International Journal of Computer and Communication Technology

Volume 1 | Issue 3 Article 4

July 2010

ECG QRS Enhancement Using Artificial Neural Network

Sambita Dalal
Department of Applied Electronics and Instrumentation Engineering ITER, S'O'A University Bhubaneswar, India, sandeepta.dalal@gmail.com

Laxmikanta Sahoo
Department of Applied Electronics and Instrumentation Engineering ITER, S'O'A University Bhubaneswar, India, laxmikanta.lk@gmail.com

ECG QRS Enhancement Using Artificial Neural Network

Sambita Dalal, Laxmikanta Sahoo
Department of Applied Electronics and Instrumentation Engineering
ITER, S'O'A University
Bhubaneswar, India
sandeepta.dalal@gmail.com , laxmikanta.lk@gmail.com

***Abstract*—Soft computing is a new approach to construct intelligent systems. The complex real world problems require intelligent systems that combine knowledge, techniques and methodologies from various sources. Neural networks recognize patterns and adapt themselves to cope with changing environments. Artificial neural network has potential applications in the field of ECG diagnosis measures. So noise reduced QRS complex of ECG signal is of utmost importance for automatic ECG interpretation and analysis. Noise is an unwanted energy, which interferes with the desired signal. Noise cancellation is mainly used as interference canceling in ECG, echo elimination on long distance telephone transmission lines and antenna side lobe interference canceling. In the study, the ECG signal is trained following various artificial neural network based algorithms to enhance the QRS complex by reducing noise for further analysis.**

Keywords- neural network; ECG; adaptive filter

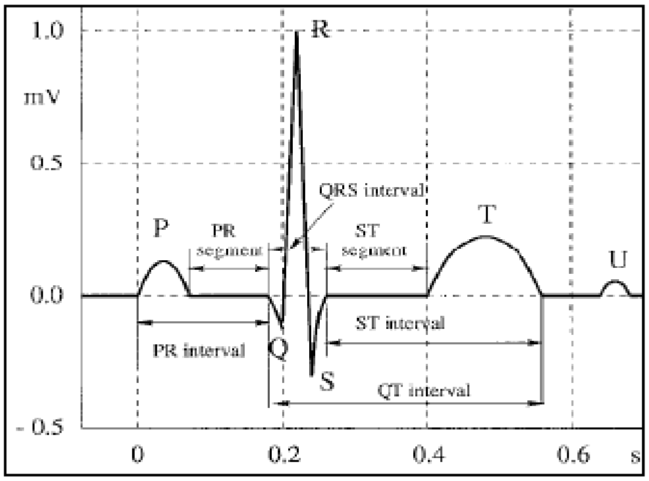

Figure 1. Standard scalar electrocardiogram

I. Introduction

Heart diseases, which are one of the death reasons of man/women, are among the important problems on this century. One of the ways to diagnose heart diseases is to use electrocardiogram (ECG) signals. ECG records the electrical activity of heart. The ECG signal is a time-varying signal reflecting ionic current flow which causes the cardiac fibers to contract and subsequently relax. A standard scalar electrocardiogram is shown in Fige1. It consists of P wave, PR-interval, PR segment, QRS complex, ST segment, ST interval and QT interval and T wave. A single normal cycle of ECG represents the successive atrial depolarization/ repolarization and ventricular depolarization/repolarization which occur with every heart beat. The P wave represents atrial depolarization, the QRS-complex represents left ventricular depolarization and the T wave represents the left ventricular repolarization.

Various methods, for QRS detection, found in literature are: using slope or derivative of ECG signal, methods based on digital filters, statistical methods, pattern recognition, Artificial Neural Networks, Genetic Algorithm so on.

II. Artificial Neural Network

Neural network can be trained to perform a particular function by adjusting the value of the connections (weights) between elements. Commonly neural networks are adjusted, or trained, **so** that a particular input leads to a specific target output. The basic architecture consists of three types of neuron layers: input, hidden, and output layers. The network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically many such input target pairs are used in the supervised learning to train a network.

