinguish these patterns in the brain.

The next section provides information related to the headset hardware specifications. It gives brief introduction to the built-in filtering processes performed before the raw EEG data is transmitted into the computer.

4.3.2. Hardware Specifications of the Headset

The raw EEG data is collected through a sensor technology called wet sensors. There are soft pads on top of the electrodes placed on the headset and these pads are wetted by saline solution in order the increase the conductivity between the scalp and the headset.

The data inside the headset is collected through a C-R high pass hardware filter with 0.16 Hz cut-off frequency. After this process, the signal is preamplified and low pass filtered at 83 Hz cut-off frequency. The low-passed filtered data is passed through a fifth order Sinc filter to notch out the frequencies between 50-60 Hz which correspond the frequencies such as the mains frequencies. The sampling rate of the headset is 128 Hz per each channel. The next section discusses about the SDK software libraries used to collect raw EEG data from the headset. It gives basic definitions of the key functions used in designing the software.

4.3.3 Software Development Kit (SDK) Libraries

In order to access the raw EEG data, the SDK *dll* libraries which are written in C++ has to be used. These *dll* files manage the connection between the headset hardware and PC. There are a few important library functions that need special consideration and manage the data collection. These are *EE_DataSetBufferSizeInSec(), EE_DataGetNumberOfSample(), EE_DataGet(), EE_DataAcquisitionEnable(),and EE_DataUpdateHandle().*

EE_DataSetBufferSizeInSec() sets the size of the internal data buffer that the raw data will be stored. The size is set in seconds and this determines the maximum size of the buffer. Within this specific amount of time, data is written into the buffer and after the determined buffer time is over, the buffer is cleared and the new data is overwritten. One should be careful about setting the size of the buffer so that the data will not be lost.

EE_DataUpdateHandle() is one of the main functions that should be called before *EE_DataGet().* It updates the content of the data handle created by *EE_DataCreate(),* to the point of the new data since the last call. Note that the amount of the data fetched by this call is the same as the number of samples returned by *EE_DataGetNumberOfSample().*

EE_DataGet() is the main function that fetches the data from the data buffer into a C array. Once this function is called the current data in the buffer is cleared and the buffer is started to be filled with new incoming data.

The most critical issue is setting the buffer size large enough so that there is enough time between EE_DataGet() calls and thus function calls can fetch the data before it has cleared.

In addition, *EE_DataAcquisitionEnable()* is the function that sets the data collection flag to true so that the data buffer could be readable.

4.3.5 Real Time Training, Testing, and the GUI Design

In order to perform the training and testing for the real time system, a graphical user interface has been created utilizing the Matlab GUI design environment. Figure 28 shows the system GUI designed for the real time implementation.

