Engineering Journal of the University of Qatar, Vol. 15, 2002, pp. 63-79

APPLICATION OF NEURAL GENERATED ERROR SIGNAL IN CLASSIFICATION AND COMPRESSION OF ECG BEATS

Raed Abu-Zitar
Department of Computer Science
Al-Isra Private University
Amman, Jordan
Email: rzitar@isra.edu.jo

ABSTRACT

This paper presents a neural network architecture connected in cascade for ECG Holter system beat classification and compression. The system works in real time, on line, and is capable of recognizing and compressing up to 40 artificial and real QRS templates. The parallelism of neural network applied in this paper increases the efficiency of computations. An error signal derived from difference between predictor and testing signal is used in the classification and compression. This error signal is processed in two ways; firstly it is encoded with Hoffman sequence and saved as the compressed signal that may be used later for reconstructing the original ECG signal. Secondly the error signal is converted into primitive ternary signal and used in sharp and direct classification of the ECG beats. The proposed architecture consists of the following: a neural network used to generate linear predictions for signals. Another neural network generates error signals by calculating difference between predictions taken from first neural network and a testing signal. A third neural network does the classifications utilizing the error signals instead of complex raw signal. At first, Hermite functions are used in generating artificial ECG signals. Testing is done by adding noise to the original Hermite functions used in learning. The whole process is repeated for real ECG beats taken from MIT database. The results show clear success in classification besides additional success in compression.

KEY WORDS: Genetic Algorithm, ECG, Self Organizing Map.

1. INTRODUCTION

Adequate literature exists on ECG beat classification [1],[2],[3]. Most of these handle raw signals directly and use stages of unsupervised and supervised learning to discover the classes and then quantize them. For medical applications, it is

essential to formulate a real time on-line system that can classify in seconds and, at most, in minutes the class of the beat. Artificial neural networks are employed to exploit their natural ability in pattern-recognition tasks for successful classification of ECG beat [2],[3],[4],[5],[6]. An obvious approach to alleviate this problem is to use as much as possible training data to develop ECG classifier. Larger databases indirectly imply larger amounts of data to deal with, and the need for data compression and less complicated features arise. Even large databases are not enough to cover every ECG waveform for all potential patients. Moreover, it is practically impossible to make the classifier learn to correct errors during normal clinical use. Therefore, we believe that the classifier should be adaptable with different patients and prompt on retrievals. The neural networks are excellent example of adaptable systems that can learn different tasks specially if the learning algorithm used is efficient and the amount of training data is not excessive.

In this paper a method is proposed that uses only a primitive ternary signal consisting of 0's, 1's,or -1's. This simple representation is enough to reflect a distinct feature for every different ECG beat. In this manner large amount of data storage and processing time are saved. Moreover, the resulting classification needs a single stage of unsupervised learning without any need for extra stage of "edge sharpening" [2]. The original error signal is also used in reconstructing the original ECG beat with high degree of precision. In that sense, we have "milked" the output of our proposed system until the last drop utilizing the error signal in two important applications; classification and compression. The original error signal is considered as the compressed ECG beat signal, additional stages of traditional compression techniques can be used to reach minimal file size [7].

The architecture of the proposed methods consists of three neural networks; the first network is a simple neural network trained by the genetic algorithm (GA) using supervised learning to predict the signal directly. With the GA, it usually takes a short training phase to converge to minimal error state. Then another network connected in series with the first trained network is used to measure the difference between the original signal and the predicted signal. The output is treated in two ways; a copy is stored in a file as the compressed version of the data , another copy is made as a template consisting of 1,0, -1 ternary values and fed to a third neural network that does the clustering with unsupervised learning algorithm. Each time the unsupervised neural network is subjected to a new template, it either classifies it to an existing class or it considers it as distinguished new pattern and, consequently, adds new class. Eventually, the number of classes will reach its maximum possible value. Keeping in mind the hardware feasibility of neural networks and the fact that linear prediction coefficients can be kept as constants for different beats for the same patient, this method can be considered as efficient online technique working in real time to help in providing crucial medical

decisions. A lot of work has previously been done on linear predictions and patterns classification [4],[5], but cascading neural networks and using error ternary patterns help in data reduction, cuts processing effort, and increases noise immunity. In this paper, and as first step, we use Hermite functions to model ECG signals for designing and testing our classifier, however, real ECG signals are used in building the predictor part. As a final step, our system is tuned up again and tested with signals taken from standard arrhythmia database. Next, we briefly describe the architectures and algorithms used in designing our classifier before we go into the simulations of learning and retrievals on different signal types.

2. PRELIMINARIES

Genetic Algorithm

Genetic algorithms were first introduced by John Holland [8] in 1975. David Goldberg, one of the pioneers in this field, introduced GA in its standard form and applied it in real life applications. GA's are global search techniques that utilize the mechanics of real life genetics. They are combinations of deterministic and stochastic search tools that are hard to fool by local optima. The applications of these algorithm are very wide, especially in the fields of optimization and machine learning [9],[10]. The GA in its standard form consists of three basic operations; reproduction, crossover, and mutation. The algorithm works on a population of binary strings, each string represents a set of variables to be optimized. This population is initially randomly generated, then for each member of the population a fitness value is calculated. This fitness value is directly calculated from the objective function under scope. Reproduction works by stochastically selecting two members of the population. The probability of selection for each member is directly proportional to the fitness of that member. After selection comes crossover, which is a partial swapping of the selected pairs with randomly selected crossover point. The crossover results in new offspring that are then subjected to mutation. Mutation is a random alteration of single binary bit, from 0 to 1 or vice versa. Usually, probabilities of mutation are very low, while probabilities of crossover are around 0.5. Both of these operations could be adaptive or related in a way to the objective function [10]. The process continues until a whole new generation of strings is established. Of course, the old generation is removed and the average fitness of the new population is then detected. A decision whether o proceed in generations or to quit is made according to the average fitness of the current population.

