ELECTROCARDIOGRAM ABNORMALITY DETECTION BASED ON MULTIPLE MODELS

Gangyi Zhu, Congrong Guan, Ke Tan, Peidong Wang

Department of Computer Science and Engineering
The Ohio State University, Columbus, OH 43210

1. ABSTRACT

This paper introduces a study of applying three independent machine learning approaches to detect abnormal electrocardiogram (ECG) beats from the MIT-BIH arrhythmia database [1]. The ECG signals are classified into two different classes - normal and abnormal - based on the statistical analysis of the features extracted by discrete wavelet transform (DWT). The models we utilize are Support Vector Machine (SVM), Feedforward Neural Network (FNN) and Elman Recurrent Neural Network (Elman RNN). In this paper, we explain the frameworks of these three models and test their performance on our DWT features.

***Index Terms*—** ECG Abnormity, Discrete Wavelet Transform, Support Vector Machine, Feedforward Neural Network, Elman Recurrent Neural Network

2. INTRODUCTION

Electrocardiogram (ECG) records the electrical activity of the heart which shows the regular contraction and relaxation of heart muscle. The information provided by electrocardiography is significantly valuable to assess the functionalities of the heart and cardiovascular system. The patient can receive appropriate treatment if heart abnormalities can be detected in the early stage. Therefore, numerous works and methods are proposed to detect heart diseases [2, 3]. Here we adopt three models in our paper as well as a discrete wavelet transform method to decompose the ECG signals from MIT-BIT arrhythmia database containing 48 files each of which records 30 minutes of ECG signals. The process can be generally divided into three procedures: 1) beat detection; 2) feature extraction; 3) classification.

The wavelet transform (WT) has been widely applied to many signal processing tasks [2, 3]. Since it can manipulate data in compressed parameters named features instead of the huge volume of the raw data, we can represent the ECG signals as a few parameters. Therefore, WT can significantly enhance the efficiency of the following classification step.

The first classification model is support vector machines (SVM) [4]. The second approach is Feedforward Neural Network (FNN) [5]. And Elman Recurrent Neural Networks (Elman RNN) [6] is the third classification approach adopted in our paper.

The result shows we are able to detect heart beat abnormalities with good accuracy.

The following paper is organized as follows. In section 3, we discuss the approach of the discrete wavelet transform to represent the ECG signals. From section 4 - 6, three adopted models are introduced. In section 7 and 8 , the experiment results are presented and the conclusion is drawn.

3. FEATURE EXTRACTION USING DISCRETE WAVELET TRANSFORM

Given that ECG signals can be viewed as an overlap of multiple structures occurring on different time scales at different time spots, we adopt wavelet analysis to isolate these structures of different time scales because of its ability to process non-stationary signals. According to the work of Güler *et al.* [7], the number of decomposition levels is set as 4. Therefore, the original ECG signals are decomposed into four subbands D_1-D_4, and one approximation A_4. And the wavelet coefficients are computed using the Daubechies wavelet of order 2. The wavelet coefficients were calculated by a Python package called PyDev [8].

After the features are calculated, the following statistic analysis are performed on all subbands D_1-D_4 and A_4 [7]:
1. Mean of the absolute values of the coefficients in each subband;
2. Average power of the wavelet coefficients in each subband;
3. Standard deviation of the coefficients in each subband;
4. Ratio of the absolute mean values of adjacent subbands.

All these statistical features for D_1-D_4 and A_4 serve as input to all the following three classifiers.

