EKG Waves Classification

Luis Oliveros Colón
Artificial Intelligence for ECE
Stevens Institute of Technology
NJ, USA
loliveros@stevens.edu
luis.oliveros.colon@gmail.com

***Abstract*—EKG's are an important test on daily medical examination, giving an accurate and crucial information about the heart condition of the patient. The analysis and interpretation of this waves required of a professional with a higher education, this specialist staff may not accessible to all population, either by shortage of this staff or because this population live in remote zones of the planet.**

In this project, it is proposed an alternative way to give support to this specialized staff, to check and and analysis EKG waves to detect as soon as possible those cases where the patients have high chances to suffer from a heart disease.

***Index Terms*—EKG waves, Kn Nearest Neighbor, Neural Networks, Time Series**

I. Introduction

A. Objective

The main goal of this project is the classification of EKG or ECG waves (the heartbeat waves measure by a monitor), each EKG corresponds to a patient that may have or may not have a heart disease, for this purpose it has made used of different algorithms to test their accuracy in this type of data. Both training data and test data can be classified into four different classes:

- N: Normal beat
- S: Supraventricular premature beat
- V: Premature ventricular contraction beat

B. Data

Each sample corresponds to a full EKG wave, represented by a vector, each vector has 188 points, the first 187 represent the waveshape of the heartbeat, while the last element in the vector represents the label given to that waveshape.

- N: 0
- S: 1
- V: 2

C. Methods

For this project, it has been decided to test the performance of two algorithms using time series as samples. In this work it has been used Kn nearest neighbor and NN and these two algorithms will be compared with a third algorithm (LSTM algorithm) to test how good the performance is, this last algorithm has proved a high efficiency on time series.

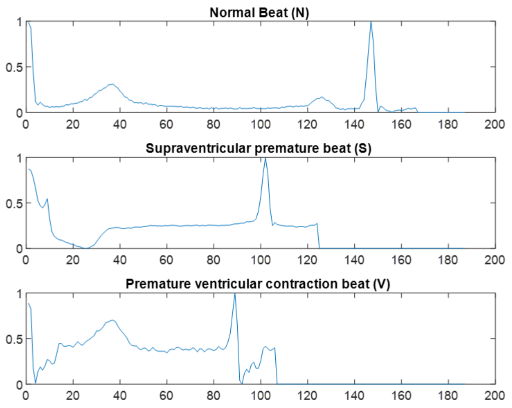
II. Implementation

In order to perform this classification, it has been decided to explore the use different tools to see the performing of each of them, the tools that have been used for this purpose are:

- **Kn - Nearest – Neighbor Estimation**
- **Feature extraction and Neural Network**
- **LSTM algorithm**

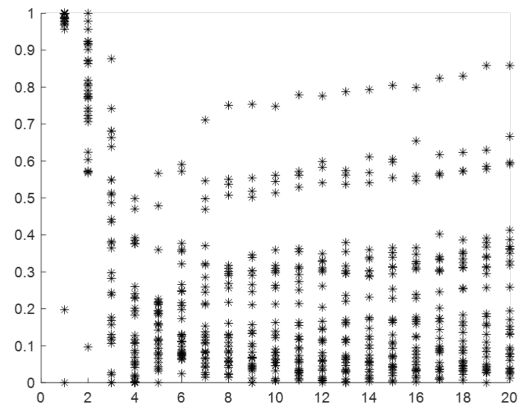
A. Kn - Nearest – Neighbor Estimation

As we can see in the figure below, for each kind of type of EKG wave, we can appreciate slightly differences between them.



The idea behind the used of this method is to consider that each class has 187 probability density functions (one density function for each of 187 elements that represent each EKG wave).

So, let's say that we take all training samples that belong to class N, for each of the 187 points there will be a concentration of values around one specific value of the y-axis. The figure below shows 30 training samples that belong to class N, and only it has been represented the first 20 points out of the 187 for better understanding.



So, for each point of the x-axis there is going to be 30 points, and these points hopefully would be concentrating around one specific area, for example in point 1 of the x-axis, most of the values are around the value 1 of the y-axis, this concentration can be defined with a distribution density function:

$$p_n(x) = \frac{k_n/n}{v_n} \tag{1}$$

Where kn represents the k nearest samples closest to our sample value x, n represent the total number of samples at the point in x-axis (in this case 30) and V represents the maximum distance between the value of x and kn samples. So for the x-axis point 1, most of the values are around 1, if our test sample, its first value it is also around 1, the probability for that first sample and its first point to be class N will be really high, if we apply this same concept to the 187 points that a sample has, we can estimate the probability of the test sample to be class N.