Figure (1) shows an outline for the standard GA basic operations. In our case, the GA is used to train a feed forward neural network (NN) [11] to implement the ECG prediction process. This NN happened to have a single neuron. As a matter of fact, we could have used larger neural network to implement the predictor job, and that would have made learning much easier. But, with the GA, learning is fast and convergence takes only several thousand iterations (very low CPU time). In our model, the prediction factors are the weights of the neural network. Those weights are encoded and mapped into the binary string as our optimization variables. The objective function is the error between the predicted signal and the real signal[10]. In the third section, we explain the linear prediction method.

Self-Organization Map (SOM)

SOM originally proposed by Kohonen[12],[13] is an unsupervised learning technique that has the ability to learn from examples and extract statistical properties of the examples presented during training. The input itself is used to stimulate the SOM to classify it into classes according to embedded features within input vector itself. In a way, this feature map is analogous to the spatial organization of sensory processing areas in the brain. Let $m_i(t)$ be the weights of the *ith* neuron in SOM at time instant *t*. The weights of SOM are updated according to the following simple formula:

$$m_i(t+1)=m_i(t)+h_{ci}(t)(x(t)- m_i(t)) \quad (1)$$

$h_{ci}(t)$ is the so called *neighborhood kernel*, which determine the size of the neighborhood of the *i*th neuron within which are all neighboring neurons. Updates are in response to the present feature vector $x(t)$. Initially, the neighborhood is large. The size reduces as clustering converges, until no neighboring neurons get updated.

3. LINEAR PREDICTION METHOD (LPM)

Any ECG signal $s(i)$ can be approximated by another sequence $\hat{s}(i)$ determined by unique set of predictor coefficients and $P(i)$ sample signals. That is

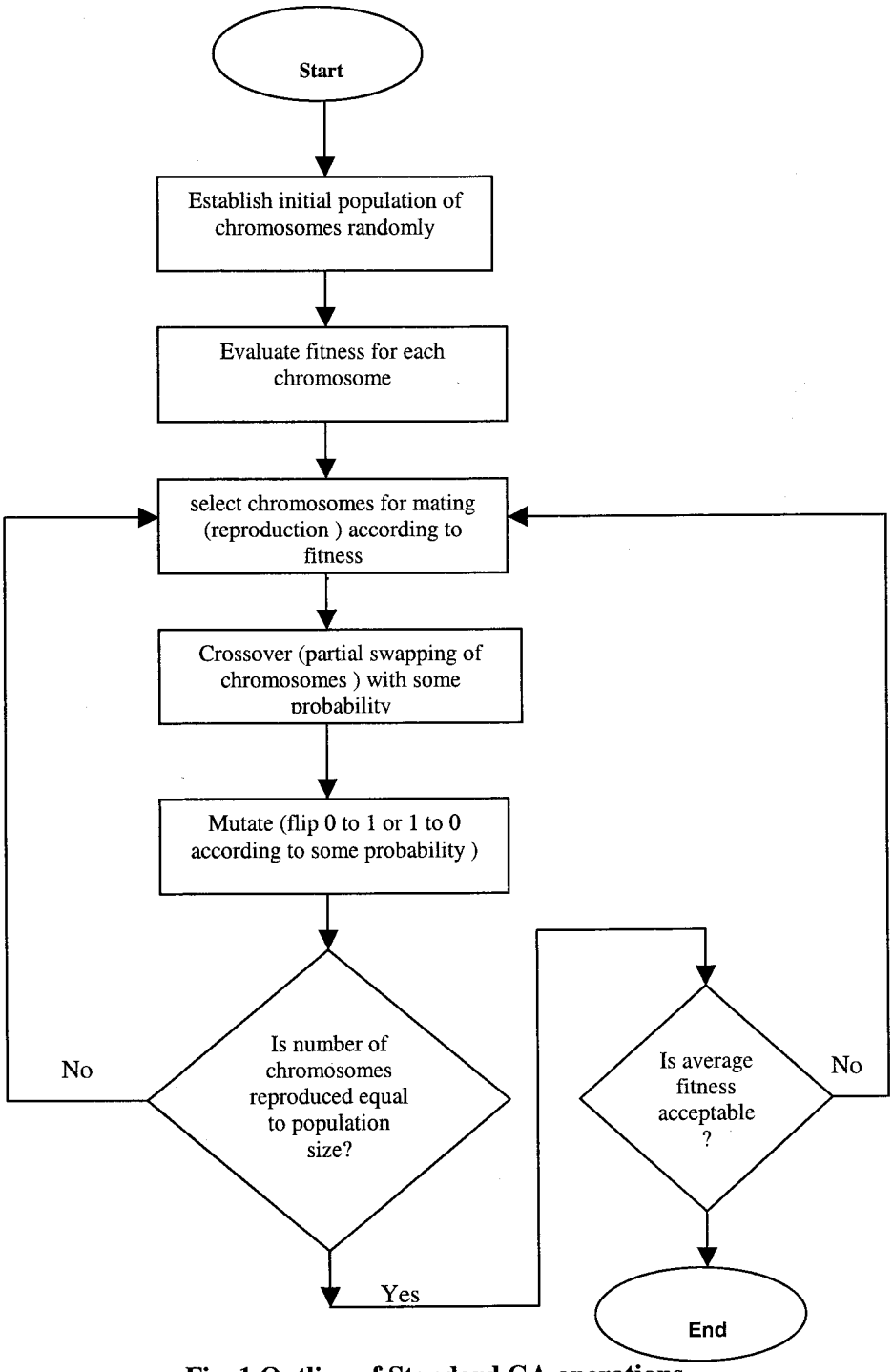

Fig. 1 Outline of Standard GA operations.

