CHAPTER VI
CLASSIFICATION OF ELECTROCARDIOGRAM SIGNALS WITH EXTREME LEARNING MACHINE

6.1. EXTREME LEARNING MACHINE

Extreme Learning Machines (ELM) has very high potential that can resolve problems of data regression and classification. This technique overcomes some challenging constraints on the use of feed-forward neural networks and other computational intelligence methods. Recent developments of ELM technique have shown that ELM inherits the advantages of both neural networks and support vector machines while having much faster learning speed, requiring less human intervene and robust property with respect to the parameter selection.

![Figure 6.1: An Example of ELM](image)
All the parameters of ELMs can be analytically determined rather than being tuned. This algorithm provides good generalization performance at extremely fast learning speed [113].

From function approximation point of view ELM is very different compared to the traditional methods. ELM shows that the hidden node parameters can be completely independent from the training data.

- In conventional learning theory and implementations, the training data has to be seen before generating the hidden node parameters.
- In ELM learning theory and implementations, the hidden node parameters can be generated before seeing the training data.

6.1.1. Salient Features of ELM

Compared to popular Back Propagation (BP) Algorithm and Support Vector Machine (SVM), ELM has several salient features:

- **Ease of Use**: Except predefined network architecture, no other parameters need to be manually tuned. Users need not have to spend several hours or days tuning and training learning machines.

- **Faster Learning Speed**: The time taken for most of the training will be in milliseconds, seconds and minutes. Other conventional methods cannot provide such a fast learning speed.

- **Higher Generalization Performance**: The generalization performance of ELM is better than SVM and back propagation in most cases.

- **Applicable for all Nonlinear Activation Functions**: All piecewise continuous functions which includes discontinuous, differential, non-differential functions can be used as activation functions in ELM.

- **Applicable for Fully Complex Activation Functions**: Complex functions can also be used as activation functions in ELM.
It is known that traditionally all the parameters of the feed forward networks need to be tuned and thus there exists the dependency between different layers of parameters. Gradient descent-based methods have been used in various learning algorithms of feed forward neural networks. It is very clear that gradient descent based learning methods are generally very slow due to improper learning steps or may easily converge to local minima. And many iterative learning steps are required by such learning algorithms in order to obtain better learning performance.

6.1.2. Optimization Method Based Solution to ELM

This section shows that with the standard optimization method ELM can be linearly extended to SVM (with less optimization constraints) and the implementation of SVM can be made much simpler.

Given a set of training data \((x_i, t_i), i = 1, ..., N\), where \(x_i \in \mathbb{R}^d\) and \(t_i \in \{-1, 1\}\), these training data are usually not separable in the input space in most cases. If the training data cannot be separated by a linear decision function \(w \cdot x + b = 0\) one can map the training data \(x_i\) from the input space to a feature space \(Z\) through a mapping \(\phi(x): x_i \rightarrow \phi(x_i)\).

Unlike SVM, any set of distinct training data transformed from the input space to the ELM feature space with the ELM mapping \(h(x)\) are linearly separable in the ELM feature space with probability one. However, it is most possible that some testing data may be within the classification margin if zero training error is strictly obtained. In this sense, one may wish to separate the training data with an acceptable minimal training error instead of the zero training error so that the testing error can thus be minimized accordingly

\[
\beta \cdot h(x_i) \geq 1 - \xi_i \quad \text{If } t_i = 1
\]
\[
\beta \cdot h(x_i) \leq 1 + \xi_i \quad \text{If } t_i = -1
\]

(6.1)
That is

\[ t_i \beta \cdot h(x_i) \geq 1 - \xi_i, \quad i = 1, \ldots, N. \] (6.2)

where \( \beta \) is the output weight, \( h(x) \) actually maps the data from the d-dimensional input space to the L-dimensional hidden layer feature space (ELM feature space) \( H \).

