CHAPTER 4

FEATURE EXTRACTION AND CLASSIFICATION

4.1. INTRODUCTION

This chapter briefly describes about the techniques of feature extraction in which power spectral density is used. The three methods of PSD are a parametric, non parametric and high resolution method. The neural network models are used for the classification purpose. Here two neural network models, namely Feed Forward Neural Network (FFNN) and Recurrent Neural Network (RNN) are used. These methods are discussed briefly.

4.2 FEATURE EXTRACTION

The input data to an algorithm is too big to be processed and if it is suspected to be notoriously redundant then the input data will be transformed into a reduced representation set of features. Transforming the input data into the set of features is called feature extraction. If the features extracted are carefully chosen it is expected that the features set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input. Many different features have been thought to be extracted from EEG signals. PSD of the segmented signals is estimated and used as features. PSD describes how the energy of a signal or a time series is distributed with frequency. The six PSD algorithms, namely Covariance, Modified Covariance, MUSIC, Burg, Welch and Yule-Walker are used.

4.3 POWER SPECTRUM DENSITY (PSD)

One of the most important applications in Digital Signal Processing (DSP) is the power spectrum of signals which are periodic or random. A spectrum is presentation of magnitude against frequency of one parameter. The PSD explains how the power or energy of a signal is distributed across frequency. Power spectrum is commonly defined as the Fourier
Transform of the autocorrelation function. In continuous and discrete notation, the power
equation becomes:

$$PS(f) = \int_{\tau}^{T} r_{xx}(\tau) e^{-2\pi f T \tau} d\tau$$

(4.1)

$$PS(f) = \int_{\tau}^{N-1} r_{xx}(\tau) e^{-2\pi f T \tau} d\tau$$

(4.2)

Where $r_{xx}(n)$ is autocorrelation function. Since auto correlation functions has odd symmetry,
the sine terms and Eq. 4.2 can be simplified to include only real cosine terms.

$$PS(f) = \int_{0}^{T} r_{xx}(\tau) \cos(2\pi m f r T \tau) d\tau$$

(4.3)

$$PS(f) = \int_{n=0}^{N-1} r_{xx}(\tau) \cos(2\pi m f r T \tau)$$

(4.4)

These equations are continuous and discrete form is sometimes referred to as the cosine
transforms. This approach to evaluating the power spectrum has lost favor to the so-called
direct approach, given by equation 4.5 below primarily because of the efficiency of Fast
Fourier Transform (Kay et al., 1988).

The direct approach is motivated by the fact that the energy contained as an analog signal,
$x_t$ is related to magnitude of the signal squared, integrated over time

$$E = \int_{-\infty}^{\infty} |x_t|^2 dt$$

(4.5)

By an extension of Parseval theorem it is easy to show that

$$\int_{-\infty}^{\infty} |x_t|^2 dt = \int_{-\infty}^{\infty} |X_f|^2 df$$

(4.6)
Hence, equals the energy density function over frequency also referred to as the energy spectral density, the PSD or simply power spectrum. In the direct approach, the power spectrum is calculated as the magnitude square of the Fourier Transform of the waveform of interest:

\[ PS(f) = |Xf|^2 \]  \hspace{1cm} (4.7)

The PSD is computed as the distribution of power per unit frequency. Power spectral density is normally used for signal feature extraction. Hence PSD is frequency domain analysis. The performance of various PSD estimation techniques are parametric method, non-parametric methods and high resolution method are compared to different epoch lengths in case of real signal (Akankshya Shradhanjali et al., 2013).

1. Parametric methods are based on the estimation of a linear time invariant system from noise by autoregressive-moving-average (ARMA) model such as Covariance, Burg, Modified Covariance and Yule–Walker. These methods have improved performances, although they are affected by the SNR level.

2. Nonparametric methods include conventional Fourier analysis, optimal band pass filtering analysis, Periodogram, and Welch. These methods do not solve the limits of the frequency resolution of the classical Fourier analysis.

3. High-resolution methods include techniques such as MUSIC and Eigenvector. These methods can detect frequencies with low SNR and compute the autocorrelation matrix, and its Eigen-values can be separated into signal and noise spaces. These methods define a Pseudo-spectrum function with large peaks that are subspace frequency estimates, and they are commonly used in the communication area. They have been recently introduced into the area of induction machine diagnosis by the application of the MUSIC method.
4.3.1 Parametric Spectral Analysis Method

Autoregressive (AR) method is a commonly used method. The most important advantage of AR model parameter provides stable statistical estimates with high frequency resolution. For this reason, in the analysis we have used Burg, Covariance, Modified Covariance and Yule-Walker methods. Estimation accuracy of parameters in AR signal models is a suggested matter and estimations are found by solving linear equation. The magnitude of the signal in the period given in the AR method is acquired by the total no of the magnitudes of previous samples and in addition, it minimizes the errors. Model sequence is connected to AR factors. Selection of the degree of AR model is very important and is defined by different criteria. Though a number of methods are used to determine the degree of the model.

\[ AIC(p) = \ln \sigma^2 + \frac{2p}{N} \] (4.8)

Here \( \sigma^2 \) is the estimated variance of linear estimating error and \( N \) is the length of the data. With the increase in the AR model parameter, \( \sigma^2 \) diminishes and therefore, \( \ln \sigma^2 \) diminishes as well. In this case, \( p \) must be valid as minimum (Djuric et. al. 1999; Keith et al., 1993; Dhaparidze et al., 1983).