$$\hat{s}(i) = \sum_{k=1}^{P} a(k)s(i-k) \qquad (2)$$

where $a(k)$ is the k*th* linear predictive coefficient. Upper part of Figure(1) shows a two-layer neural network(NN) topology (input and hidden layers) for ECG linear predictions. We employed genetic algorithm (GA) to quickly solve for the coefficients. The GA is a massively parallel algorithm used for global optimization. The GA can quickly train the single neuron neural network as described in [9],minimizing the residual error signal $e(i)$ to almost zero. $e(i)$ is given by:

$$e(i) = \|s(i) - \hat{s}(i)\| \qquad (3)$$

We randomly picked 5 records from the AHA arrhythmia database and computed the coefficients for 7 normal beats. For every beat, we used 500 points for training taking current point and previous two points of the wave as inputs and the predicted next point on the wave as the output. The values of weights for the NN are the values of the linear predictive coefficients. First we had to verify whether LPM coefficients had to be individually computed for each ECG beat. We examined the coefficients for P = 2, 3, 4, and 5. The RMS error between the actual signal and the constructed signal is not much better for $P >> 2$. This is reported by others[4],[1].

Observing the values of the coefficients a_1 and a_2 for the case P=2, its obvious that there is great similarity between the values of coefficients for different beats for the same patient. Therefore, it is enough to keep these coefficients constant at some average value during beat classification for every patient. We couldn't really prove that these coefficients are also similar for different patients as mentioned in [6].

In our simulations, we picked P=3 in generating predicted signals (i.e. we had a_1, a_2, and a_3) in order to have the most minimal residual error. During training of the predictor NN, a reference model of 250 points for every beat is used, and training is repeated until residual error is minimized to a preset measure. After $e(i)$ is minimized, weights are kept constant. Now predictor NN is capable of generating $\hat{s}(i)$ given $\hat{s}(i-1)$, $\hat{s}(i-2)$, and $\hat{s}(i-3)$. The process is repeated for every record until the list of LPM coefficients is generated. In our predictor NN we used a three-weights (three coefficients) single neuron NN to build the predictor. The average of values for each of three coefficients a_1,a_2, and a_3 for a single patient are used in building a constant reference model for a normal beat for that patient. The aim here is to build some reference model for normal signals that can create a

minimal error signal. The output itself for the reference model is not our first concern. Table(1) shows the values of coefficients for 7 beats of three randomly selected different records for one patient.

Table 1 First three LPM coefficients for 7 beats from 3 different ECG records.

Record	V78			N12			N11		
Beat	a_1	a_2	a_3	a_1	a_2	a_3	a_1	a_2	a_3
	- 2.66	2.54	-0.80	-1.99	1.26	-0.20	-2.40	1.97	-0.54
2	-2.23	1.69	-0.41	-2.57	2.57	-0.87	-2.56	2.20	-0.64
3	-2.04	1.40	-0.74	-2.58	2.37	-0.77	-2.28	1.66	-0.36
4	-2.19	1.71	-0.49	-2.66	2.5	-0.86	-2.37	2.02	-0.56
5	-2.27	1.84	-0.56	-2.41	2.05	-0.60	-2.67	2.34	-0.67
6	-2.74	1.24	-0.23	-2.29	1.83	-0.50	-1.34	0.01	0.24
7	-2.23	1.84	-0.61	-2.48	2.18	-0.67	-1.29	0.09	0.28

4. The Cascaded Neural Network Design

The output of the predictor NN is fed to the second NN that generates the error signal $e_g(i)$ given by:

$$e_g(i) = W1s(i) + W2\, \hat{s}(i) \qquad (4)$$

where W1=1 and W2=-1. This NN subtracts the second input from the first input generating the error signal in two forms. The first form is the raw one that is saved in a separate file as the compressed data. The second form is generated by "thresholding" $e_g(i)$ and generating the ternary signal. The "thresholding" is done according to equation below:

$$\text{Ternary}(e_g(i)) = \begin{cases} +1 & e_g(i) > Th(b) \\ -1 & e_g(i) < -Th(b) \\ 0 & Th(b) \geq e_g(i) \geq -Th(b) \end{cases} \qquad (5)$$

where $Th(b)$ is the threshold calculated for every beat according to equation(6) below.

$$Th(b) = K \sum_{i=1} b(i)^2 \ / \ T \qquad (6)$$

where b is the index of beat in record under study, i is the index of values of the beat signal , n is number of points in single beat, T is the beat length in seconds and K is a scaling factor (around 0.33).

The primitive ternary signal generated by the second NN is fed to the third SOM network. This network reads a vector of inputs (250 points) at once, and uses them in the classification process. Of course, the output of the second network is a series of points that is saved in a buffer and subjected to the SOM every 500 points at once. This SOM uses 20 neurons in a (4X5) topology. This type of networks has the ability to force adjacent neurons in the feature map (network) to respond to similar feature inputs dividing inputs into classes. Each class is associated with a single neuron in map. Figure (2) shows a diagram explaining the architecture of this all-neural classifier.