This whole thing can be applied to estimate the a posteriori probabilities P(wi—x) when we have different classes. And the a posteriori probability is defined as follows:

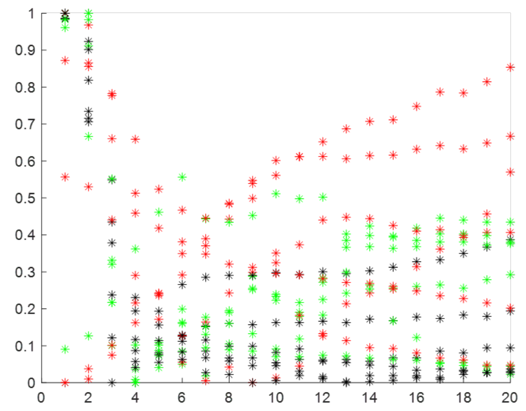
$$P(w_i|x) = \frac{p(x, w_i)}{\sum_{j=1}^{c} p(x, w_i)} = \frac{k_i}{k} \tag{2}$$

Where k represents the number of samples inside our window and it is defined by the following expression:

$$k = \sqrt{k} \tag{3}$$

And ki represents the number of samples inside the window that belong to class i.

The next figure represents the values that 6 training samples of each class for the first 20 points of the waveshape. We can appreciate that there exist parts where one class is more dominant that the others for certain values.



Now the procedure to classify a test sample is, when a sample is selected to be classified, we will analyze each of the 187 points that form this sample, we will take point by point and we will measure how likely is each point to be class N, S or V based on the values obtained on equation 2. Once we have all the probabilities for the 187 points, we can decide what class the test sample belongs.

The code will compute a matrix called Prob matrix of 3x187 (i=number of classes x j=points per sample) for each of the test sample. The element of this matrix represent the probability of the point j to belong to the class i. Once we have all this information we can calculate the Bayesian risk of choosing one class or another (error matrix), this matrix we will give us the risk that entails choosing class i for the point j. Computing the overall risk for all points, total error, we can decide which class we are assigning to that test sample based in the lowest error obtain in the vector, total error.

a) Results: After running the program and comparing the computed results obtained with the real results from the test data, the program reached the following confussion matrix:

$$\begin{array}{ccc} 435 & 162 & 205 \\ 110 & 338 & 181 \\ 74 & 56 & 1062 \end{array}$$

Where the rows correspond to the action taken (row1=classify in class1, row2 = classify in class2, row3=classify in class3) and the columns to the true nature of the state (column1 = sample belongs to class1, column2 = sample belongs to class2, column3 = sample belongs to class3).

$$eff = 0.70$$

While testing the program some problems were found and this is the way I addressed. Sometimes the distance between one point of the test sample and the corresponding training samples for that point will be zero, that is not a problem as long as the number of distances equal to zero are less than k. But, if the number of distances equal to zero are bigger than k, then the procedure that we are given it is not reliable anymore. Then we must change the way to compute the probability for that point in that sample. The way to address this problem is to use as n the total number of distances equal to zero instead of the expression on equation 3. This way the probability would be compute as, the number of distances equal to zero for class i divided by the total number of distances equal to zero of all classes.

B. Feature Extraction & Neural Networks

The way this method is addressed is by getting to know the true nature of the data we are working with and what it is the real reason why the waves are different. Every normal heartbeat has the following general shape wave:

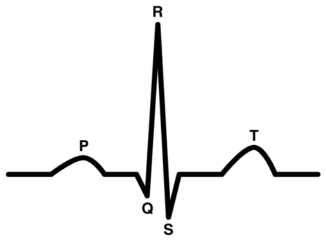


Fig. 1: EKG wave shape

We should be able to find what makes different the waves for each of the diseases that we are studying. According sources on the internet [1], what makes a Supraventricular premature beat and Premature ventricular contraction different are the following characteristics in their peaks P, QRS and T.

• Supraventricular premature beats: the wave P seems to disappear, and T wave gets closer to the wave QRS.

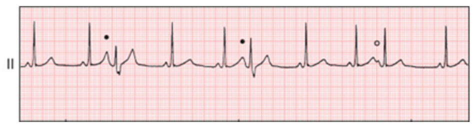


Fig. 2: Supraventricular premature beats wave shape

• Premature ventricular contraction: In this case it looks like both P and T are completely flat while the peak S seems to have increased considerably.