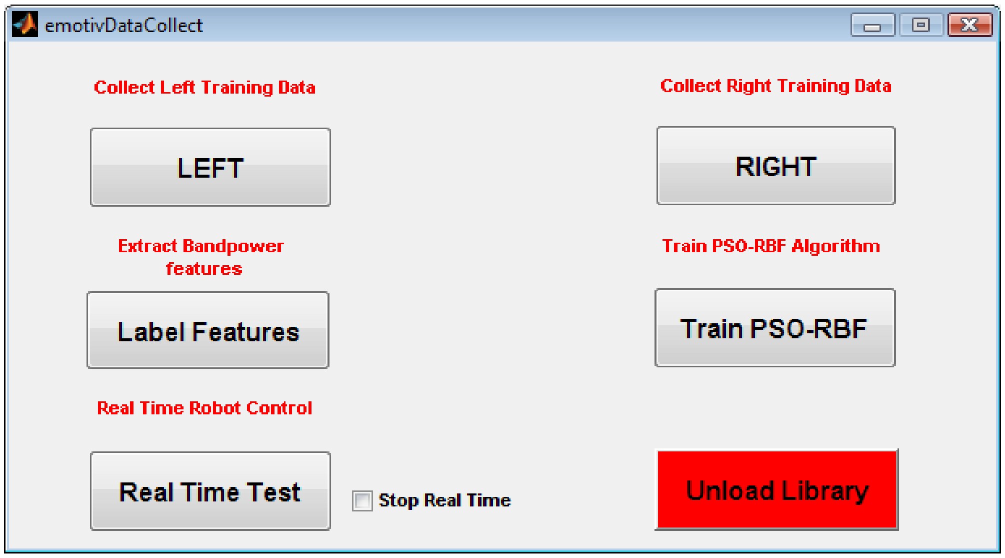


Figure 28 Software GUI Designed for Real Time System Training and Robot Control

In Figure 27, *Left* and *Right* training buttons are used to supervise the subject by providing visual stimuli which will be explained later in detail. When this button is pressed, the mean bandpower features within *alpha* and *beta* bands are extracted from the single trial and saved into a global feature buffer which is implemented as a Matlab Matrix variable. The raw EEG data is also stored into another global data buffer in order to be used later analysis of the data.

Label Features button is used to label the collected trials with either one or zero label for the left thinking and right thinking, respectively.

Train PSO-RBF button starts running the PSO algorithm for 100 iterations and find the cluster centers for the RBFN. The number of hidden layers are taken the same as the number of features obtained from the signal. After that the variance and weight combinations that give the highest cross validation accuracy is saved into a file to be later used in the real time.

Real Time Test button is used to collect the data continuously in real time and generate the band power features from the six seconds of the trials the same as it is used to train the system for each trial. The results are printed according to the threshold value 0.5 coming from the RBFN multiplication.

Unload Library button clears the allocated memory and pointers for the next usage.

In order to provide a visual stimulus to the subject for the left or right thinking, the EmoCube script files are used. The EmoCube works as a standalone executable server application that accepts UDP packets in order to move the cube to several locations in the screen within the main frame. The neutral position of the EmoCube is shown in Figure 29. When the left training button is pressed, a UDP message is sent from Matlab to the EmoCube server and while the cube is moving, the data collected from the headset is processed and recorded into global buffer in *.mat* file format for the future use. The UDP message format sent from Matlab to the EmoCube server is:

"<Hex Code of the direction to control the cube >, <The power of the movement between one and 100>"

For training of the system, 271 trials that each represents the mean band power of six seconds visual trials within the *alpha* and *beta* frequencies are collected. After that, the PSO-RBFN algorithm is trained. The cross validation accuracy obtained with the training has reached to 62 percent.