4.3.1.1 Covariance Method

The Covariance method for AR spectral estimation is based on minimizing the forward prediction error. This method fits an autoregressive model to the signal by minimizing the forward prediction error in the least squares sense. In the covariance method all the data points are needed to compute the prediction error power estimate. No zeroing of the data is necessary. The AR parameter estimates the solution of the equations and can be written (Lawrence et al., 1987; Kay et al., 1988).

\[
\begin{bmatrix}
  c(1,0) \\
  \vdots \\
  c(p,0)
\end{bmatrix} + \begin{bmatrix}
  \vdots \\
  c(1,1) \ldots c(1,p)
\end{bmatrix} \begin{bmatrix}
  \hat{a}(1) \\
  \vdots \\
  c(p,1) \ldots c(p,p)
\end{bmatrix} = \begin{bmatrix}
  0 \\
  \vdots \\
  0
\end{bmatrix}
\] (4.11)
Where
\[ c(j, k) = \frac{1}{N-P} \sum_{n=p}^{N-1} x^* (n - jx(n-k)) \] (4.12)

From Eq. (4.11) the AR parameter estimates are found as:
\[ a = -c_{p-1}^{p} c_{p} \] (4.13)

This statement is acquired and white noise variance is
\[ \sigma^2 = c[0,0] + \sum_{k=1}^{p} a[k] c(0,k) \] (4.14)

Based on this statement, power spectral density statement of AR parameter, PSD estimation is formed as:
\[ \hat{p}_{\text{cov}} = \frac{\sigma^2}{\left| 1 + \sum_{k=1}^{p} \hat{a}(k) e^{-j2\pi fk} \right|^2} \] (4.15)

Figure 4.1 Read Task for Subject1 Using Covariance Method
Figure 4.2 Relax Task for Subject1 Using Covariance Method

Figure 4.3 Maths Task for Subject1 Using Covariance Method
Figure 4.4 Spell Task for Subject 1 Using Covariance Method

Alpha and Beta wave obtained is converted into data in which the amplitude value is in voltage. To extract the PSD features, we used Covariance algorithm. 12 and 24 features were extracted for the single channel and the two channel system respectively. It is done by using Matlab software version 7.0. These features are used to train and test the neural networks. Figure 4.1 to 4.4 shown power spectrum distribution range for read, relax, maths and spell tasks using covariance algorithm.

4.3.1.2 Burg Method

The Burg method belongs to the class of parametric methods based on AR spectral estimation is based on minimizing the forward and backward prediction errors while satisfying the Levinson-Durbin Recursion. Burg method is one of the techniques proposed by Burg in 1967 for computation of AR modeling coefficients. The Burg method does not use windowing to the data. A number of different combinations of these two parameters have been tested and a model order of 2,300 proved to be appropriate for the data length used. For this method, it is possible to determine the model order according to criteria for minimizing...
the noise influence. However, none of these criteria applied in the present case, will the model order determine, based on a trial and error approach.

\[ x[n] = -\sum_{k=1}^{p} a_k x[n - k] + e[n] \quad (4.9) \]

Where \( x[n] \) is the observed output of the system, \( e(n) \) is the unobserved input data and \( a_k \) are its coefficients. The input \( e(n) \) is considered as zero mean white noise process with unknown variance \( \sigma^2 \), and \( p \) is the order of the system. This model is commonly referred as AR (p). The \( a_k \) coefficients are determined minimizing the forward and backward prediction errors in the least square sense. The PSD estimation is obtained from the following equation:

\[ P_B(f) = \frac{E_p}{\left|1 + \sum_{k=1}^{p} a_p e^{-j2\pi fk}\right|^2} \quad (4.10) \]

Where \( E_p \) is the total least square error of order \( p \). The major advantages of the Burg method are its high frequency resolution, the AR model is always stable and it is computationally very efficient. It exhibits, however, several limitations such as line splitting in the PSD for low signal to noise ratio and frequency shifting from the true frequency, especially for short data record (Lawrence et al., 1987; Kay et al., 1988).

![Figure 4.5 Read Task for Subject1 Using Burg Method](image)
Figure 4.6 Relax Task for Subject1 Using Burg Method

Figure 4.7 Maths Task for Subject1 Using Burg Method
Alpha and Beta wave obtained is converted into data in which the amplitude value is in voltage. To extract the power spectral density features, we used Burg algorithm. 12 and 24 features were extracted for the single channel and the two channel system respectively. It is done by using Matlab software version 7.0. These features are used to train and test the neural networks. Figure 4.5 to 4.8 shown power spectrum distribution range for read, relax, maths and spell tasks using Burg algorithm.