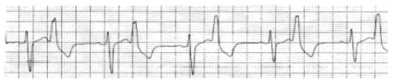


Fig. 3: Premature ventricular contraction wave shape

It is not the intention to make a deep analysis in what are the main differences between these waves but to make a point in the fact that the main differences seem to be related with the peaks of these waves and where these peaks are located. Therefore, by extracting information and features related with the peaks could led us to some determinant features that will help the neural network to classify each of the EKG waves with its corresponding class.

The code for this method has been divided into 3 different parts.

1) Feature Extraction: In this first part of the code we will extract the features of each sample to use in the Neural Network. Because the difference lays on the peaks and when those peaks are placed, we will put especial attention on them, as mentioned above, the normal beats are supposed to have three peaks, and for the others there should be some sort of irregularities on two of those peaks.

The following features are the chosen to use in the input units of our neural network:

- Mean
- Length
- Value of the highest peak / mean
- Value of the second highest peak / mean
- Value of the third highest peak / mean
- (Position peak 1 – position peak 2) / length
- (Position peak 1 – position peak 3) / length
- (Position peak 2 – position peak 3) / length
- Value of the highest peak * Value of the second highest peak
- Value of the highest peak * Value of the third highest peak
- Value of the second highest peak * Value of the third highest peak
- Value of the highest peak * Value of the highest peak
- Value of the second highest peak * Value of the second highest peak
- Value of the third highest peak * Value of the third highest peak

We can change or keep adding more features into the program. This first code (matrix features train test data.m) will generate two matrix that will be saved in the folder that we are working on. The first, is called Matrix Feature Training Data, where the rows correspond to each sample of the training data and the columns correspond to the value of each feature listed above from each of the samples. The second matrix is called Matrix Feature Test Data, same thing for the test samples.

2) Computation of the Neural Network Weights: Once we have saved the matrix in our workspace, we can run the second code. In this second part we are to define a neural network that follows the stochastic backpropagation algorithm and a second one that follows batch backpropagation algorithm in order to see what results yielded each of them.

We will be calculating, for both algorithms, the weights that link the input – hidden units and the hidden – output units. These are the main parameters used for this NN:

- Number of hidden units, nH = 100. (Following the rule: number of weights = (number training samples) / 10)
- Number of input units, d = 8. (This parameter will be defined by the first part of the code, the number of chosen features, bias not included in this number)
- Number of output units or classes, c = 3.
- Activation function: f(net)=a*tanh(b*net)
- a = 1.716
- b = 2/3
- Theta = 0.1

- Learning rate eta = 0.001 (This value has been determined experimentally)

Class	Target Vector
N: Normal Beat	[1,-1,-1]
S: Supraventricular Premature	[-1,1,-1]
V: Premature Ventricular	[-1,-1,1]

The criterion function used for these algorithms is:

$$J = \frac{1}{n}\sum_{p=1}^{n} J_p \tag{4}$$

Where p represents sample-p and Jp:

$$J_p(W) = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2 \tag{5}$$

Equation number four it is going to be the one that we will use for stopping the program for the learning process, once the value of this equation reaches under the value assigned to theta, the program will stop.

a) Stochastic Backpropagation Algorithm: At first place, theta was set up to 0.1, but the system was not able to reach an error rate under 1.1 as the figure below shows:

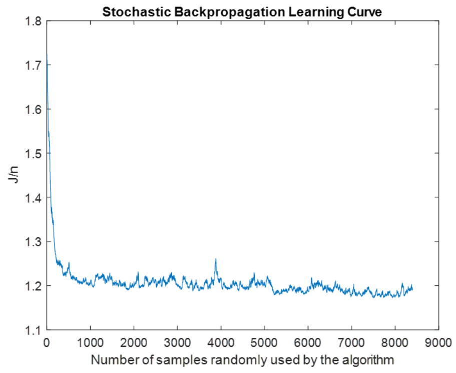


Fig. 4: Stochastic Backpropagation Learning Curve

Since picking a low theta does not work, other criteria can be chosen for stopping the system from the training stage. For that purpose, we can extract a data set from the training data and use it as validation data set. By comparing both curves (training and validation) we can decide when it is appropriate to stop the learning process.

After 10 hours of running the program, this was the result. As we can see, the error from the validation data differs only slightly from the error corresponding to the training data. As a matter of fact, both curves seem to have very similar values throughout the whole process. So it was decided to stop the program and kept the weights that were computed on the last iteration.