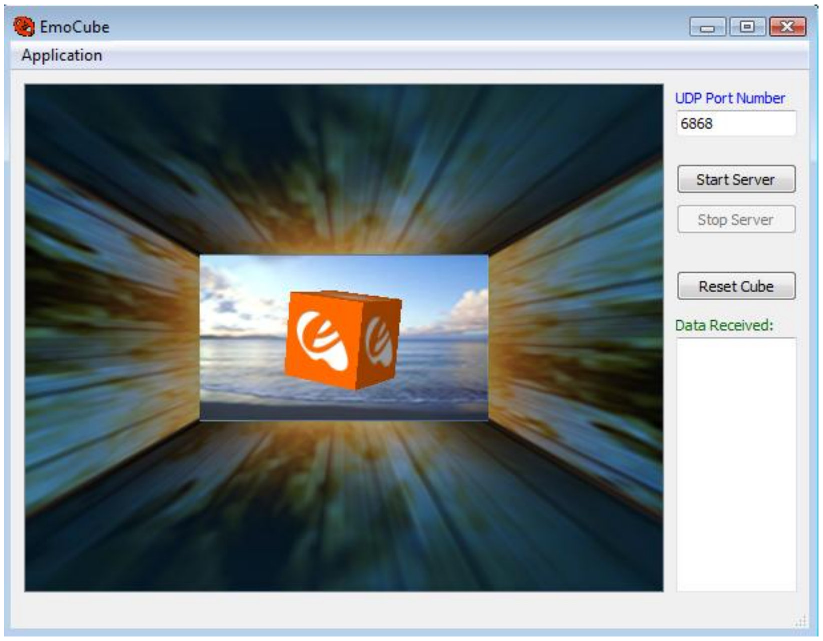
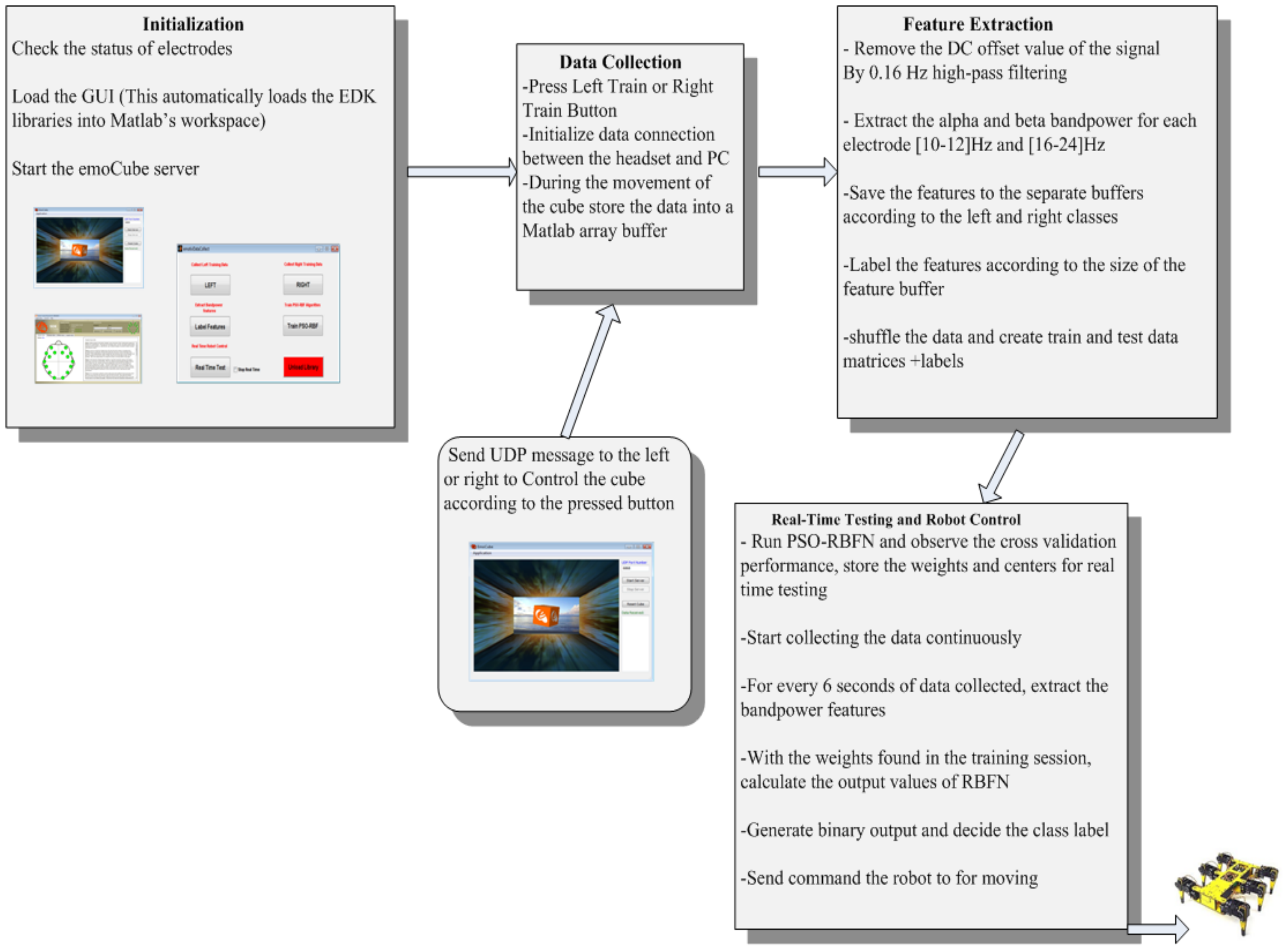


Figure 29 Visual Stimuli by using the EmoCube

After the training of the PSO-RBFN classifier, the real time testing has been performed. The block diagram that includes describes the whole system is shown in Figure 30. The next section gives information about how the generated classifier output commands is converted into a mobile robot control inputs in order to move a mobile robot called Hexapod.