4.3.1.3 Modified Covariance

The Modified Covariance method estimates the AR parameters by minimizing the average of the estimated forward and backward prediction error power.

\[
\hat{h}(n) = - \sum_{k=1}^{p} a(k) h(n - k)
\]

\[
\hat{h}(n) = - \sum_{k=1}^{p} a^*(k) h(n + k)
\]
The modified covariance for estimating the spectral content fitting an AR linear prediction filter model of a given order of signals are used. The input is a frame of consecutive time samples, which is assumed to be the output of an AR system driven by white noise (Hayes et al., 1996).

\( A(k) \) is the AR filter parameter. Modified covariance is found by a minimizing the average of the power estimations of AR parameters

\[
\hat{p} = \frac{1}{2}(\hat{p}^f + \hat{p}^b) \quad (4.18)
\]

Here \( n \) is the exemplification number

\[
\hat{p}^f = \frac{1}{N - P} \sum_{n=0}^{N-1} \left| h(n) + \sum_{k=1}^{p} a(k) h(n - k) \right|^2 \quad (4.19)
\]

\[
\hat{p}^b = \frac{1}{N - P} \sum_{n=0}^{N-1-P} \left| h(n) + \sum_{k=1}^{p} a^*(k) h(n + k) \right|^2 \quad (4.20)
\]

PSD can be acquired by using values of \( a(k) \) in between \( k=1, 2, \ldots, p \). Estimation of white noise variance is acquired with this statement.

\[
\hat{\tau}^2 = c_h h(0,0) + \sum_{k=1}^{p} \hat{a}(k) c_h h(0,k) \quad (4.21)
\]

PSD is acquired with the mathematical statement in the below

\[
ph h(f) = \frac{\hat{\tau}^2}{1 + \sum_{k=1}^{p} \hat{a}(k) e^{-j2\pi fk}} \quad (4.22)
\]

The difference between the modified covariance and covariance technique is the definition of the autocorrelation estimator. Based on the estimates of the AR parameters, PSD prediction is expressed as follows:
\[
\hat{p}_{m,\text{cov}}(f) = \frac{\hat{\sigma}^2}{\left|1 + \sum_{k=1}^{p} \hat{a}(k)e^{-j2\pi f k}\right|^2}
\]  \hspace{1cm} (4.23)

The advantages of the modified covariance method for estimating the parameters of the AR model are yields statistically stable spectral estimates with high frequency resolution.

**Figure 4.9 Read Task for Subject1 Using Modified Covariance Method**

**Figure 4.10 Relax Task for Subject1 Using Modified Covariance Method**
Alpha and Beta wave obtained is converted into data which the amplitude value is in voltage. To extract the power spectral density features, we used Modified Covariance algorithm. 12 and 24 features were extracted for single channel and two channel system respectively. It is done by using Matlab software version 7.0. These features are used to train and test in neural
networks. Figure 4.8 to 4.12 shown power spectrum distribution range for read, relax, maths and spell tasks using Modified Covariance.

4.3.1.4 Yule–Walker Method

It is assumed that the data \( \{x(0), x(1), \ldots, x(N - 1)\} \) are observed. In the Yule-Walker method, the autocorrelation methods as it is sometimes referred to the AR parameters are estimated by minimizing an estimate of prediction error power.

\[
\text{variance} = \rho = \frac{1}{N} \sum_{n=-\infty}^{\infty} \left| x(n) + \sum_{k=1}^{p} a(k)x(n-k) \right|^2 \quad (4.24)
\]

The samples of the \( x(n) \) process which is not observed (i.e., those not in the range \( 0 \leq n \leq N-1 \)) is set equal to zero in Eq.(4.25). The estimated prediction error power is minimized by differentiating Eq.(4.25) with respect to the real and imaginary parts of the \( a(k) \). This may be done by using the complex gradient to yield (Hayes et al., 1996).

\[
\frac{1}{N} \sum_{n=-\infty}^{\infty} \left( x(n) + \sum_{k=1}^{p} a(k)x(n-k) \right)x^{*}(n-1) = 0 \quad (4.25)
\]

With: \( l=1,2, \ldots, p \). This set of equation in terms of autocorrelation function estimates becomes:

\[
r_p + R_p a = 0 \quad (4.26)
\]

Where

\[
r(k) = \left\{ \begin{array}{ll}
\frac{1}{N} \sum_{n=0}^{N-1-k} x^{*}(n)x(n+K), & k = 0,1, \ldots, p \\
r^{*}(-k), & k = (-p + 1), (-p + 2), \ldots, -1
\end{array} \right\} \quad (4.27)
\]

From Eq.(4.30) the are parameter estimates are found as:
\[ a = -R_p^{-1}r_p \]  