At first, the Hermite functions are used to model QRS complexes. This was originally proposed by Sornmo et. al[3]. Although, this model does not reflect an optimum modeling of a real life ECG with all noise and irregularities associated with it, it offers excellent testing signals before a move toward real ECG is made. Based on our observations on this artificial model we study the learning capabilities and sensitivity of our classifier. We generated a series of 20 QRS shapes using linear combinations of the first three Hermite functions: Φ_0, Φ_1 , Φ_2

$$H_j(i) = \begin{cases} \Phi_0(i)(N-2j)/N + \Phi_1(i)\,2j/N & \text{for} \quad 0 < j \le N/2 \\ \Phi_0(i)(2j-N)/N + 2\Phi_1(i)(N-j)/N + \Phi_2(i)\,(j-N)/N & \text{for} \quad N/2 \ge j < N \end{cases} \qquad (7)$$

N is equal to 20. With these templates we could fluently change its morphology from mono "phasic" to "biphasic" and from "biphasic" to three-phasic. These functions are used as reference signals in training. Training is implemented in two phases; first , the predictor network is trained using the GA using 5000 sample points representing the whole series of the 20 Hermite function templates (each wave is modeled by 250 points). Second, the ternary signals are generated using second neural network. Here, training is not needed since this neural network is just an execution machine that generates two kinds of error signals as mentioned earlier. The ternary signals, consisting of 5000 points is then used in training the SOM on a basis of 250 points in every training vector. The SOM is initialized with 20 neurons and after only 1000 learning phases 20 classes for each Hermite wave are generated. Testing this classifier in retrievals is not worthy without employing

noisy signals. Noise is added to the Hermite waves with SNR values of 10dB, 5dB, 3dB, and 0dB. Classification is 100% until SNR values of 3dB and below are reached. Table(2) shows a summary of those results. Figures (3) & (4) show the noisy artificial ECG's (Hermite waves) with worst case added noise (SNR= 0dB).