Initialization

Check the status of electrodes

Load the GUI (This automatically loads the EDK libraries into Matlab's workspace)

Start the emoCube server

Data Collection

-Press Left Train or Right Train Button
-Initialize data connection between the headset and PC
-During the movement of the cube store the data into a Matlab array buffer

Send UDP message to the left or right to Control the cube according to the pressed button

Feature Extraction

- Remove the DC offset value of the signal By 0.16 Hz high-pass filtering

- Extract the alpha and beta bandpower for each electrode [10-12]Hz and [16-24]Hz

-Save the features to the separate buffers according to the left and right classes

-Label the features according to the size of the feature buffer

-shuffle the data and create train and test data matrices +labels

Real-Time Testing and Robot Control

- Run PSO-RBFN and observe the cross validation performance, store the weights and centers for real time testing

-Start collecting the data continuously

-For every 6 seconds of data collected, extract the bandpower features

-With the weights found in the training session, calculate the output values of RBFN

-Generate binary output and decide the class label

-Send command the robot to for moving

Figure 30 The Whole System Block Diagram

4.3.6. Robot Control with the Designed BCI

The outputs of the RBFN during real-time EEG processing are converted into the serial port inputs for a walking robot called Hexapod. The joints of the robot are powered by servo motors which are controlled by SSC-32 servo controller, Mini-ABB controller board and basic ATOM microcontroller.

The microcontroller used on the Hexapod is BasicAtom28 and it contains internal memory with 384 Bytes of RAM and 8K of Flash. Utilizing the Basic Atom IDE, a control program is written and loaded on the microcontroller. This program continuously listens to serial port commands from the PC and parses the commands sent from the PC's serial port. The parsed commands are converted into the servo moves in order to turn the robot to the right or to the left. The ASCII representative constants that are sent from the PC's serial port to control the robot are listed in Table 16.

Table 16 List of ASCII Characters sent from PC to Control the Hexapod

Constant	**Control Direction**
119	'w' Forward
115	's' Backward
97	'a' Left
100	'd' Right

The commands are sent from Matlab utilizing Matlab’s serial port commands. The m-files are separated into single command files that might be called individually for each action.

5. Conclusions

This thesis has made several contributions to the literature in the field of Machine learning, Biorobotics, and Cybernetics. During the scope of this study, FFSVM, IFFSVM are applied for the first time to the biological datasets in the literature and their performances are compared with RBFN classifiers. In addition, clustering abilities of the PSO, *K*-means and Fuzzy *C*-means algorithms are studied and their weaknesses and strengths are analyzed by applying each technique on several datasets.

A novel approach called PFFSVC is proposed to improve the membership values for the classification of the data members which could be used as additional features during the classification. The effects of clustering algorithms on the classifiers accuracies are investigated. Thus, this study also contributes uniquely to the literature in a way by analyzing the clustering algorithm effects on the classification ability of RBF Networks in their parameter selection.

According to the results obtained from several standard datasets, It has been found that the RBFN classifier together with PSO and FCM clustering is able to reach the performance of state of the art classifiers. In addition, it has been observed that the PSO clustering helps the RBFN classifier better catch the highest classification performance. Thus, it has been successfully applied to the real world applications such as a medical patient tracking system, and a brain computer interface application. Consequently, the following are the major contributions of this thesis:

- FFSVC and IFFSVC classifiers are applied on classification of the biological datasets for the first time.
- PSO, FCM and *K*-means clustering algorithms are compared according to their clustering abilities.
- The performance of RBFN classifier that the hidden layer parameters are found by PSO and FCM clustering is compared with the performance of FFSVC and IFFSVC techniques.
- The PSO-RBFN classifier is first used in real-time application in order to classify human biosignals and control a mobile robot.

As future work of this study, the real time data acquisition and classification can be extended to control the wheelchair robot available in the Multi Agent BioRobotics Lab that an assistive device for the disabled people could be created.

References

[1] B. Greene, S. Meredith, R. B. Reilly, and G. Donohoe, "A novel, portable eye tracking system for use in schizophrenia research," In *Proc. ISSC 2004*, 2004, pp. 1-5 .

[2] K. Suetsugu, Y. Tagawa, T. Inada, and N. Shiba, "FES Position Control of Forearm Using EOG," *Advances in Neuro-Information Processing*, pp. 494-503, 2009.

[3] R. Barea, L. Boqute, and M. Mazo, "Wheelchair guidance strategies using EOG," *Journal of Intelligent and Robotic Systems*, vol. 34, pp. 279-299, 2002.

[4] F. Lotte, M. Congedo, A. Lecuyer, F. Lamarche, and B. Arnaldi, "A review of classification algorithms for eeg-based brain-computer interfaces," *Journal of Neural Engineering*, vol. 4, pp. 1-13, 2007

[5] S. Yuge, Y. Ying, and X. Xinhe, "EEG Analysis of Alcoholics and Controls Based on Feature Extraction," In *Proc of ICSP2006*, 2006, pp. 6-11.