(4.28)

The estimate of the white noise variance \( \sigma^2 \) is calculated as:

\[ \sigma^2 = r(0) + \sum_{k=1}^{p} a(k)(-k) \]  

(4.29)

From the estimates of the auto regressive parameters, PSD estimation is given as:

\[ p(f) = \frac{\sigma^2}{|1 + \sum_{k=1}^{p} a(k)e^{-j2\pi fk}|^2} \]  

(4.30)

---

**Figure 4.13 Read Task for Subject1 Using Yule–Walker Method**
Figure 4.14 Relax Task for Subject1 Using Yule–Walker Method

Figure 4.15 Maths Task for Subject1 Using Yule–Walker Method
Alpha and Beta wave obtained is converted into data in which the amplitude value is in voltage. To extract the power spectral density features, we used Yule–Walker algorithm. 12 and 24 features were extracted for the single channel and the two channel system respectively. It is done by using Matlab software version 7.0. These features are used to train and test the neural networks. From the Figure 4.13 to 4.16 shown power spectrum distribution range for read, relax, maths and spell tasks using Yule-Walker algorithm.

4.3.2 Non-Parametric Methods

The simple and easy, non-parametric methods do not assume a fixed structure of a model. It can expand to accommodate the complexity of data. It is based on fewer assumptions like wide sense stationary; hence their applicability is much wider than parametric methods.

4.3.2.1 Welch Method

Welch spectral prediction is a method based on Fast Fourier Transform. The Welch method is also a nonparametric method and it is based on some modification of the
Periodogram aiming to overcome some of the known drawbacks. In the Welch algorithm, the input signal $x$ is segmented into eight sections of equal length, each with 50% overlap. Any remaining entries in $x$ that cannot be included in the eight segments of equal length are discarded. Each segment is windowed with a hamming window that is of the same length as the segment (Subha et al., 2010; Proakis et al., 1996).

$$P_{REF} = \left| \frac{1}{N} \sum_{n=1}^{N} X(n) \exp(-2\pi f) \right|$$

(4.31)

The prediction of power spectral density with Welch method is expressed as follows

$$\hat{P}_{welch}(f) = \frac{1}{L} \sum_{t=0}^{L-1} \hat{s}_{xx}(f)$$

(4.32)

$L$ is the length of the time series. Examining the short data registries with conjoint and non rectangular window reduces the predictive resolution. The Welch method also obtains a better resolution than Periodogram if the signal noise level is low (Subha et al., 2010; Proakis et al., 1996).

![Figure 4.17 Read Task for Subject1 Using Welch Method](image)
Figure 4.18 Relax Task for Subject1 Using Welch Method

Figure 4.19 Maths Task for Subject1 Using Welch Method
Alpha and Beta wave obtained is converted into data in which the amplitude value is in voltage. To extract the Power Spectral Density features, we used Welch algorithm. 12 and 24 features were extracted for the single channel and the two channel system respectively. It is done by using Matlab software version 7.0. These features are used to train and test in neural networks. From Figure 4.17 to 4.20 shown power spectrum distribution range for read, relax, maths and spell tasks using Welch algorithm.

4.3.3 High Resolution Method

High resolution methods aim to separate the observation space in a signal subspace, containing only useful information, and its orthogonal complement, called noise subspace. This composition makes the spectral analysis more robust and highly improves the spectral resolution.

4.3.3.1 Multiple Signal Classification

The MUSIC method belongs to the class of Eigen decomposition methods for which the observed data can be represented by $P$ complex sinusoids in white noise, as follows
$$x[n] = -\sum_{i=1}^{p} A_i e^{j2\pi f_i n} + e[n] \quad (4.33)$$

In (4.33), $A_i$ is the complex amplitude of the $i$-th complex sinusoid, $f_i$ is its frequency, and $e(n)$ is the zero mean white noise in the input $x(n)$. This method uses the eigenvector decomposition of the input signal to obtain two orthogonal subspaces. The $p$ largest Eigenvalues span the signal subspace with dimension $p$. The remaining Eigenvalues span the noise subspace with the dimension $(M-p)$, $M$ being the dimension of the estimated autocorrelation matrix. From the orthogonality condition of both subspaces, the pseudo spectrum can be obtained using the following frequency estimator:

$$P_B(f) = \frac{1}{\sum_{l=p+1}^{M}|s(f)^H V_K|^2} \quad (4.34)$$

Where $V_K$ is the noise eigenvector and $s(f)$ is the signal vector is given as:

$$s(f) = [1 \quad e^{j2\pi f} e^{j4\pi f} \ldots \ldots e^{j2\pi f(N-1)}] \quad (4.35)$$

It must be noted that the pseudo spectrum is not a true PSD estimator, it is used for locating the frequencies contained in the signal, as sharp peaks appear for $f=f_i$. After the determination of the $p$ frequencies, the true power density estimates for the $P$ sinusoids can be performed using the autocorrelation matrix (Lawrence et al., 1987; Kay et al., 1988).