As a curiosity, it has been plot at the same time a graph where it is compared the training, validation and test curves, to see how the error are in three sets of data.

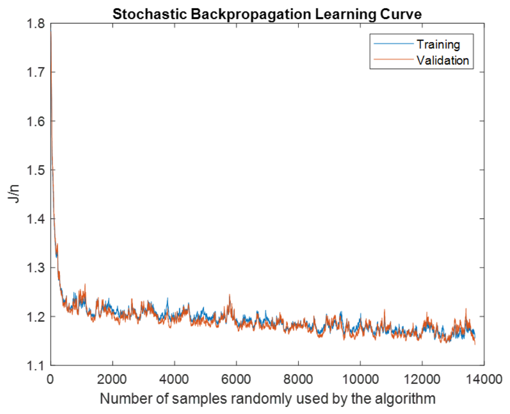


Fig. 5: Stochastic Backpropagation Learning Training/Validation Curves

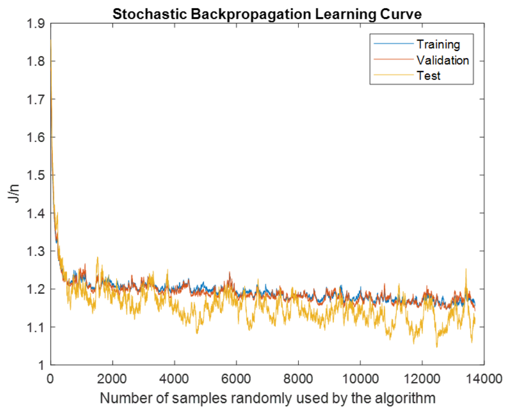


Fig. 6: Stochastic Backpropagation Training/Validation/Test Curves

Our data set consists of a list of 7049 two-dimensional 8-bit graylevel training images with their corresponding (x, y) coordinates of the 15 facial keypoints. Each input image is represented by a size of 96 × 96 pixel, with pixel values in range of [0, 255]. The given training set is a huge matrix of size 7049 × 31, where each row corresponds to one image, the first 30 target columns give the (x, y) values for each of the 15 facial keypoints, and each entry in the last column is a long list of 9216 numbers representing the pixel matrix of each image melted row by row.

b) Batch Backpropagation Algorithm: For this algorithm the error rate does not get any better, as a matter of fact it is worse. We can appreciate how for every epoch the oscillation in the error are bigger than in the previous algorithm (notice that the learning rate is the same for both), as we could expected.

Since this configuration (number of hidden units, features chosen for this purpose, normalizing data before using it for the learning stage…) cannot give better results on the error rate it is decided to use the weights obtained in the Stochastic Backpropagation Algorithm because the error rate obtained

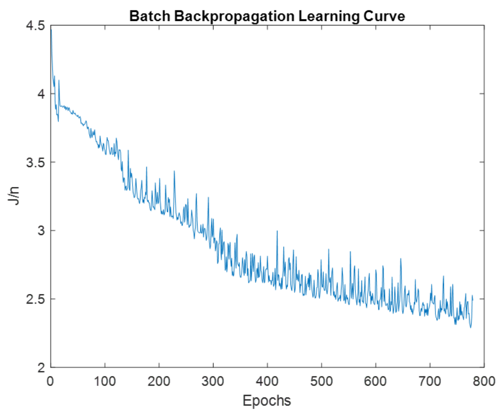


Fig. 7: Batch Backpropagation Learning Curves

was lower.

3) Classifying the Test Samples: Once we have run the second part of the code, we will have defined the Neural Network, the weights between the input and hidden units and between the hidden and output units, that is going to process the test data.

Using the Feedforward operation, we will feed the NN with the test samples and it will give their corresponding expected result for each sample. It is very unlikely that the given result by the neural network matches with the target vector, therefore it is necessary to make a decision based on the result given by the NN and how closed it is this result to a target value.