4. CLASSIFICATION OF SVM

4.1. Support Vector Machine

The SVM is designed for binary classification for linear division of feature space but can be extended to multiple class using methods like one-against-one, one-against-all and fuzzy decision and non-linear separation using kernel trick to mapping inputs to high-dimensional feature space. The SVM classifier is defined as following:

$$f(x) = sgn(w^T \phi(x) + b) \tag{1}$$

where $\phi(x)$ is the kernel function and w^T is the vector of feature space, and b is a scalar. sgn(x) is the decision function separate the results into -1 and +1. Thus creating linearly hyperplanes. The optimal one can be calculated as follows:

$$minJ(w, b, \xi_i) = \frac{1}{2}w^T w + \gamma\frac{1}{2}\sum_{i=1}^{n}\xi_i \tag{2}$$

subject to:

$$y_i(minJ(w, b) = \frac{1}{2}w^T w) \geq 1 - \xi_i \tag{3}$$

$$\xi_i \geq 0 \tag{4}$$

parameter ξ_i is slack variables if the training data are nonlinearly separable and γ is the tradeoff between maximum margin and the minimum classification error. The optimization problem of (2) is a convex quadratic program. The quadratic programming problem of soft margin SVM can be expressed in the dual form as follows:

$$max\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}y_i y_j(\phi(x_i)^T\phi(x_j))\alpha_i\alpha_j \tag{5}$$

subject to:

$$0 \leq \alpha_i \leq \gamma \tag{6}$$

$$\sum_{i=1}^{n} y_i\alpha_i = 0 \tag{7}$$

By solving the equation above, we can find the α_i coefficients. And by substituting these coefficients to the following equation we can obtain the parameters of the classifier:

$$w = \sum_{i=1}^{n}\alpha_i y_i \phi(x_i) \tag{8}$$

4.2. Sequential Minimal Optimization

To solve the quadratic programming problem, Sequential minimal optimization (SMO) method is introduced in this paper. SMO is an iterative algorithm that breaks this problem into a series of smallest possible sub-problems. For the SVM constraints the smallest possible problem involves two Lagrange multipliers as follows:

$$0 \leq \alpha_1, \alpha_2 \leq \gamma \tag{9}$$

$$y_1\alpha_1 + y_2\alpha_2 = 0 \tag{10}$$

The algorithm proceeds as follows:
1. Find a Lagrange multiplier that satisfies the Karush-Kuhn-Tucher (KKT) conditions.
2. Pick a second multiplier and optimize the pair .
3. Repeat steps 1 and 2 until convergence.
When all Lagrange multiplier found, the problem has been solved.

5. CLASSIFICATION USING FEEDFORWARD NEURAL NETWORK

Backpropagation (BP) Neural Network [9] is a multi-layer feedforward network trained by error backpropagation algorithm. Due to its good generalization performance against many machine learning problems, it has been popular in pattern classification and other related areas.

5.1. The Architecture of Feedforward Neural Network

A regular BP neural network [10] has an input layer, an output layer, and one or more hidden layers. Each layer contains one or more neurons and there are synaptic connections between every two adjacent layers. Figure 1 shows the structure of a neuron and the architecture of a six-layer neural network.

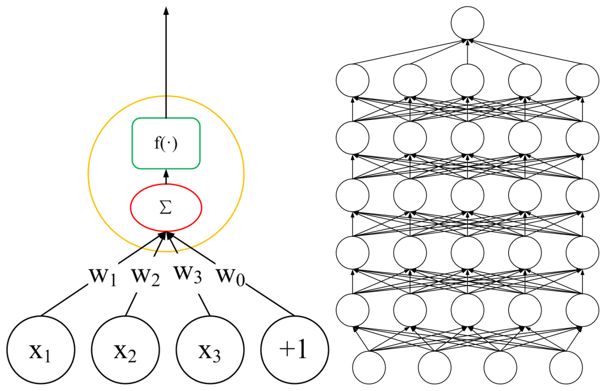

Fig. 1. Left: the structure of a neuron; right: the architecture of a six-layer neural network

A neuron is a nonlinear system with multiple inputs and a single output. Its output is given by $y = f(\sum_{i=0}^{n} w_i x_i) = f(\mathbf{w^T x})$, where $x_0 = 1$ and $f(\cdot)$ is a nonlinear function called activation function. Sigmoid function is typically used in classification task.