![Figure 4.21 Read Task for Subject1 Using MUSIC Method](image_url)
Figure 4.22 Relax Task for Subject1 Using MUSIC Method

Figure 4.23 Maths Task for Subject1 Using MUSIC Method
Alpha and Beta wave obtained is converted into data in which the amplitude value is in voltage. To extract the PSD features, we used MUSIC algorithm. 12 and 24 features were extracted for the single channel and the two channel system respectively. It is done by using Matlab software version 7.0. These features are used to train and test in neural networks. Figure 4.21 to 4.24 power spectrum distribution range for read, relax, maths and spell tasks using MUSIC algorithm.

4.4 ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks (ANN) are composed of interconnecting artificial neurons. Artificial neural networks may either be used to gain an understanding of biological neural networks, or for solving artificial intelligence problems without necessarily creating a model of a real biological system. The real, biological nervous system is highly complex and a very flexible system. Adaptive nature is the important feature of artificial neural network. An artificial neural network is a mathematical function conceived as a simple model of real biological neurons. This is a simplified model of real neurons known a threshold logic unit.
The input neurons arrive in the form of signal
The signals build up in the cells
Finally the cell discharge through output
The cell can start building up a real neuron is shown in Figure 4.25.

Figure 4.25 Artificial Neuron with Activation Function

There are three layers consists in a network such as namely input layer output layer and hidden layers. In input layer, the neurons receive the external input signal and perform no computation, but simply transfer the input signal to neurons to another layer. Neurons in the hidden layer detect the feature; the weight of the neurons represents the feature the hidden in input patterns. These features are presented to the output layer for determining the output pattern. With one hidden layer, we can represent any continuous function can be represented. The neurons in output layer receive signal from neurons either in the input layer or hidden layer (Yasrebi et al., 2008).

The most important characteristic of an artificial neural network is its ability to learn. Learning is a process in which the network adjusts its parameters the (synaptic weights) in response to input stimuli so that the actual output response converges to the desired output response converges to the desired output response. When the actual output response in the same as the desired one, the network has completed the learning phase and the network has
acquired knowledge. Learning or training categorized into supervised, unsupervised and reinforced training. The supervised training is commonly used.

Supervised training requires the pairing of each input vector with a target vector representing the desired output. During the training session an input vector is applied to the network, and its result response differs from the target response, the network generates an error signal. The error signal is then used to calculate the adjustment that should be made in the synaptic weights so that the actual output matches the target output. The error minimization in this kind of training requires a supervisor or teacher, hence the name supervised training.

Unsupervised training is employed in self organizing neural net. In contrast to supervised learning, it does not require a teacher. In this method of training the input vector similar type of grouped without the use of training, the input vector of similar types of grouped without the use of training data to specific how a typical member of each group looks. During this training the neural network receives the input pattern and organizes this pattern into categories. Even though unsupervised training does not require a teacher, it requires certain guidelines to form groups based on color, shape or any other properties. Meanwhile reinforced training, it is similar to supervised training. In this method, the teacher does not indicate how close the actual output to the desired output is, but yields only pass or fail indication. Thus, the error generated during reinforced training is binary.

4.5 CLASSIFICATION USING NEURAL NETWORK

ANN Classification is an example of supervised learning. Multilayer networks are applied with success to determine different issues by training them in a supervised manner with an extreme algorithmic rule called back-propagation algorithmic rule. ANN classification is the process of learning to separate samples into different classes by finding common features between samples of known classes. Making guesses on unknown samples is regularly a way of testing the ANN classifier. Two types of samples have used to distinguish samples,
one is called testing samples and other is training samples. The performance of ANN algorithm was accessed by following measures:

- **True Positive (TP)**
  The ANN identifies an input signal exactly, classifying and labeling the signal correctly.
  \[ TPR = \frac{TP}{P} \]  (4.36)
  Positive = (True Positive + False Positive)

- **True Negative (TN)**
  The ANN identifies an input signal exactly, classifying and labeling the signal differently.
  \[ TNR = \frac{TN}{N} \]  (4.37)
  Negative = (True Negative + False Negative)

- **False Positive (FP)**
  The ANN identifies a different input signal, classifying and labeling the signal correctly.
  \[ FN = \frac{FP}{(FP + TN)} \]  (4.38)

- **False Negative (FN)**
  The ANN identifies a different input signal, classifying and labeling the signal differently.
  \[ FN = \frac{FN}{(TP + FN)} \]  (4.39)
\( \text{Sensitivity (SE)} \)

The ability of the classifier to detect the exact EEG signals. Number of True Positive (TP) results divided by sum of True Positive (TN) and False Negative (FN) are expressed in percentage

\[
\text{SE} = \frac{TP}{(TP + FN)} \times 100
\]

\( \text{Specificity (SE)} \)