Table 2 Percentage of classification by SOM under different SNR

SNR	10*dB*	5*dB*	3*dB*	0*dB*
Correct Class	100%	100%	60%	30%

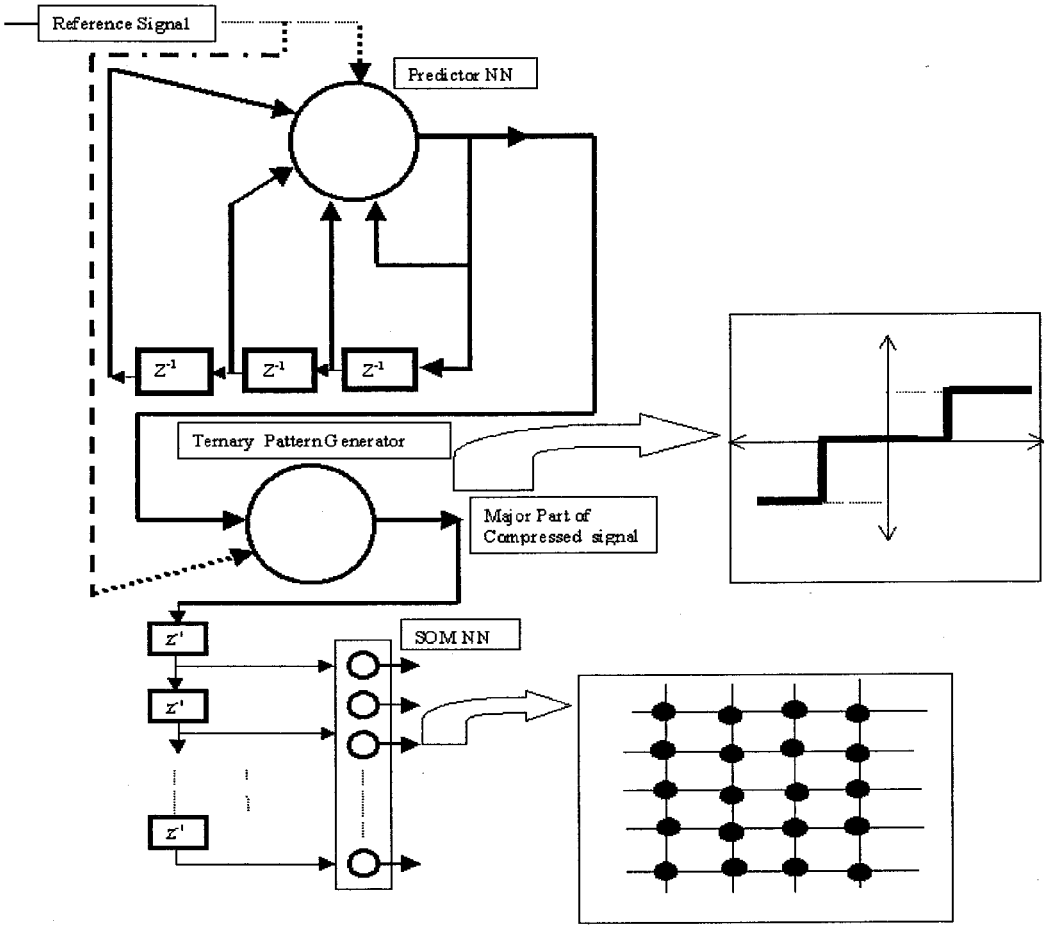

Fig. 2 Cascaded neural network classifier

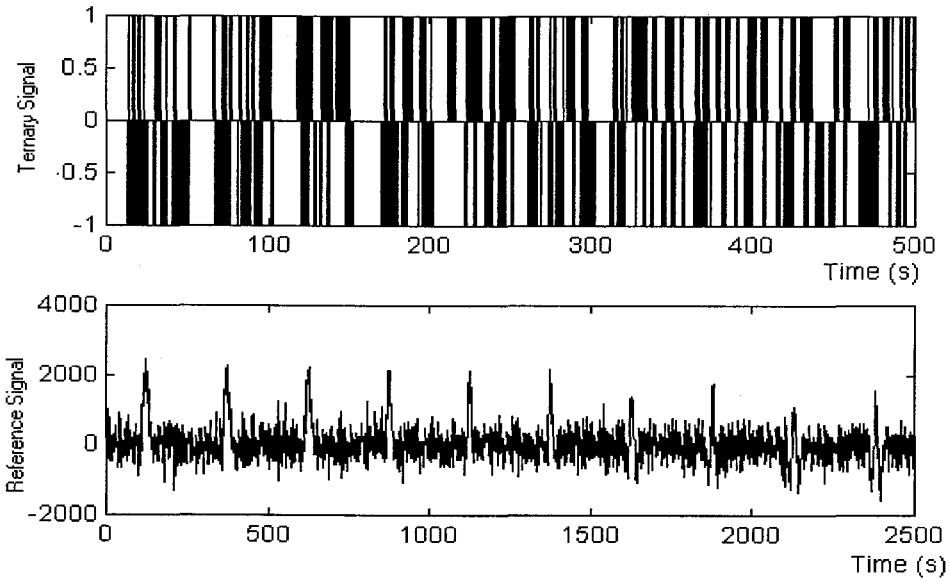

Fig. 3 Artif.ECG's with worst case(0dB SNR) (below)/ternary error signal(up)(1).

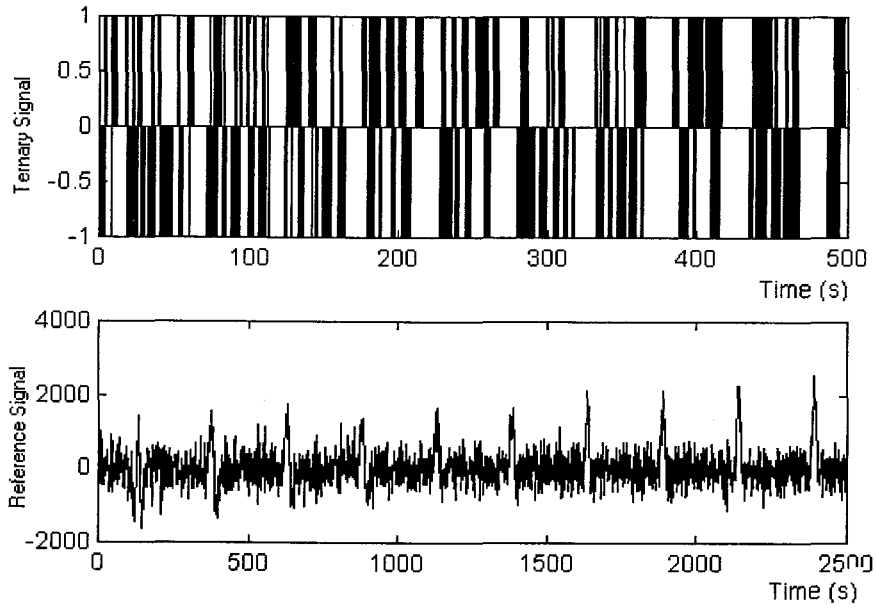

Fig.4 Artif.ECG's with worst case(0dB SNR) (below)/ternary error signal(up)(2).

5. TESTING NEURAL CLASSIFIER WITH REAL ECG SIGNALS

After the encouraging results we had with artificial ECG beats, we moved toward building a new classifier with real data taken from MIT database. Definitely, we could not use all the records available on the MIT/BIH arrhythmia database, we did our simulations on 10 randomly picked records. Each record here is a 30 min recording representing one patient. First 5 min of the record are used in building (training) the classifier and the rest are used in testing it. The 5 min for every patient record are about 350 ECG beats with sampling rate of 250 samples/s. The predictor network is trained using those 350 ECG beats. The GA is our training algorithm and fast convergence is achieved. As described earlier, the feature extraction stage comes next. The reference signal (original real signal) and the predicted signal are used to generate the error signal and the primitive ternary signal and which is our feature vector for every beat. Although, the generated ternary template length is equal to the ECG Beat length, the fact that it is a primitive signal makes it very simple and requires fewer space to store. In our study, we are interested in classifying the signals into three basic categories; class1, which is normal beats, supraventricular ectopic beat, and others, class2, which is the premature ventricular contraction beats (PCV's that resemble abnormal beats), and class3 which is unclassifiable beats [2]. These classifications are from the AAMI recommended practice [14]. Figures (5)& (6), show two samples of "notorious" real ECG signals taken from different MIT/BIH records. The reference (real) signal is shown in (b), the predicted signal is shown in (c), the ternary template is shown in (a), and the generated error signal shown in (d). In retrievals, rest of the record (25 minutes), are used as input for the predictor network and the ternary pattern generator network. No learning (fixed weights) is implemented for predictor network, and of course, no learning for the SOM. Training is only implemented during first 5 minutes of every new different records. For the 10 picked records from the standard MIT/BIH database, the process of 5 min training and 25 min retrieving is repeated for each record. Studying the performance of our classifier is based in three criteria; *classification* rate, *sensitivity*, and *predictivity* . This classification is based on the evaluation of our classifier according to its ability to classify the signal, first, as an ECG and not noise, and then to correctly classify it to the proper class. Every record contains around 2000-3000 beats, every beat is passed to the classifier and tested. The beat then could be TP, FP, TN, and PN. Where P and N means detected and not detected respectively, while T and F also mean correct or incorrect classification respectively. As a result of retrievals, we calculated for each record the aforementioned three definitions to judge the performance of our classifier. The calculations are as follows: Classification rate = $(TN+TP)/(TN+TP+FN+FP)$, Sensitivity = $TP/(TP+FN)$, and Predictivity = $TP/(TP+FP)$. Accordingly, total number of normal beats equals $TN+FP$ and total number of PVC beats equal $TP+FN$,for every single record. Table(4) shows beat-