$$f(z) = \frac{1}{1 + e^{-\beta z}} \tag{11}$$

Assume the network has m hidden layers and $n_1, n_2, \ldots, n_m$ within each hidden layer. The input pattern $\mathbf{x}$ is a p-dimensional vector. Its synaptic weights and activation functions are respectively $\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_{m+1}$ and $f_1, f_2, \ldots, f_{m+1}$. The inputs of hidden layers are $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{m+1}$. The real output $\mathbf{y}$ and the desired output $\mathbf{d}$ are q-dimensional vectors. Based on above denotation, the outputs of each layer are:

$$
\begin{cases}
x_1(j) = f_1(\sum_{i=0}^{p} W_1(i,j)x(i)), & j = 1,2,\ldots,p \\
x_2(j) = f_2(\sum_{i=0}^{n_1} W_2(i,j)x_1(i)), & j = 1,2,\ldots,n_1 \\
\cdots \\
x_m(j) = f_m(\sum_{i=0}^{n_{m-1}} W_m(i,j)x_{m-1}(i)), & j = 1,2,\ldots,n_{m-1} \\
y(j) = f_{m+1}(\sum_{i=0}^{n_m} W_{m+1}(i,j)x_m(i)), & j = 1,2,\ldots,n_m
\end{cases}
\tag{12}
$$

5.2. Error Backpropagation Algorithm

The object of error backpropagation algorithm is to minimize the cost function:

$$J(\mathbf{W},\theta) = \frac{1}{2}\sum_{i=1}^{q}(d(i) - y(i))^2 \tag{13}$$

According to the gradiend descent method and the chain rule, weight update for the output layer is as follows:

$$\Delta W_{m+1}(i,j) = \eta\delta_{m+1}(j)x_m(i) \tag{14}$$

where

$$\delta_{m+1}(j) = (d(j) - y(j))f'_{m+1} \tag{15}$$

We can easily obtain the weight update for other layers following the same rules. As we can see, the order of weight update computation is from higher layers through lower layers. That is why the learning algorithm is called error backpropagation [10].

6. CLASSIFICATION USING ELMAN RECURRENT NEURAL NETWORKS (ELMAN RNNS)

There are many forms of Recurrent Neural Networks (RNNs). One of the simplest ones is the Elman RNN [11].

6.1. The Framework of Elman RNN

The difference between Elman RNNs and Feedforward Neural Networks is that in a Elman RNN, the outputs of a hidden layer are copied to a special layer called context layer. Then during the next forward process, the values stored in the context layer are used as additional inputs to the hidden layer [11].

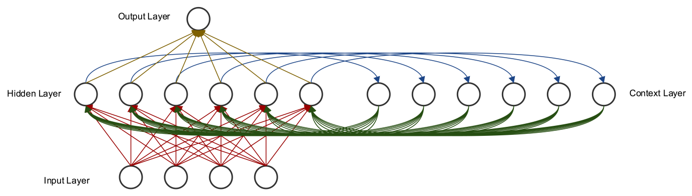

Fig. 2. Framework of a classical Elman RNN.

6.2. The Weights Update Method

The weights update method for Elman RNNs [5] is quite similar to FNNs. After the error is back-propagated to the hidden layer, it is used to update both the weights between hidden layer and input layer and the weights between hidden layer and context layer.

In addition to stochastic gradient descent, there are many other weights update methods. A prior work on ECG classification [6] used LevenbergMarquardt algorithm to train a Elman RNN and yielded a good classification result.