The ability of the classifier to detect the different brain signals. Specificity of a test refers to the possibility that test results will be negative among subject whose EEG signals are not clear. Number of True Negative (TN) results divided by sum of True Negative (TN) and False Negative (FN) are expressed in percentage

\[
\text{SE} = \frac{TN}{(TN + FN)} \times 100
\]

\( \text{Positive Productivity (PD)} \)

Number of True Positive (TP) results divided by sum of True Positive and False Positive (FP) results are expressed in percentage

\[
\text{Positive Predictivity} = \frac{TP}{(TP + FP)} \times 100
\]

4.6 STEPS IN DESIGNING THE NEURAL NETWORK

Design and development of neural networks for classification purpose consist of the following steps are:

4.6.1 Data Set Pre-Processing

Features extracted from the EEG signals must be pre processed to provide feature values in the proper range.
4.6.2 Training and Testing the Sets

Common observation is to divide the time series into three different sets, training, testing and validation sets. The training set is the largest set and it is used by the neural network to learn the patterns in the data. The testing set ranging in the size from 10% to 30% of the training set is used to evaluate the generalization ability of the supposedly trained network. The network that performs best on the training set is chosen for the real time testing. The ratio between the training and testing sets chosen in our experiment varies from 80%:20%.

4.6.3 Number of Hidden Layers

Hidden layer provides the network with its ability to generalize. Within the neural network one hidden layer with sufficient number of hidden neurons is capable of approximate any continuous function. In practice a neural network with three hidden layers are widely used and have performed well. Increasing the number of hidden layers also increases the computational time and danger of over fitting which leads to out-of-sample recognition performance. The FFNN used in this research is designed with one hidden layer.

4.6.4 Number of Hidden Neurons

It is necessary to notice that the rules which calculate the amount of hidden neurons as a multiple number of input neurons implicitly assume that the training set is at least twice as large as the number of weights and preferably four or more times as large. If it is not case then the thumb case can quickly lead to the over fitted models since the hidden neurons are directly dependent upon the input neurons. The solution is either increase the size of the training set or to a set an upper limit on the number of input neurons so that the number of weights is at least half the number of training facts. Selecting the number of best hidden neurons involves experimentation. The three hidden neurons for FFNN are chosen experimentally, depending upon the features selected.
4.6.5 Number of Output Neurons

Deciding the number of output neurons is somewhat more straightforward depending on the number of classifications. In this research, five and three output neurons are used in the FFNN models using single channel and two channels respectively.

4.6.6 Transfer Functions

The transfer function is mathematical formulas that determine the output of a processing neuron. They are referred as transformation. The purpose of training function is to prevent output from reaching very large values which can paralyze neural networks and their training. The sigmoid transfer function is chosen for the FFNN as the input and output sets.

4.6.7 Evaluation Criteria

The most common function utilized in NN is the sum of the squared errors. Other error functions offered by software vendors include least absolute deviations, least fourth power, asymmetric least squares and percentage difference.

4.6.8 Neural Network Training

Training a NN to learn the patterns in the data involves iteratively presenting it with examples of correct answers; the objective of the training is to set the weights between the neurons that determine the global minimum of the error function. Unless the model is over fitted, this set of weights should provide good generalization. The FFNN neural network is trained using the Back Propagation training algorithm.

4.6.9 Number of Training Iterations

In this approach to training is stopped only if there is no improvement in the error function based on a reasonable number of selected starting weights. Training is stopped after a
predetermined number of iterations and the network’s ability to generalize on the testing set is evaluated and training is resumed.

4.6.10 Neural Network Models

Neural networks have been successfully used in a variety of medical applications. Recent advances in the field of neural networks have made them attractive for analyzing signals. The application of neural networks has opened a new area for solving problems not resolvable by other signal processing techniques. In contrast to the conventional spectral analysis methods, ANNs not only model the signal, but also make a decision as to the class of signal. Another advantage of ANN analysis over existing methods of biomedical signal analysis is that, after an ANN has trained satisfactorily and the values of the weights and biases have been stored, testing and subsequent implementation is rapid. During implementation of the combined neural networks the predictions of the networks in the first-level were combined with second-level neural network. The results of the present study showed that significant improvement is achieved with accuracy by applying neural networks as the second-level model compared to the stand-alone multilayer perception.

4.7 FEED FORWARD NEURAL NETWORK BASED CLASSIFIER

Feed-forward neural network (FFNN), which is shown in Figure 4.26 one of the most common and first developed types of ANN. Back propagation algorithm resembles multilayer Feed Forward Network. The errors propagate backwards from output nodes to the input nodes. Network activation flow in one direction only, from the input layer to output layer passing through the hidden layer. Each unit is connected in the forward direction to every unit in the next layer. It is a multilayer feed forward network with one layer of hidden units. The input layer is connected to hidden layer and output layer is connected by means of interconnection weights. The bias is provided for both hidden and output layer to act upon the net input. Mathematical formula used an algorithm can be applied to any network which is the advantage of using BPN (Bishop et al., 1995). The architecture of the FFNN is shown in Figure 4.26.
4.7.1 BackPropagation Learning Method

The most widely used neural-network learning method is the BP algorithm. This backpropagation algorithm can easily train a neural network. Learning in a neural network involves modifying the weights and biases of the network in order to minimize a cost function. The cost function always includes an error term a measure of how close the network's predictions are to the class labels for the examples in the training set. Additionally, it may include a complexity term that reacts a prior distribution over the values that the parameters can take.