From the neural network we know that the three output units will give a value between 1.716 and -1.716, given these results we need to create a classifier that will categorize each test sample. For that purpose, the following matrix will be created for each test sample:

$$\begin{pmatrix} 1-z_1 & -1-z_1 \\ 1-z_2 & -1-z_2 \\ 1-z_3 & -1-z_3 \end{pmatrix}$$

Each element of this matrix represents the difference between an output unit result and one of the possible values in the target vector. From table 1, we can appreciate how every class has one value equal to 1 and two values equal to -1. To determine which class a sample belongs to, the program will measure the following quantities:

$$\begin{aligned} d_1 &= \sqrt{(1-z_1)^2 + (-1-z_2)^2 + (-1-z_3)^2} \\ d_2 &= \sqrt{(-1-z_1)^2 + (1-z_2)^2 + (-1-z_3)^2} \\ d_3 &= \sqrt{(-1-z_1)^2 + (-1-z_2)^2 + (1-z_3)^2} \end{aligned} \quad (6)$$

Out of the three quantities, the program will pick the lower one, which we will basically calculating for which target vector the error is less from the result obtained in the Neural Network. If the lower one is d1 then the sample will belong to class 1, d2 for class 2 and d3 to class 3.

4) Results: Once the Neural Network has classified all the test data, it will compare the computed class with the real class, the results obtained were not good as expected:

$$eff = 0.57$$

The confusion matrix:

541	111	89
370	265	400
208	180	959

It has been tried to increase the efficiency of the Neural Network by modifying the parameters with not success, the learning rate has been decided to have 0.001 since with this values the error rate achieved lower values, 0.1 seemed to be quite higher than the optimal learning rate because the error rate only increased with every iteration.

C. LSTM Algorithm RNN

As a last method for classifying the EKG waves, we are going to make used of this method available in Matlab, the main purpose for using this developed method is to compare the results with the previous ones, so we can make a objective comparation since the data is the same in the three methods.

These are the results:

$$eff = 0.76$$

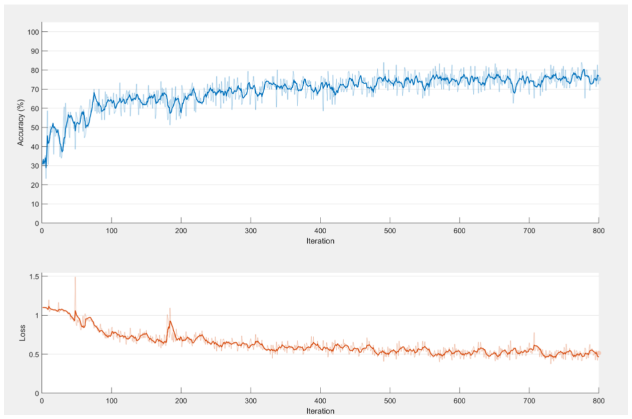


Fig. 8: LSTM Accuracy & Curves

III. CONCLUSION

As possible modifications that could be carried on improving the algorithms, it would be interesting to work on the quality of the data, executing some labors of cleaning and preparation before using it in the algorithms. Most of the samples have a certain amount of zero value elements after the last nonzero element value in the array. It is not enough by deleting them, it is also necessary, once the zero elements on the end of the array are erased, to make all the samples the same length to properly use this algorithm, so the implementation of a code to stretch the samples by adding points in between inside the sample array would be necessary.

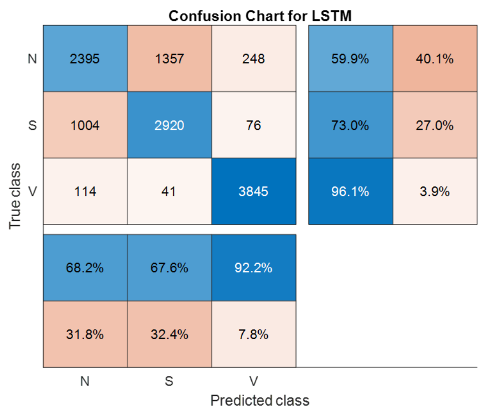


Fig. 9: LSTM Confusion Matrix

It will be interesting how this modification can make any improvements on the efficiency if it does.

As we can see the level of accuracy on the last algorithm used, it can be observed that is not far from the level of accuracy obtained in the first method, where the accuracy obtained was 70 %, what makes us think that by working on the data pre-processing can make some improvements because this last algorithm has proved its efficiency with some other data sets. Given these results, we could say that the first method used is good enough while the second could be modified in some ways (for example by changing some of the parameters) to increase the level of performance.

REFERENCES

[1] Richard O. Duda, Peter E. Hart, David G. Stock. Pattern Classification. Second Edition. 2001

[2] [1]: https://www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/premature-ventricular-contraction

[3] Highly comparative time-series analysis: the empirical structure of time series and their methods. Ben D. Fulcher, Max A. Little and Nick S. Jones