by-beat, record-by-record testing results for the 10 picked records. Table (5) shows the values for the classification rate, sensitivity, and predictivity for our classifier.

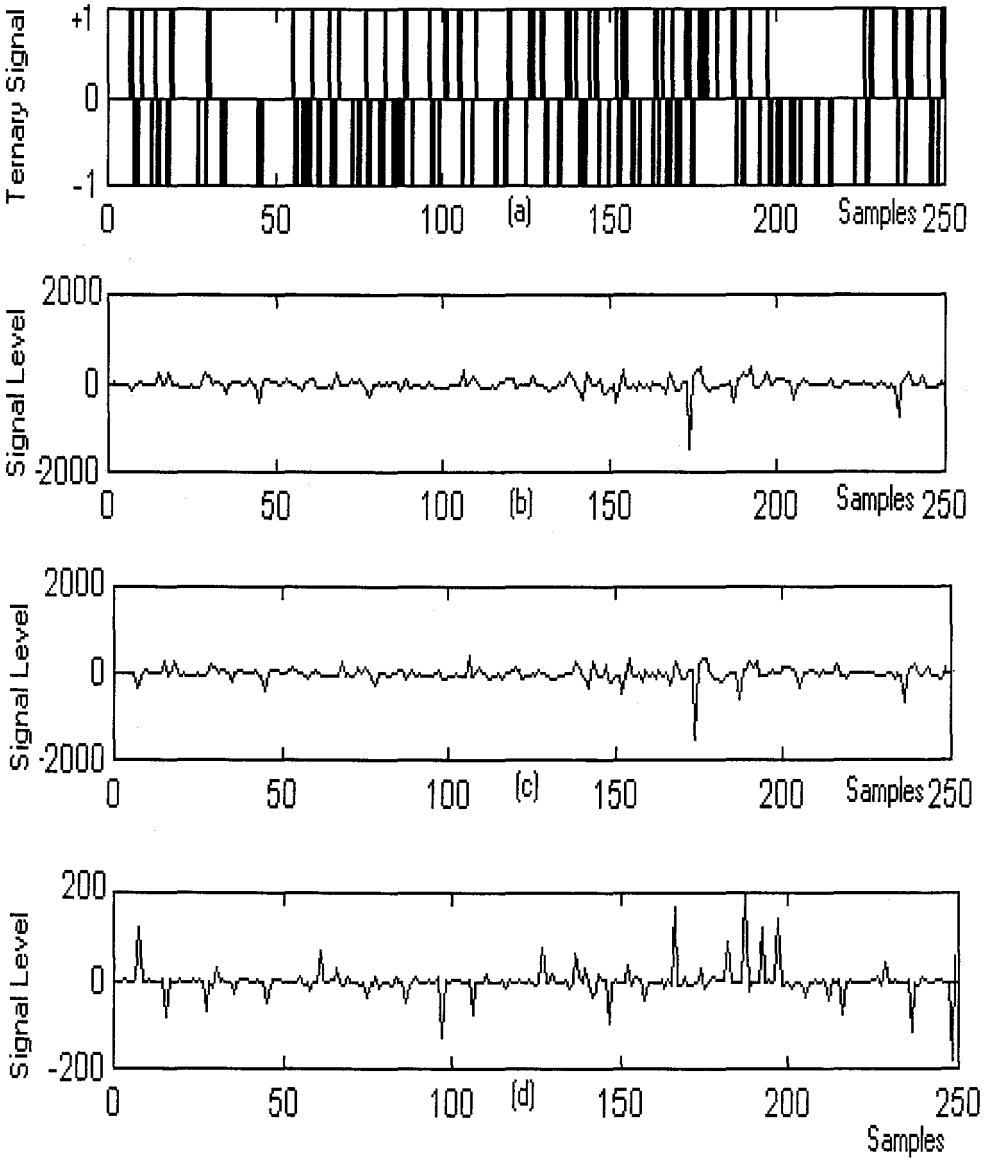

Fig.5 (a) Ternary template. (b) Reference ECG. (c) Predicted ECG. (e) Error signal.