[6] Wiener, *Cybernetics or Control and Communication in the Animal and the Machine*, MIT Press, 1965.

[7] B. Webb and T. R. Consi, *Biorobotics*, Cambridge, MA: MIT Press, 2001.

[8] R. Barea, L. Boquete, M. Mazo, and E. Lopez, "Wheelchair guidance strategies using EOG,"*Journal of intelligent and robotic systems,* vol. 34, no. 3, pp. 279-299, 2002.

[9] Win van Drongelen, *Signal Processing for Neuroscientist*, Burlington, MA: Elseveir, 2007.

[10] S. Sanei, *EEG Signal Processing*, John Wiley, 2007.

[11] G. Pfurtscheller and C. Neuper, "Motor imagery and direct brain-computer communication," In *Proceedings of the IEEE*, vol. 89, pp. 1123-1134, 2001.

[12] G. Dornhege, *Toward brain-computer interfacing*, Cambridge, Mass: MIT Press, 2007.

[13] A. Vallabhaneni, T. Wang, and B. He, "Brain Computer Interface," *Neural Engineering*, Ed. B. He, Springer US, pp. 85-121, 2005.

[14] T. Berger, J, Chapin, G. Gerhardt, and D. McFarland, *Brain-Computer Interfaces: An international assessment of research and development trends*, Springer, 2008.

[15] S. Vorobyov and A. Cichocki, "Blind noise reduction for multisensory signals using ICA and subspace filtering, with application to EEG analysis," *Biological Cybernetics*, vol. 86, no. 4, pp. 293-303, April, 2002.

[16] T. P. Jung, C. Humphries, and T.W. Lee, "Removing Electroencephalographic Artifacts: Comparison between ICA and PCA," *Neural Networks and Signal Processing*, vol. 8, no. 4, pp. 63-72, 1998.

[17] K. Nakayama, Y. Kaneda, and A. Hirano, "A Brain Computer Interface Based on FFT and Multilayer Feature Extraction and Generalization Neural Network," In *Proceedings of 2007 International Symposium on Intelligent Signal Processing and Communication Systems*, 2007, pp. 101-104.

[18] T. Felzer and B. Freisleben, "Analyzing EEG Signals Using the Probability Estimating Guarded Neural Classifier," *IEEE Transactions on Neura Systems and Rehabilitation Engineering*, vol. 11, no. 4, pp. 361-371, December 2003.

[19] H. Lee and S. Choi, "PCA+HMM+SVM for EEG pattern classification," In Proc. *7th Int. Symp. on Signal Processing and its Applications*, 2003.

[20] R. Palaniappan, "Brain computer interface design using band powers extracted during mental tasks," In *Proc. 2nd Int. IEEE EMBS Conf. on Neural Engineering*, 2005.

[21] G. Pfurtscheller, D. Flotzinger, and J. Kalcher, "Brain–computer interface-a new communication device for handicapped persons," *Journal of Microcomputing Applications*, vol. 16, pp. 293-299, 1993.

[22] R. Boostani and M. H. Moradi, "A new approach in the BCI research based on fractal dimension as feature and Adaboost as classifier," *Journal of Neural Engineering*, vol. 1, pp. 212-217, 2004.

[23] B. Obermaier, C.Neuper, C. Guger, and G. Pfurtscheller, "Information transfer rate in a five-classes brain–computer interface," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 9, no. 3, pp. 283 - 288 , 2001.

[24] E. Estrada, H. Nazeran, J. Barragan , J. R. Burk, E. A. Lucas, and K. Behbehani, "EOG and EMG: Two Important Switches in Automatic Sleep Stage Classification," In *Proceedings of the 28th IEEE EMBS Annual International Conference*, 2006.

[25] E. Alpaydin, *Introduction to Machine Learning*, 1st ed., Cambridge, Masschusetts: MIT Press, 2004.

[26] A. Ganapathiraju, J. Hamaker, and J. Picone, "Applications of support vector machines to speech recognition," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2348-2355, 2004.