The activation function considered for each node in the network is the binary sigmoid function defined by $s = 1$ as output $= 1/(1+e^{-x})$, where $x$ is the sum of the weighted inputs to that particular node. This is a common function used in many FFNN. This function limits the output of all nodes in the network to be between 0 and 1. Note all neural networks are basically trained until the error for each training iteration stopped decreasing. Noise FFNN models were developed and trained and tested with the band power features. The back propagation training algorithm involves three stages (Sivanandam et al., 2003) the feed
forward of the input training pattern, the calculation and back propagation of the associated weight error and the weight adjustments. The training algorithm is shown below

x : input training vector

\[ x = x_1, x_2, \ldots, x_i, \ldots, x_n. \]

t : Output target vector

\[ t = t_1, t_2, \ldots, t_j, \ldots, t_m. \]

δ : Portion of error correction weight adjustment for \( w_{jk} \) that is due to an error at the output unit \( Y_k \), which is back propagated to the hidden units that feed into \( Y_k \).

\( \delta_j \) : Portion of error correction weight adjustments for \( v_{ij} \) that is due to the back propagation of error to the hidden unit \( Z_j \).

α : Learning rate.

\( X_i \) : \( i^{th} \) input unit.

\( v_{oj} \) : Bias on \( j^{th} \) hidden unit.

Net input to \( Z_j \),

\[ z_{inj} = v_{oj} + \sum i x_i v_{ij} \]

And

\[ z_j = f(z_{inj}) \]

If \( w_{ok} \) is the bias on \( k^{th} \) output unit and \( Y_k \) is the \( k^{th} \) output unit. Then the net input to \( Y_k \)

\[ y_{ink} = w_{ok} + \sum j z_j w_{jk} \]

And

\[ y_k = f(y_{ink}) \]

STEP1: Initialize the weights.

STEP2: While stopping condition is false, do steps 3 to 10.

STEP3: For each training pair \( x:t \), do steps 4 to 9
STEP4: Each input unit $X_i$, $i=1, 2… n$ receives the input signal, $x_i$ and broadcasts it to the next layer.

STEP5: For each hidden layer neuron denoted as $Z_j$, $j=1,2,…,p$.
$z_{inj}=v_{oj}+\sum_i x_i v_{ij}$
$z_j=fz_{inj}$
Broadcast $z_j$ to the next layer.

STEP6: For each output neuron $Y_k$, $k =1, 2,…,m$
$y_{ink}=w_{ok}+\sum_j z_jw_{jk}$
$y_k=fy_{ink}$

STEP7: Compute $\delta_k$ for each output neuron, $Y_k$.
$\delta_k=t_k-y_kf'_{y_{ink}}$
$\Delta w_{jk}=\alpha \delta_k z_j$
$\Delta w_{ok}=\alpha \delta_k$, since $z_0=1$

STEP8: For each hidden neuron
$\delta_{inj}=\sum_k=1m \delta_kw_{jk}$ $j=1,2,…,p$
$\delta_j=\delta_{inj}f'_{z_{inj}}$
$\Delta v_{ij}=\alpha \delta_j x_i$
$\Delta v_{oj}=\alpha \delta_j$

STEP9: Update weights
$w_{jk}^{new}=w_{jk}^{old}+ \Delta w_{jk}$
$v_{ij}^{new}= v_{ij}^{old}+ \Delta v_{ij}$

STEP10: Test for stopping condition.
The FFNN is trained using Levenberg Back Propagation training algorithm because it finds a solution even if it starts very far off the final minimum. The network is modeled with 9 hidden neurons, which is usually determined by a number of trial and error method runs. 75% of the data is used for training and 100% data are used for testing the network model for both single and double channel systems. The FFNN is modeled using 12 and 24 input neurons for single channel and two channel system. Respectively five and four output neurons for single channel and two channel system are used to verify the individuals. The learning rate is chosen as 0.0001. Training is conducted until the average error falls below 0.001 or reaches maximum iteration limit of 1000 and testing error tolerance is fixed at 0.5.

4.8 RECURRENT NEURAL NETWORK

The ANN architecture that we explore for modeling and classifying EEG is the Recurrent Neural Network (RNN) are shown in Figure 4.27. RNN were originally developed by Jeffrey Elman in 1990 for finding patterns in natural language and have since been successfully applied to a number of practical problems. In addition to their history of successful application, RNN are appealing because it has been demonstrated that they are universal approximates of finite state machines 36.