Thus, from the standard optimization theory point of view, the objective of ELM in minimizing both the training errors and the output weights can be written as

\[
\text{Minimize:} \quad L_p = \frac{1}{2} ||\beta||^2 + c \sum_{i=1}^{N} \xi_i \quad \text{Subject to:} \quad t_i \beta \cdot h(x_i) \geq 1 - \xi_i, \quad i = 1, \ldots, N
\]

(6.3)

\[ \xi_i \geq 0, \quad i = 1, \ldots, N. \]

This is very similar to SVM’s optimization problem with two main differences:

- Different from the conventional SVM the randomness is adopted in the ELM mapping \( h(x) \), that is, all the parameters of \( h(x) \) are chosen randomly.
- The bias \( b \) is not required in the ELM’s optimization constrains since in theory these separating hyper plane in the ELM feature space passes through the origin.

The Lagrange function of the primal ELM optimization is

\[
L_{ELM}(\beta, \xi, \alpha, \mu) = \frac{1}{2} \beta \cdot \beta + c \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i(t_i \beta \cdot h(x_i) - (1 - \xi_i)) - \sum_{i=1}^{N} \xi_i \mu_i
\]

(6.4)
where \( \alpha_i \) and \( \mu_i \) are the Lagrange multipliers and are non-negative values. In order to find the optimal solutions of equation 5.4, we should have

\[
\frac{\partial L_{ELM}(\beta, \xi, \alpha, \mu)}{\partial \beta} = 0 \Rightarrow \beta = \sum_{i=1}^{N} \alpha_i t_i h(x_i) \tag{6.5}
\]

\[
\frac{\partial L_{ELM}(\beta, \xi, \alpha, \mu)}{\partial \xi} = 0 \Rightarrow c = \alpha_i + \mu_i, \forall i \tag{6.6}
\]

Substitute (6.5) and (6.6) into (6.4) and to train ELM for classification is then equivalent to solving the following dual optimization problem

Minimize: \( L_D = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} t_i t_j \alpha_i \alpha_j h(x_i) \cdot h(x_j) - \sum_{i=1}^{N} \alpha_i \)

Subject to: \( 0 \leq \alpha_i \leq c, i = 1, ..., N \)

Different from the conventional dual SVM optimization problem, the above dual ELM optimization problem does not have the condition \( \sum_{i=1}^{N} \alpha_i t_i = 0, \forall i \) due to the truth that in theory the separating hyper plane in the ELM feature space tends to pass through the origin.

### 6.1.3. ELM Kernel

An ELM kernel function can be defined as

\[
K_{ELM}(x_i, x_j) = h(x_i) \cdot h(x_j)
\]

\[
= [G(a_1, b_1, x_i), ..., G(a_L, b_L, x_i)]^T [G(a_1, b_1, x_j), ..., G(a_L, b_L, x_j)]^T \tag{6.8}
\]

where \( G(a, b, x) \) a nonlinear is piecewise continuous function satisfying ELM universal approximation capability theorems [112–114] and \( \{(ai, bi)\}_{i=1}^{L} \) are randomly generated according to any continuous probability distribution.

Thus,

Minimize: \( L_D = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} t_i t_j K_{ELM}(x_i, x_j) \alpha_i \alpha_j - \sum_{i=1}^{N} \alpha_i \)

Subject to: \( \sum_{i=1}^{N} t_i \alpha_i = 0, 0 \leq \alpha_i \leq \lambda, i = 1, ..., N \)

The decision function of ELM is
6.2. APPLICATION OF EXTREME LEARNING MACHINE FOR ECG CLASSIFICATION

The preprocessing and feature selection phase discussed in chapter 4 are also used in this chapter. After the preprocessing steps, the classification of the ECG signals is carried out through ELM. The complete process of the proposed approach is shown in the figure 6.2.

\[
f(x) = \text{sign} \left( \sum_{s=1}^{N_s} \alpha_s x_s K_{ELM}(x, x_s) \right)
\]  

(6.10)

Figure 6.2: Block diagram of the Classification Process using ELM
Algorithm of ECG Classification using ELM with DWT-AR modeling

**Step 1:** Input ECG signals are taken from MIT-BIH arrhythmia database.

**Step 2:** Raw ECG signal is typically contaminated by noise components. Morphology Filter (MF) is used to remove these high frequency noise components and baseline drift with low distortion and less computational burden.