Artificial Neural Network (ANN) is the generalization of mathematical models of biological nervous systems. In the artificial adaptation of human brain the artificial neural network has preserved three basic characteristics. Neural network learns from experience; generalize from learned responses, and abstract essential**,** pattern from inputs. Neural networks have been trained to perform complex functions in various fields of application including pattern recognition, identification, classification, speech or image processing and control systems.

III. LMS Adaptive Filter Technique

Least mean squares (LMS) algorithms is a type of adaptive filter used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean squares of the error signal (difference between the desired and the actual signal). It is a stochastic gradient descent method in which the filter is only adapted based on the error at the current time.

A. Linear Adaptive Filtering Problem Formulation

Fig. 2 describes a linear adaptive filtering problem. An unknown system h(n) is to be identified and the adaptive filter attempts to adapt the filter $\hat{h}(n)$ to make it as close as possible to h(n), while using only observable signals x(n), d(n) and e(n); but y(n), v(n) and h(n) are not directly observable.

Definition of symbols:

$$x(n)=[x(n),x(n \quad 1),\ldots\ldots,x(n \quad p \quad 1)]^T \tag{1}$$

$$h(n)=[h_0(n),h_1(n),\ldots\ldots,h_{p-1}(n)]^T,\ h(n) \quad C^P \tag{2}$$

$$y(n) = h^H(n) \cdot x(n) \tag{3}$$

$$d(n) = y(n) \quad v(n) \tag{4}$$

$$e(n)=d(n) \quad y(n)=d(n) \quad \hat{h}^H(n) \cdot x(n) \tag{5}$$

Updated algorithm follows as:

$$\hat{h}(n \quad 1) = \hat{h}(n) \quad \mu\, x(n) \cdot e^*(n) \tag{6}$$

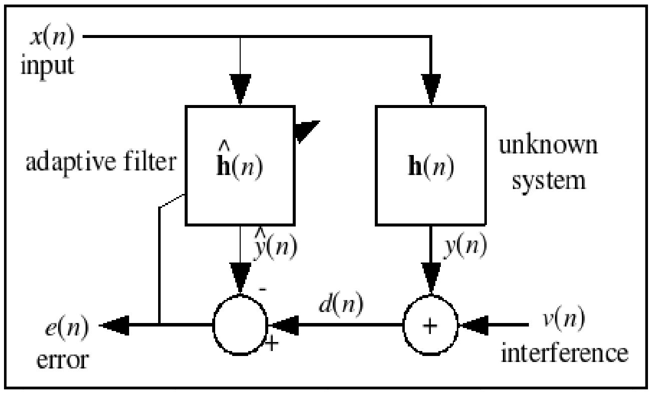

Figure 2. Linear Adaptive Filter Problem

B. LMS Adaptive Filter With A Dynamic Structure.

The general LMS adaptive filter removes noise or obtains a desired signal by adapting the filter coefficient with the least-mean-square algorithm based on a given filter order [7,8]. As shown in Fig. 3, the output of the LMS adaptive filter can be expressed as:

$$\hat{S}(n) = S(n) \quad N(n) \quad N^{\wedge}(n), \tag{7}$$

$$N^{\wedge}(n) = \Sigma\, W_i \; NR(n \quad i), \tag{8}$$

$$W_i(n \quad 1) = W_i(n) \quad 2\mu\, S(n)\, NR(n \quad), \tag{9}$$

where i: 0, 1, 2……. L

μ: convergence constant

L: filter order

S (n): original ECG signal

N (n): noise signal

Ŝ (n) = E (n): filtered ECG signal

N^ (n): the estimated noise signal

W_i (n): the filter coefficient

NR (n): the reference number

Convergence error for a period of ECG signal can be stated as:

$$E(p) = \Sigma\, \hat{S}(n)^2 \tag{10}$$

$$\Delta E = [E(p) - E(p-1) - E(p-2) \quad E(p-3)] \tag{11}$$

where p is the period of the ECG signal and ΔE is the difference between the error in the previous convergence and that in the current convergence.