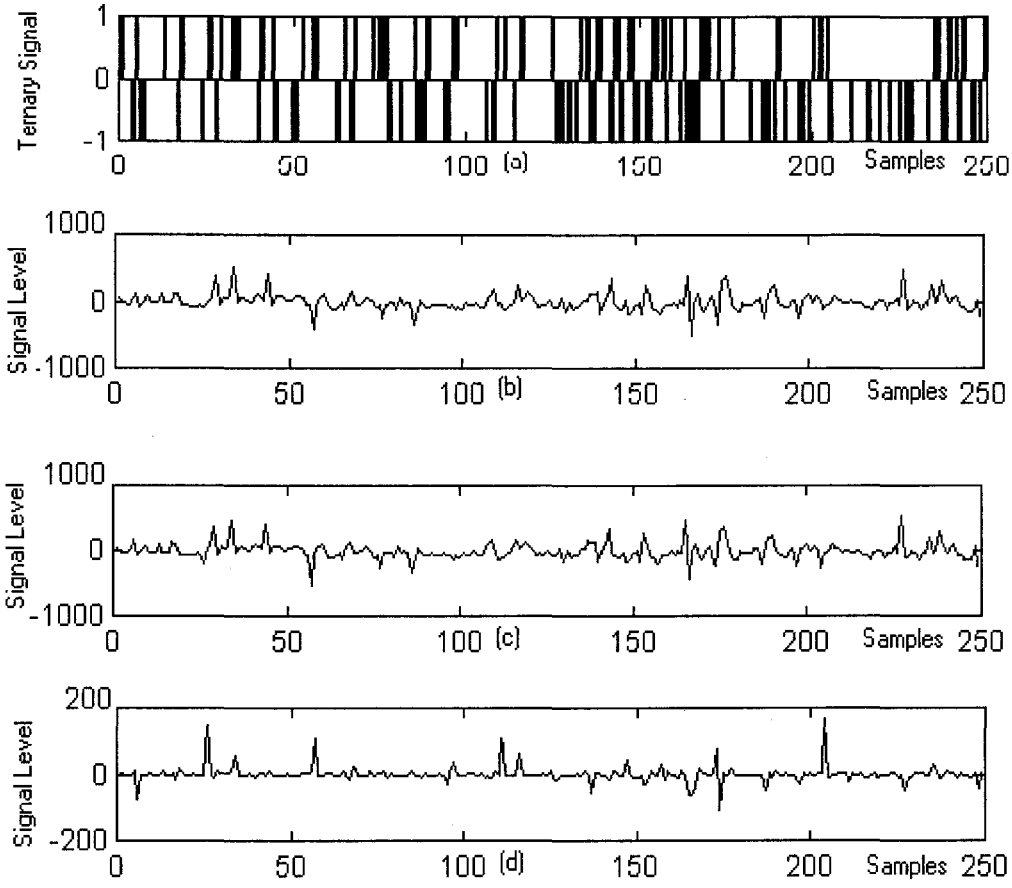

Fig. 6 (a) Ternary template (b) Reference ECG(c) Predicted ECG. (e) Error signal.

Table 3 Beat Classes of interest in this study.

Class1	**Normal Beat**
Class2	**Abnormal Beats (Premature Ventricular Contractions)**
Class3	**Unclassifiable**

Table 4 Beat-BY-Beat, Record-BY-Record Testing of the Neural Classifier.

	Neural classifier				Data in Record	
Record No.	TP	FP	FN	TN	Normal	PVC
1	707	24	54	1820	1844	761
2	0	0	198	1660	1660	198
3	0	0	18	2020	2020	18
4	360	132	50	2340	2472	410
5	0	0	71	2420	2420	71
6	0	0	173	2042	2042	173
7	877	107	48	1791	1898	925
8	0	0	1	2859	2859	1
9	177	86	8	2231	2317	185
10	0	0	220	2851	2851	220
sum	2121	349	841	22034	22383	2962

Table 5 Performance evaluation of Neural Classifier(values are in percent %, NaN means not a number)

Record No.	Classification rate	Sensitivity	Predictivity
1	97.0	93.0	97.0
2	89.3	0.0	NaN
3	99.1	0.0	NaN
4	93.7	88.0	73.0
5	97.1	0.0	NaN
6	92.2	0.0	NaN
7	94.5	95.0	89.0
8	100	0.0	NaN
9	88.1	96.0	67.0
10	74.4	0.0	NaN
avg	95.2	37.2	81.5

6. CONCLUSIONS

The simulations in this study covered both artificial and real ECG beats classification with additional outcome of error values can be used in reconstructing original data. As shown in Figures (5) & (6), the original signal can be regained by adding the error signal to the predicted signal. The predicted signal in itself can be

generated by turning on the predictor neural network. This neural network is made up of three real numbers only (the weights). The size of the error signal file is definitely smaller than the size of the original data file. According to our simulations, the original signal has maximum values of around 2000, while the maximum of error signals has values of around 200. The rest of error values were very low. The ratio of error files size to data files size was around 1 to 100 , in some cases even less than that. Of course, traditional techniques of data compression can be applied here to error signal for further reductions, but this would require less time and effort than what the original signal would require if it is to be compressed directly. We should keep in our mind that our main goal is to classify the original signal, and this additional advantage we had of compressing data is an additional big plus to our technique. Moreover, our classifier is a proposed single neural chip classifier, that doesn't require any elementary preprocessing for the ECG beat such as principal component analysis, or temporal parameters calculations (i.e. RR instantaneous interval, or width of complex QRS [1],[2]). The feature vector here doesn't carry readable information about the beat, but the embedded intrinsic information generated and realized by the cascaded neural network is used in distinguishing the different classes.