[27] O. Lemon and O. Pietquin, "Machine learning for spoken dialogue systems," In *Proceedings of the European Conference on Speech Communication and Technologies*, 2007.

[28] B. Kovesi, J. M. Voucher, and S. Saoudi, "Stochastic K-means algorithm for vector quantization," *Pattern Recognition Letters*, vol. 22, no. 6, pp. 603-610, 2001.

[29] A. Ahmadyfard and H. Modares, "Combining PSO and *K*-means to Enhance Data Clustering," In *Proc. International Symposium on Telecommunications*, 2008.

[30] K. Kim and H. Ahn, "A recommender system using GA K-means clustering in an online shopping market," *Expert systems with applications*, vol. 34, no. 2, pp. 1200-1209, 2008.

[31] J. C. Dunn, “Some recent investigations of a new fuzzy partitioning algorithm and its application to pattern classification problems”, *Journal of Cybernetics*, vol. 4, pp 1-15, 1974.

[32] N. R. Pal, K. Pal, and J. C. Bezdek, "A mixed c-means clustering model," In *Proceedings of the Sixth IEEE International Conference on Fuzzy Systems*, 1997.

[33] E. Cox, *Fuzzy Modelling and Genetic Algorithms for Data Mining and Exploration*, San Francisco, CA: Elseveir, 2005.

[34] B. Thomas, G. Raju, and S, Wangmo, "A Modified Fuzzy C-Means Algorithm for Natural Data Exploration," In *Proc of World Academiy of Science and Engineering*, 2009, pp. 478-481.

[35] X. Wang, Y. Wang, and L. Wang, " Improving fuzzy c-means clustering based on feature-weight learning," *Pattern Recognition Letters*, vol. 24, pp. 1125-1132, 2004.

[36] T. Stubbings and H. Hutter, "Classification of analytical images with radial basis function networks and forward selection," *Chemometrics and Intelligent Laboratory Systems*, vol. 49, no. 163-172, pp. 89-98, 1999.

[37] J. M. Yih, Y. H. Lin, and H. C. Liu, "Clustering Analysis Method based on Fuzzy C-Means Algorithm of PSO and PPSO with Application in Real Data," *International Journal of Geology*, vol. 1, no. 4, pp. 89-98, 2007.

[38] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," In *Proc. IEEE Int'l. Conf. on Neural Networks*, IV, pp. 1942–1948, 1995.

[39] F. Sahin and A. Devasia, “Distributed Particle Swarm Optimization for Structural Bayesian Network Learning”, *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*, Advanced Robotics Systems International, December 2007.

[40] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," In *Proceedings of the 2001 congress on evolutionary computation*, 2001, pp. 81-86.

[41] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," In *proc. of Evolutionary Computation Congress*, 2000, pp. 84-88.

[42] R. C. Eberhart, and Y. Shi, *Computational Intelligence: Concepts to Implementations*, USA: Morgan Kaufmann, 2007.

[43] D. W. Van der Merwe, A. P. Engelbrech, "Data clustering using particle swarm optimization," In *Proceedings of IEEE Congress on Evolutionary Computation*, 2003, pp. 215-220.

[44] S. Haykin, *Neural Networks: a comprehensive foundation*, 3 ed. , Upper Saddle River, NJ: Prentice Hall, 2008.

[45] T. M. Cover, "Geometrical and Statistical properties of systems of linear inequalities with applications in pattern recognition". *IEEE Transactions on Electronic Computers* , 1965, pp. 326–334.

[46] A. S. Loong, O. H. Choon, and L. H. Chin, "Criterion in Selecting the Clustering Algorithm in Radial Basis Functional Link Nets," *WSEAS Transactions on Systems*, vol. 7, no. 11, pp. 1290-1299, 2008.

[47] J. V. Marcos, R. Hornero, and D. Alvarez, "Radial basis function classifiers to help in the diagnosis of the obstructive sleep apnoea syndrome from nocturnal oximetry," *Medical and Biological Engineering and Computing*, vol. 46, no. 4, pp. 323-332, 2008.

[48] L. Hongyang and Jia He, "The Application of Dynamic K-means Clustering Algorithm in the Center Selection of RBF Neural Networks," In *Proc 3rd International Conference on Genetic and Evolutionary Computing*, vol. 177, no. 23, pp. 488 - 491, 2009.