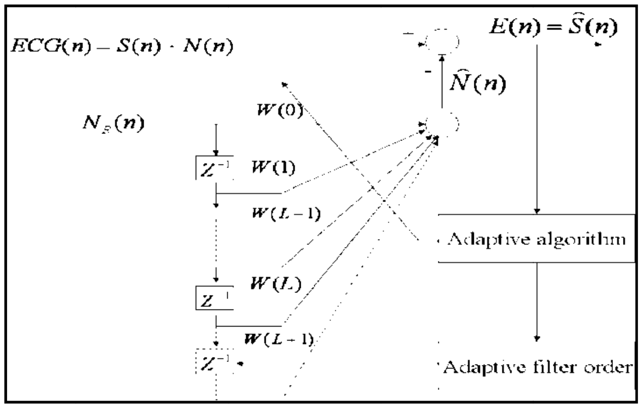

Figure 3. LMS Adaptive filter with a dynamic structure

IV. Kohonen Rule

The Kohonen network is an extension to the competitive learning network, although this is chronologically incorrect. Now, when learning patterns are presented to the network, the weights to the output units are thus adapted such that the order present in the input space ℜN is preserved in the output, i.e., the neurons in S. This means that learning patterns which are near to each other in the input space (where 'near' is determined by the distance measure used in finding the winning unit) must be mapped on output units which are also near to each other, i.e., the same or neighboring units. Using the formula of the SOM learning algorithm, the winning unit is determined. Thus, if inputs are uniformly distributed in ℜN and the order must be preserved, the dimensionality of S must be at least N. The mapping, which represents a discretisation of the input space, is said to be topology preserving. However, if the inputs are restricted to a subspace of ℜN, a Kohonen network can be used of lower dimensionality.

A. The SOM Learning Algorithm

During the training period, each unit with a positive activity within the neighborhood of the winning unit participates in the learning process. The learning process can be described by the equation:

$$w_i = \alpha (t) (x-w_{i,}) U (y_i) \tag{12}$$

where w, is the weight vector of the ith unit and x is the input vector. The function U (y_i) is zero unless $y_i > 0$ in which case U (y_i) = 1, ensuring that only those units with positive activity participate in the learning process. The factor α (t) is written as a function of time to anticipate our desire to change it as learning progresses.

For an input vector x, the winning unit can be determined by:

$$\|x-w_c\| = \min \{\|x-w_i\|\} \tag{13}$$

where the index c refers to the winning unit.

Instead of updating the weights of the winning unit only, a physical neighborhood around the unit is defined, and all units within this neighborhood participate in the weight-update process. Each weight vector participating in the update process rotates slightly toward the input vector, x. Once training has progressed sufficiently, the weight vector on each unit will converge to a value that is representative of the coordinates of the points near the physical location of the unit.

If c is the winning unit, and N_c. is the list of unit indices that make up the neighborhood, then the weight-update equations are:

$$w_i(t+1) = \begin{cases} w_i(t) + \alpha(t)\big(x - w_i(t)\big), \\ 0 \qquad\qquad otherwise \end{cases} \tag{14}$$

V. Perceptron Learning Rule

The perceptron is a single layer neural network whose weights and biases could be trained to produce a correct target vector when presented with the corresponding input vector. The training technique used is called the perceptron learning rule. Suppose there is a set of learning samples consisting of an input vector x and a desired output d (k). For a classification task, the d (k) is usually +1 or −1. The perceptron learning rule can be stated as follows:

1) *Start with random weights for the connections.*

2) *Select an input vector x from the set of training samples.*

3) *If output yk != d(k) (the perceptron gives an incorrect response), modify all connections wi according to: δwi = η(d(k) − yk)xi; (η = learning rate).*

4) *Go back to step 2.*

Perceptrons are trained on examples of desired behavior. The desired behaviour can be summarized by a set of input, output pairs where p is an input to the network and t is the corresponding correct (target) output. The objective is to reduce the error e, which is the difference between the neuron responses a and the target vector t. Each time learning rule is executed, the perceptron has a better chance of producing the correct outputs. The perceptron rule is proven to converge on a solution in a finite number of iterations if a solution exists.