The simulations of the artificial ECG show that the 20 ternary templates of Hermite function are properly classified by the SOM into 20 distinct clusters. Learning phase took only 200 iterations, while retrievals are prompt and online. Although $H(n)$ and H (n +/-1) shapes are similar, the created ternary templates are sensitive enough for the SOM to classify them into different classes. Setting the "thresholding" value was crucial in creating distinct templates. The use of ternary templates helped in fine tuning cluster classes, while using real values directly requires another stage of "vector quantization" [15]. Testing same classifier with noisy reference signals showed robustness of that classifier; this is due to ability of pattern generator network to tolerate abrupt changes in the reference signal values. The pattern generator network transforms the reference signal into a brief primitive message enough to reflect the class of the signal and disregarding to some extent unnecessary details. The SOM network at the last stage reacts to the ternary pattern signal in parallel settling to a single neuron out of the 20 possible neurons in the map. Each neuron in the SOM is supposed to represent a class, learning continues until each artificial ECG (Hermite) is associated with a single neuron. On line retrievals with noisy signal show excellent classification rate.

Simulations for the real ECG taken from MIT database presented in Table(4) show high rate of classification for each beat in every record with TN and TP sums higher than FN and FP for the 10 records. Performance evaluation of the classifier is depicted in Table(5). It shows excellent classification rate over the 10 records we had. The classification rate reflects the total correct detection of beats over the

total correct and false detection. The patient (record) "adaptive-learning-then-retrieving" methodology, resulted in these good results. The 5 minutes of training for every patient separately have enhanced the performance of the classifier. Although we have not demonstrated that, but a global classifier for all records is a waste of time. One thing we have to mention is that the sensitivity of our classifier, which is the percentage of correctly detected beats to total number PVC's (correctly detected or incorrectly not detected), is high for some records and very low for others. This is due to the fact that the morphology of the beats for some patients varies from others even if the beats are normal or PVC's. The predictivity, which is the percentage of correctly detected beats to all detected beats (whether correct or not correct) is also high for our classifier. This measure reflects the percentage of correct alarms to total number of alarms.

In this paper, we developed an all-neural, compact, hardware realizable, patient adaptable ECG beats classifier. We evaluated its performance on artificial and real beats for different records. To the best of our knowledge, this approach has never been done before. We believe that this approach is a step toward other automated patient monitoring algorithms that would support decentralized remote patient-monitoring systems.

REFERENCES

1. **Al-Shrouf,A.,1994**."Application of Method of Linear Interpolation for ECG Signal Processing and Data Compression," Ph.D. dissertation, Sylsia Tech. Univ., Poland.

2. **Hu, Y., Palreddy, S.** and **Tompkins, W. J. , 1997** "A Patient-Adaptable ECG Beat Classifier Using a Mixture of Experts Approach," IEEE Trans. on Biomedical Engineering, vol. 44, no. 9,pp. 891-900.

3. **Sornmo, L.Borjesson, P., Nygards, M., Pahlm, O., 1981**"A Method for Evaluation of QRS Shape Features Using a Mathematical Model for ECG",IEEE Trans. on Biomedical Engineering, vol 28,no 10,pp. 713-717.

4. **Lin, K.** and **Chang, W.,1989** "QRS Feature Extraction Using Linear Prediction," IEEE Trans. on Biomedical Engineering, vol.36, no 10, pp.1050-1055.

5. **Eisenstein, B.** and **Vaccaro, R., 1982** "Feature Extraction by System Identification," IEEE Trans. on Systems, Man, and Cybernetics, vol.SMC-12,no 1, pp.42-50.

6. **Frankiewicz, Z.,** and **Shrouf, A.,1991**"ECG Beat Classification Using Linear Prediction Error Signal," Proceedings of Medical Informatics, Vienna, Austria.

7. **R. C. Gonzalez** and **P. Wintz**,"Digital Image Processing (2d. ed.)," Addison-Wesley, Reading, Mass.

8. **Holland, JH. , 1975**, Adaptation in Natural and Artificial Systems . Ann Arbor, MI: University of Michigan Press.

9. **Goldberg, D.E.,1989**"Genetic Algorithms in Search Optimization, and Machine Learning", Reading, MA: Addison-Wesley.

10. **Abu Zitar, R.A., Hassoun, M.H.,1995**,"Neurocontrollers trained with rules extracted by a genetic assisted reinforcement learning system", IEEE Trans. on Neural Networks, **6** (4), 859--879.

11. **Hassoun, M. H., 1995,** Fundamentals of Artificial Neural Networks, MIT Press, Cambridge, Mass.

12. **Kohonen, T.,1993**"Self-organizing maps: Optimization approaches, in Artificial Neural Networks, T. Kohonen, K.Makisara, O.Simula, and J.Kanga, eds., pp. 1147-1156. IEEE, New York.

13. **Kohonen, T.,1990** "The self-organizing map," Proc. IEEE, vol. 78, no. 9, pp. 1464- 1480.

14. **Mark, R.** and **Moody, G., 1987**,"AAMI-recommended practice: Testing and reporting performance resultsof ventricular arrhythmia detection algorithms," Association for the Advancement of Medical Instr, Arrhythmia Monitoring Subcommittee, Tech. Rep. AAMI ECRA.

15. **Bortolan, G., Degani, R.** and **Willems, J. L.,1991** "ECG classification with neural networks and cluster analysis," in Proc. Computers in Cardiology, pp. 177-180.