![Figure 4.27 Architecture of Recurrent Neural Network](image)
In other words, any given finite state machine can be simulated by some RNN given enough hidden units and the proper connection weights. An RNN consists of two distinct layers. The first layer, referred to as the hidden layer, is composed of a number of neurons with sigmoidal activation functions. The number of neurons in the hidden layer, also referred to as hidden units, is a parameter given during the construction of an ERNN. Each hidden unit has full incoming connections from each input as well as a constant bias value of one. Additionally, each hidden unit has full recurrent connections between every other hidden unit with a single time step delay. In other words, the current value of each network input and the output of every hidden unit in the previous time step are fed into each hidden unit at the current time step.

The values stored in the delay lines of the recurrent connections of an RNN fully represent the state of the network at any given time. As such, we refer to these values as the context of the network. As is common practice, the initial context of our RNN is set to the zero vectors with the assumption that the network will sufficiently acclimate to the input signal and achieve an acceptable context after some initial transient period. The second layer, referred to as the visible layer, consists of one neuron per output. Each neuron in the visible layer, also referred to as a visible unit, is strictly linear, i.e., the output of each visible unit is a weighted sum of the outputs of the hidden units and a constant bias value of one. In order to formalize the RNN architecture, let us begin by defining the following constant:

- L: the number of network input
- M: the number of hidden units
- N: the number of network outputs

Next, let \( x_t \) be the \( L \times 1 \) vector of inputs to our network at time \( t \) and \( z_t \) be the \( M \times 1 \) output of our hidden layer at time \( t \). Also, let \( \tilde{x}_t \) be \( x_t \) with a constant one appended and \( \tilde{z}_t \) be \( z_t \) with a constant one appended to our bias values. Then the output of our hidden layer at time \( t \) can be defined by the following recurrence relation:

\[
Z(t) = \varphi(H\tilde{x}(t) + Sz(t - 1))
\]  

(4.43)
Where $H$ is the $M \times L + 1$ matrix of feed forward weights in the hidden layer, $S$ is the $M \times M$ matrix of recurrent weights in the hidden layer and $\phi$ is our choice of sigmoid. Note that we define our initial context $z_0 = 0$ as discussed earlier. Finally, the output of our visible layer at time $t$ can be defined as

$$y(t) = V \bar{z}(t)$$  \hspace{1cm} (4.44)

Where $V$ is the $N \times M + 1$ weight matrix for our visible layer. For the sake of notational brevity, we denote the output of an ERNN at time $t$ as

$$y(t) = emn(x(t))$$  \hspace{1cm} (4.45)

The RNN with one single hidden layer is trained to identify the architecture of the RNN. The RNN is trained using Gradient Descent Back Propagation algorithm because it finds a solution even if it starts very far off the final minimum. The network is modeled with 9 hidden neurons, which is chosen based on trial and error method. 75% of the data is used for training and 100% data are used for testing the network model for both single and double channel systems. The RNN is modeled using 12 and 24 input neurons for single channel and two channel system. Respectively five and four output neurons for single and two channel system are used to verify the individuals. The learning rate is chosen as 0.0001. Training is conducted until the average error falls below 0.001 or reaches maximum iteration limit of 1000 and testing error tolerance is fixed at 0.5.

### 4.9 DESIGN OF GRAPHICAL USER INTERFACE

Matlab is a commercially available, widely used, interactive, technical computing software. Matlab versions 7.0 and higher provide serial communication functionality, GUI testing is a process to test application's user interface and to detect if application is functionally correct. GUI testing involves carrying set of tasks and comparing the result of same with the expected output and ability to repeat the same set of tasks multiple times with different data input and same level of accuracy. GUI testing can be performed both manually, by a human tester or could be performed automatically with the use of a software program. For this research, the GUI is used to show the authorized subject’s name which is shown Figure 4.28 to 4.32.
Figure 4.28 Analysis of EEG Signals Using GUI for Subject 1

Figure 4.28 Analysis of Signals Using GUI for Subject 1

Figure 4.30 Analysis of EEG Signals Using GUI for Subject 2
Figure 4.30 Analysis of EEG Signals Using GUI for Subject 3

Figure 4.31 Analysis of EEG Signals Using GUI for Subject 7
4.10 SUMMARY

This chapter describes the feature extraction techniques and the classification process used. Feature extraction techniques, namely PSD with parametric and non-parametric and high resolution methods are discussed briefly. Parametric methods, the Burg method, Covariance method, Modified Covariance and Yule-Walker methods are used. The techniques behind these methods are discussed briefly. In the non-parametric methods, Welch method is used and Multiple Signal Classification method from high resolution method is also used. In the classification process, two networks, namely FFNN and RNN are used for the classification of EEG brain signals. The different features of the network model, namely training time and testing time, no of input units, no of output units and no of hidden layers are discussed. A GUI system is developed for the final stage, which is verified the subjects for their corresponding signal. The performance result of these networks are discussed and tabulated in Appendix D to AA.