7. EXPERIMENTAL RESULTS

7.1. Results of SVM

We apply the SVM based on the polynomial kernel function. There are 187200 instances for training data set and each instance has 38 features. Then we use 5-fold cross validation to choose both the best cost C and γ parameters in the kernel function. First randomly choose 50% of the training set as the cross validation set. Then divide the whole set into 5 subsets of equal size. Each subset is then used as the test set in turn with the remaining 4 subsets as the training sets of the classifier. We select C and in the exponential sequence $C = 2^{-4}, 2^{-3}..., 2^2$ with γ being the default value 2^0. The results are shown in the first three columns of table 1. It is found that the performance is optimal when C is 2^{-2}. Then we fix C parameter as 2^{-2} and find the optimal γ parameter in this model. The results are shown in the last three columns of table 1.

Table 1. Cross validation accuracy with C and γ parameter

C	γ	accuracy(%)	C	γ	accuracy(%)
2^{-4}	2^0	62.15	2^{-2}	2^{-4}	23.41
2^{-3}	2^0	66.20	2^{-2}	2^{-3}	49.77
2^{-2}	2^0	**67.31**	2^{-2}	2^{-2}	**76.97**
2^{-1}	2^0	67.13	2^{-2}	2^{-1}	70.58
2^0	2^0	66.77	2^{-2}	2^0	66.77
2^1	2^0	66.49	2^{-2}	2^1	66.49
2^2	2^0	66.33	2^{-2}	2^2	62.29

Then we use the optimal parameters found above to test the accuracy in the test set. The test set has 62400 instances and the accuracy of the model is 66.67%.

7.2. Results of FNN

We use a four-layer BP neural network including two hidden layers with 16 neurons in each hidden layer to handle this classification task. The inputs of the model are 38-dimensional feature vectors and the output layer has one single neuron. The classification threshold of the output neuron is 0.5, which means the input pattern is classified to Class "N" (Normal) if $y > 0.5$ and Class "A" (Abnormal) otherwise. We use sigmoid function with $\beta = 1$ as the activation function for all neurons. The whole model was implemented in C++ and no machine learning toolkit was used.

To make sure the convergence of the training process, a self-adaptive learning rate is set up. The initial value of learning rate η is 0.0005, and we reset $\eta = \eta \times 0.8$ if the value of the cost function increases in the process.

The classification accuracy on training set with different epochs is shown in figure 3.

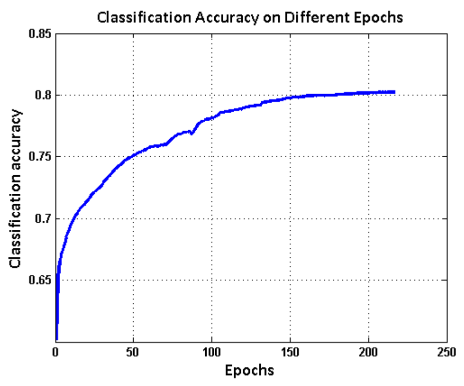

Fig. 3. The classification accuracy on training set with different epochs

The classification accuracy is 83.64% (after 2000 epochs) on training set and on test set is 69.45%.

7.3. Results of Elman RNN

The experimental settings of Elman RNN are similar to the FNN. There are four layers in the network. Input layer is consisted of 38 nodes. The numbers of nodes in the hidden layer and the context layer are both 32. There is only one node in the output layer. The classification threshold of the output neuron is also 0.5 and the same sigmoid function is used. The whole model was implemented in Python and no machine learning toolkit was used.

An adaptive training method is also used in the training process of the Elman RNN. The initial learning rate is 0.02. For each epoch, the training accuracies of both current state and the previous state are stored. If the accuracy of current state is lower than the previous state, the training process will roll back to the previous state and the learning rate will be divided by 2.

The training process will stop if the accuracy improvement between current state and the previous state is not larger than 0.001% or the learning rate becomes smaller than 10^{-20}.

The accuracies at different epochs are shown in figure 4.