The perceptron learning rule thus can be summarized as follows:

$$W^{new} = W^{old} + ep^T \quad \text{and} \quad b^{new} = b^{old} + e, \tag{15}$$

where e=t−a.

A. Adaline Network

The perceptron learning rule is applied to the 'adaptive linear elements', also named as Adaline network. The perceptron learning rule uses the output of the threshold function either -1 or +1 for learning, whereas, the delta rule uses the net output without further mapping into output values -1 or +1.In a simple physical implementation as shown in the Fig. 4, usually the central block, the summer, is also followed by a quantiser which outputs either +1 of −1, depending on the polarity of the sum. Although the adaptive process is here exemplified in a case when there is only one output, it may be clear that a system with many parallel outputs is directly implementable by multiple units of the above kind.

If the input conductances are denoted by w_i, i = 0; 1; ; ; ; n, and the input and output signals by x_i and y, respectively, then the output of the central block is defined to be:

$$y=\sum_{i=1}^{n} w_i x_i + \theta, \tag{16}$$

where θ = w_0. The problem is to determine the coeficients w_i, i = 0, 1......., n, in such a way that the input-output response is correct for a large number of arbitrarily chosen signal sets. If an exact mapping is not possible, the average error must be minimised, for instance, in the sense of least squares. An adaptive operation means that there exists a mechanism by which w_i can be adjusted, usually iteratively, to attain the correct values.

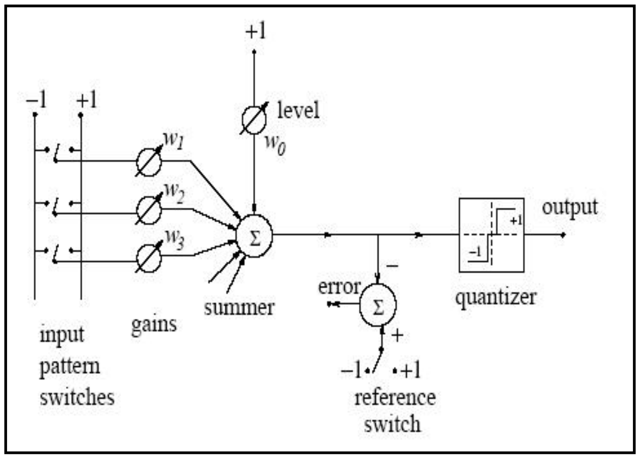

Figure 4. Block Diagram For Adaline Network

VI. BACKPROPAGATION

Backpropagation networks are necessarily multilayer perceptrons (usually with one input, one hidden, and one output layer).It is a supervised learning method, and is an implementation of the Delta rule. It is most useful for feed-forward networks (networks that have no feedback, or simply, that have no connections that loop). The central idea behind this solution is that the errors for the units of the hidden layer are determined by back-propagating the errors of the units of the output layer. Backpropagation requires that the activation function used by the artificial neurons is differentiable.

Although back propagation can be applied to networks with any number of layers, just as for networks with binary units it has been shown that only one layer of hidden units succeeds to approximate any function with finitely many discontinuities to arbitrary precision, provided the activation functions of the hidden units are non-linear (the universal approximation theorem).

A. Multilayer Feedforward Network

A multilayered feedforward network has a layered structure as shown in Fig. 5. Each layer consists of units which receive their input from units from a layer directly below and send their output to units in a layer directly above the unit.

The activation of a hidden unit is a function F of the weighted inputs plus a bias, as given in equation:

$$Y_k(t+1) = F_k(s_k(t)) = F_k(\textstyle\sum_j w_{jk}(t) y_j(t) + \theta_k(t)) \quad (17)$$

The output of the hidden units is distributed over the next layer of N_h; 2 hidden units, until the last layer of hidden units, of which the outputs are fed into a layer of No output units.