[49] A. Celikyilmaz and I. B. Turksen, "A new classifier design with fuzzy functions," In *Proc. of the 11th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, 2009, pp. 1123-1134.

[50] A. Celikyilmaz, I. B. Turksen, M. Aktas, and M. Doganay, "Increasing accuracy of two-class pattern recognition with enhanced fuzzy functions," *Expert Systems with Applications*, vol. 36, no. 2P1, pp. 1337-1354, 2009.

[51] J. Platt, *Probabilistic Outputs for Support Vector Machines and comparison to regularized likelihood methods*, MIT Press, Cambridge, MA, 2000.

[52] A. Celikyilmaz and B. Turksen, "Fuzzy Functions with Support Vector Machines," *Information Sciences*, vol. 177, no. 23, pp. 5163-5177, 2007.

[53] "UCI Machine Learning Repository" [online] Avaiable: http://archive.ics.uci.edu/ml [Accessed: June 2010]

[54] B. Blankertz, G. Dornhedge, M. Krauledat, K. Robert Muller, and G. Curio, "The Non-Invasive Berlin Brain-Computer Interface: Fast Acquisition of Effective Performance in Untrained Subjects," *Neuroimage*, vol. 37, no. 2, pp. 539-550, 2007.

[55] G. Pfurtscheller and C. Neuper, "Motor imagery and direct brain-computer communication, " *In Proc. of the IEEE*, vol. 89, pp. 1123-1134, 2001.

[56] M. Zhong, F. Lotte, M. Girolami, and A. Lecuyer, "Classifying EEG for brain computer interfaces using gaussian processes," *Pattern Recognition Letters*, vol. 29, no. 3, pp. 354-359, 2008.

[57] F. Lotte, "The use of fuzzy inference systems for classification in eegbased brain-computer interfaces," In *Proc. of the 3rd international Brain-Computer Interface workshop*, 2006, pp. 12-13.

[58] D. Wettschereck and T. Diettrerich, "Improving the Performance of Radial Basis Function Networks by Learning Center Locations," *Advances in Neural Information Processing Systems*, vol. 4., 1992.

[59] T. Stubbings and H. Hutter, "Classification of analytical images with radial basis function networks and forward selection," *Chemometrics and Intelligent Laboratory Systems*, vol. 49, no. 2, pp. 163-172, 1999.

[60] "LIBSVM: a library for support vector machines" [online] Avaiable: http://www.csie.ntu.edu.tw/~cjlin/libsvm [Accessed: June 2010]

[61] E. Cinar and F. Sahin, "EOG Controlled Mobile Robot Using Radial Basis Function Networks", In *Proc. IEEE Fifth International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control (ICSCCW 2009),* September 2009

[62] K. Takahashi, T. Nakauke, and M. Hashimoto, "Remarks on Hands-Free Manipulation Using Bio-Potential Signals," In *Proc. of IEEE International Conference on Systems Man and Cybernetics*, 2005, pp. 2965-2970.

[63] S. L. Tsui, P. Jia, J. Q. Gan, and K. Yuan, "EMG-based Hands-Free Wheelchair Control with EOG Attention Shift Detection," In *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, 2007, pp. 1266-1273.

[64] R. Barea, L. Boquete, M. Mazo, and E. López, "Wheelchair Guidance Strategies Using EOG," *Journal of Intelligent and Robotic Systems*, vol. 34, pp. 279-299, 2002.

[65] J. J. Tecce, J. Gibs, and P. Olivieri, "Eye movement control of computer functions," *International Journal of Psychophysiology*, vol. 29, no. 1, pp. 319-325, 1998.

[66] W. Wijesome, K. S. Wee, and O. C. Wee, "EOG based control of mobile assistive platforms for the severely disabled," In *IEEE International Conference of Robotics and Biomimetics*, 2005, pp. 490-494.

[67] "Cleveland Medical Devices Inc." [online] Avaiable: http://www.clevemed.com. [Accessed: June 2010].

[68] "ARIA robot control libraries" [online] Avaiable: www.activmedia.com [Accessed: June 2010].

[69] "Open Computer Vision Libraries" [online] Avaiable : http://opencv.willowgarage.com/wiki/Welcome. [Accessed: June 2010].

[70] "Emotiv: Brain Computer Interface Technology" [online] Avaiable: http:// www.emotiv.com [Accessed:June 2010].