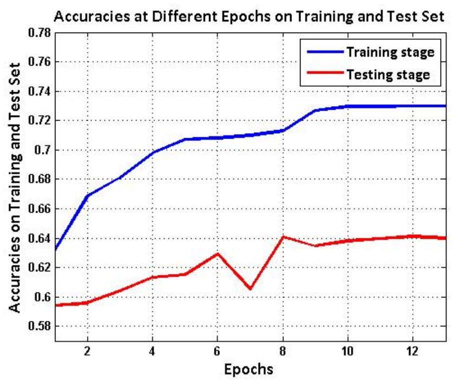

Fig. 4. The accuracies at different epochs on training set.

The figures show that with the increase of epochs, the performance on both the training set and the test set increase. The final accuracy is 72.98% on the training set and 63.98% on test set. Multiple experiments with the same settings are conducted and we get similar results.

One thing worths noticing is that as the accuracies on the training set increases, the accuracies on the test set may not increase correspondingly. This phenomenon is common because the accuracies on the training set and the test set are usually not linearly correlated.

8. CONCLUSION

In this paper, we proposed three different models for classifying ECG signal segmentations: SVM, FNN, Elman RNN. The experimental results in figure 5 show that SVM and FNN have higher classification accuracy on test set than Elman RNN. In addition, all three models outperform the baseline model. However, further research on the models is demanded since the classification performance can be impacted by model configuration, such as parameter setting, selection of activation function, etc.

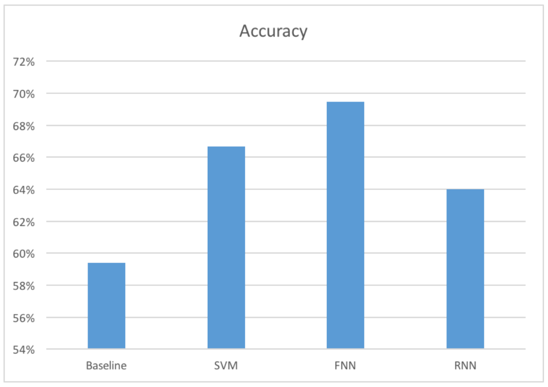

Fig. 5. Results Comparison Among Three Models and Baseline Model on test set

9. REFERENCES

[1] "The mit-bih arrhythmia database," `http://physionet.ph.biu.ac.il/physiobank/database/mitdb`, Accessed: 10-30-2015.

[2] SC Saxena, Vinod Kumar, and ST Hamde, "Feature extraction from ecg signals using wavelet transforms for disease diagnostics," *International Journal of Systems Science*, vol. 33, no. 13, pp. 1073–1085, 2002.

[3] Paul Addison, James N Watson, Gareth R Clegg, Michael Holzer, Fritz Sterz, and Colin E Robertson, "Evaluating arrhythmias in ecg signals using wavelet transforms.," *IEEE Engineering in Medicine and Biology Magazine*, vol. 19, no. 5, pp. 104–109, 2000.

[4] Narendra Kohli, Nishchal K Verma, and Abhishek Roy, "Svm based methods for arrhythmia classification in ecg," in *Computer and Communication Technology (ICCCT), 2010 International Conference on*. IEEE, 2010, pp. 486–490.

[5] Mikael Boden, "A guide to recurrent neural networks and backpropagation," 2001.

[6] Elif Derya Übeyli, "Combining recurrent neural networks with eigenvector methods for classification of ecg beats," *Digital Signal Processing*, vol. 19, no. 2, pp. 320–329, 2009.

[7] İnan Güler and Elif Derya Übeylı, "Ecg beat classifier designed by combined neural network model," *Pattern recognition*, vol. 38, no. 2, pp. 199–208, 2005.

[8] "Pydev," `http://www.pydev.org`, Accessed: 10-30-2015.

[9] Robert Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural Networks, 1989. IJCNN., International Joint Conference on*. IEEE, 1989, pp. 593–605.

[10] Richard O Duda, Peter E Hart, and David G Stork, *Pattern classification*, John Wiley & Sons, 2012.

[11] Jeffrey L Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.