**Step 3:** DWT is used to extract P-QRS-T morphological features of a beat.

**Step 4:** Third-order cumulant based AR Modeling is used to select the temporal features of a beat. It handles the variations of signal among the same type of beats belonging to the same type of arrhythmia.

**Step 5:** Features selected from the beat through the step3 and step4 are sent as Vector set to ELM Classifier. Classification of ECG signals is done and the output of the classifier denotes the type of the arrhythmia.

### 6.2.1. ELM Classification

A new learning algorithm called the Extreme Learning Machine for Single-hidden Layer Feed forward Neural networks (SLFNs) supervised batch learning. The output of an SLFN with $\tilde{N}$ hidden nodes (additive or RBF nodes [67]) can be represented by

$$f_{\tilde{N}}(X) = \sum_{i=1}^{\tilde{N}} \beta_i G(a_i, b_i, X), \quad X \in \mathbb{R}^n, \quad a_i \in \mathbb{R}^n,$$

(6.11)
where $a_i$ and $b_i$ are the learning parameters of hidden nodes and $\beta_i$ is the weight connecting the $i^{th}$ hidden node to the output node. $G(a_i,b_i,X)$ is the output of the $i^{th}$ hidden node with respect to the input $x$. For the additive hidden node with the activation function $g(x): \mathbb{R} \rightarrow \mathbb{R}$ (e.g., sigmoid or threshold), $G(a_i,b_i,X)$ is given by

$$G(a_i,b_i,X) = g(a_i \cdot x + b_i), b_i \in \mathbb{R}$$

(6.12)

where $a_i$ represents the weight vector connecting the input layer to the $i^{th}$ hidden node and $b_i$ is the bias of the $i^{th}$ hidden node. $a_i \cdot x$ denotes the inner product of vectors $a_i$ and $x$ in $\mathbb{R}^n$. For an RBF hidden node with an activation function $g(x): \mathbb{R} \rightarrow \mathbb{R}$ (e.g., Gaussian), $G(a_i,b_i,X)$ is given by

$$G(a_i,b_i,X) = g(b_i ||x - a_i||), b_i \in \mathbb{R}^+$$

(6.13)

where $a_i$ and $b_i$ are the $i^{th}$ RBF node’s center and impact factor. $\mathbb{R}^+$ indicates the set of all positive real values. The RBF network is a special case of the SLFN with RBF nodes in its hidden layer. Each RBF node has its own centroid and impact factor and output of it is given by a radially symmetric function of the distance between the input and the center.

In the learning algorithms it uses a finite number of input-output samples for training. Here, $N$ arbitrary distinct samples are considered $(x_i,t_i) \in \mathbb{R}^n \times \mathbb{R}^m$, where $x_i$ is an $n \times 1$ input vector and $t_i$ is an $m \times 1$ target vector. If an SLFN with $\bar{N}$ hidden nodes can approximate $N$ samples with zero error, it then implies that there exist $\beta_i$, $a_i$ and $b_i$ such that

$$f_{\bar{N}}(X_j) = \sum_{i=1}^{\bar{N}} \beta_i G(a_i,b_j,X_j) = t_j, j = 1, \ldots, N$$

(6.14)

Equation (6.14) can be written compactly as

$$H\beta = T$$

(6.15)
where

\[
\begin{align*}
H(a_1, \ldots, a_{\tilde{N}}, b_1, \ldots, b_{\tilde{N}}, X_1, \ldots, X_{\tilde{N}}) &= \\
\begin{bmatrix}
G(a_1, b_1, X_1) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, X_1) \\
\vdots & \ddots & \vdots \\
G(a_1, b_1, X_N) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, X_N)
\end{bmatrix}_{N \times \tilde{N}}
\end{align*}
\]

(6.16)

\[
\beta = \begin{bmatrix}
\beta_1^T \\
\vdots \\
\beta_{\tilde{N}}^T
\end{bmatrix}_{\tilde{N} \times m} \quad \text{and} \quad T = \begin{bmatrix}
t_1^T \\
\vdots \\
t_{\tilde{N}}^T
\end{bmatrix}_{N \times m}
\]

(6.17)

H is called the hidden layer output matrix of the network [105]; the \(i^{th}\) column of \(H\) is the \(i^{th}\) hidden node’s output vector with respect to inputs \(x_{1}, x_{2}, \ldots, x_{N}\) and the \(j^{th}\) row of \(H\) is the output vector of the hidden layer with respect to input \(x_{j}\).

In real applications, the number of hidden nodes, \(\tilde{N}\), will always be less than the number of training samples \(N\) and hence, the training error cannot be made exactly zero but can approach a nonzero training error. The hidden node parameters \(a_i\) and \(b_i\) (input weights and biases or centers and impact factors) of SLFNs need not be tuned during training and may simply be assigned with random values according to any continuous sampling distribution. Equation (6.15) then becomes a linear system and the output weights are estimated as

\[
\tilde{\beta} = H^T T
\]

(6.18)

where \(H^T\) is the Moore-Penrose is generalized inverse [117] of the hidden layer output matrix \(H\). The ELM algorithm which consists of only three steps, can then be summarized as

**ELM Algorithm:**

Given a training set \(X = \{(X_i, t_i) | X_i \in \mathbb{R}^n, t_i \in \mathbb{R}^m, i = 1, \ldots, N\}\) activation function \(g(x)\) and hidden node number \(\tilde{N}\),
1) Assign random hidden nodes by randomly generating parameters \((a_i, b_i)\) according to any continuous sampling distribution, \(i=1, \ldots, N\).

2) Calculate the hidden layer output matrix \(H\).

3) Calculate the output weight \(\tilde{\beta} = H^T\).

The universal approximation capability of ELM has been analyzed by Huang et al. [112] using an incremental method and it shows that single SLFNs with randomly generated additive or RBF nodes with a wide range of activation functions can universally approximate any continuous target functions in any compact subset of the Euclidean space \(\mathbb{R}^n\). \(g(x) = \frac{1}{1 + e^{-\lambda x}}\) is the sigmoidal function used as activation function in ELM.

The following command is used to carry out the ECG signal classification using ELM. The extracted morphological and temporal features are given as input to the ELM classifier.

```matlab
function [class] = elm (train_data, test_data, Elm_Type, NumberofHiddenNeurons, ActivationFunction)
% Input:
% TrainingData_File - Filename of training data set
% TestingData_File - Filename of testing data set
% Elm_Type - 0 for regression; 1 for (both binary and multi-
% classes) classification
% NumberofHiddenNeurons - Number of hidden neurons
% assigned to ELM
% ActivationFunction - Type of activation function:
% 'sig' for Sigmoidal function
% 'sin' for Sine function
% 'hardlim' for Hardlim function
% 'tribas' for Triangular basis function
% 'radbas' for Radial basis function
% Output:
% TrainingTime - Time (seconds) spent on training ELM
% TestingTime - Time (seconds) spent on predicting ALL
% testing data
% TrainingAccuracy - Training accuracy
% TestingAccuracy - Testing accuracy
```
6.3. EXPERIMENTAL RESULTS

This section provides the experimental evaluations of the ELM classification approach. The results are presented in graph and tables.

6.3.1. Dataset Description

The experimental setup and dataset used in the chapter V is taken up for this proposed approach. The performance of the ELM classification preprocessing and feature selection approach is compared with standard RBF and k-NN approaches.

6.3.2. Experimental Setup

The total number of training beats was fixed to 500, as reported in Table 5.1 as given in chapter V. For comparison purpose, two other reference nonparametric classification approaches are implemented, namely, the k-Nearest Neighbor (kNN) and the Radial Basis Function (RBF) neural network classifiers. It was desired to explore the accuracy of the ELM classifier when integrated within a standard classification scheme based on an AR feature selection and is compared with k-NN and RBF approaches.

6.3.3. Performance Evaluation

The performance of the proposed approach is evaluated based on the accuracy of the classification and the average number of detected features. Figure 6.3 shows the beats classification using ELM.
Figure 6.3: Beats Classification using ELM with DWT-AR Modeling

A. Classification Accuracy

Table 6.1 and figure 6.4 show the classification accuracy results of the classifier such as ELM, RBF and kNN. It is clearly observed from the classifiers that the ELM classifier outperforms the other two RBF and kNN classifiers in terms of Accuracy. For the patient ID 202, the accuracy obtained for the ELM approach is 92.55% whereas the accuracy obtained for RBF and kNN approaches are 83.50% and 80.46% respectively. Similarly, for other patients, the accuracy obtained is very high when compared to RBF and kNN approaches.

Accuracy of kNN

\[
\text{Accuracy (\%) = } \frac{1688 + 29}{1688 + 29 + 7 + 410} \times 100 = 80.46\%
\]

Accuracy of RBF

TP=1752, TN=30, FN=6, FP=346
Accuracy of ELM with Preprocessing and Feature Selection

\[ \text{Accuracy} (\%) = \frac{1752 + 30}{1752 + 30 + 6 + 346} \times 100 = 83.50\% \]

Table 6.1

Classification Accuracy Comparison of kNN, RBF and ELM with Feature Selection

<table>
<thead>
<tr>
<th>Patient ID</th>
<th>kNN</th>
<th>RBF</th>
<th>ELM with Preprocessing and Feature Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>202</td>
<td>80.46</td>
<td>83.50</td>
<td>92.55</td>
</tr>
<tr>
<td>203</td>
<td>81.23</td>
<td>84.20</td>
<td>93.74</td>
</tr>
<tr>
<td>208</td>
<td>80.98</td>
<td>83.55</td>
<td>95.73</td>
</tr>
<tr>
<td>212</td>
<td>81.69</td>
<td>84.98</td>
<td>94.12</td>
</tr>
</tbody>
</table>

Figure 6.4 shows the graphical representation of the classification accuracy of the classifiers such as ELM with Preprocessing and Feature Selection, RBF and k-NN. It is clearly observed from the figure that the ELM with Preprocessing and Feature Selection approach provides better results when compared with the RBF and k-NN approaches.
From this experiment, three observations can be made: 1) the ELM classifier shows a relatively low sensitivity to the curse of dimensionality as compared to the kNN and the RBF classifiers 2) the ELM classifier still preserve its superiority when integrated in a feature reduction-based classification scheme; and 3) though the ELM performs well in the whole original feature space, its accuracy can still be improved provided that a subspace of higher generalization capability can be found.

**B. Number of Features Detected**

DWT with AR modeling is used for the feature selection process. Table 6.2 shows the number of features detected automatically to discriminate each class from the others. The average number of features required by the ELM classifier is 65. The average number of features is calculated from 5 runs, while the minimum and maximum numbers of features were obtained for the Ventricular premature (V) and Normal (N) classes with 64 and 66 features, respectively.
Table 6.2
Number of Features Detected for each Class with the ELM Classification System Trained on 500 Beats

<table>
<thead>
<tr>
<th>Feature Selection Techniques</th>
<th>CLASS</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N</td>
<td>A</td>
<td>V</td>
<td>RB</td>
<td>/</td>
<td>LB</td>
</tr>
<tr>
<td>PSO-SVM</td>
<td>63</td>
<td>43</td>
<td>35</td>
<td>43</td>
<td>40</td>
<td>42</td>
</tr>
<tr>
<td>ELM- DWT and AR Modeling</td>
<td>66</td>
<td>46</td>
<td>64</td>
<td>46</td>
<td>43</td>
<td>38</td>
</tr>
</tbody>
</table>

The average number of features detected by ELM with DWT and AR modeling is 65 and the overall classification accuracy is high when compared to the PSO-SVM. So, ELM with DWT and AR modeling detects the best features for better classification.

D. Overall Accuracy (OA) Vs. Number of Selected Features

Figure 6.5 shows the effect of OA against the number of features selected for the test beats with the PSO–SVM and ELM -DWT and AR Modeling.

Figure 6.5: Overall Accuracy (OA) versus number of selected features achieved on the test beats with the PSO–SVM and ELM-DWT and AR Modeling
The OA of ELM-DWT and AR Modeling is very high when comparing with the PSO–SVM. ELM-DWT and AR Modeling shows low sensitivity to the curse of dimensionality as compared to the PSO–SVM.