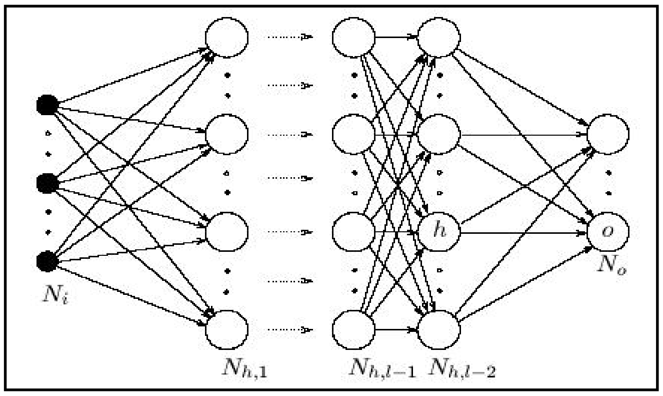

Figure 5. Multilayer feed forward neural network

B. Delta Rule

For using units with nonlinear activation functions, delta rule can be generalized. The activation function is a differentiable function of the total input, given as:

$$y_k^p = F(s_k^p) \quad (18)$$

where $s_k^p = \sum_j w_{jk} y_j^p + \theta_k$. After correct generalization of delta rule, equations:

$$\partial_o^p = (d_o^p - y_o^p) F'_o(s_o^p), \quad (19)$$

$$\text{and} \quad \partial_h^p = F'(s_h^p) \textstyle\sum_{o=1}^{N_o} \partial_o^p w_{ho}. \quad (20)$$

give a recursive procedure for computing the δ's for all units in the network, which are then used to compute the weight changes according to equation.This procedure constitutes the generalized delta rule for a feedforward network of nonlinear activation function.

C. Backpropagation Technique

The application of the generalised delta rule thus involves two phases: the input x is presented and propagated forward through the network to compute the output values y_o^p , which when compared with its desired value d_o, results in an error signal ∂_o^p which is backpropagated through the network.

1) Present a training sample to the neural network.

2) Compare the network's output to the desired output from that sample. Calculate the error in each output neuron.

3) For each neuron, calculate what the output should have been, and a scaling factor, how much lower or higher the output must be adjusted to match the desired output. This is the local error.

4) Adjust the weights of each neuron to lower the local error.

5) Assign "blame" for the local error to neurons at the previous level, giving greater responsibility to neurons connected by stronger weights.

6) Repeat from step 3 on the neurons at the previous level, using each one's "blame" as its error.

D. Backpropagation Algorithm

Initialize the weights in the network (often randomly).

For each example e in the training set,

- O= neural-net-output(network, e) ; forward pass
- T = teacher output for e
- Calculate error (T - O) at the output units
- Compute delta, wh for all weights from hidden layer to output layer ; backward pass
- Compute delta_wi for all weights from input layer to hidden layer ; backward pass continued
- Update the weights in the network
- Until all examples classified correctly or stopping criterion satisfied
- Return the network

So technically, back propagation is used to calculate the gradient of the error of the network with respect to the network's modifiable weights. This gradient is almost always then used in a simple stochastic gradient descent algorithm to find weights that minimize the error. It usually allows quick convergence on satisfactory local minima for error in the kind of networks to which it is suited. A major limitation of backpropagation is that the result may generally converge to any local minimum on the error surface, since stochastic gradient descent exists on a non-linear surface.

VII. RESULTS

The ECG signal is generated using the fourier series and implemented in the software using various functions. Then it was trained on basis of various above stated artificial neural network methods. Digital signal processing techniques has been used for filtering and smoothening of signals, enhancement of visual images, and compression of data, recognition and generation purposes. The results obtained can be viewed from the graphs in Fig. 6, Fig. 7, Fig. 8, Fig. 9, Fig. 10 and Fig. 11 as follows.

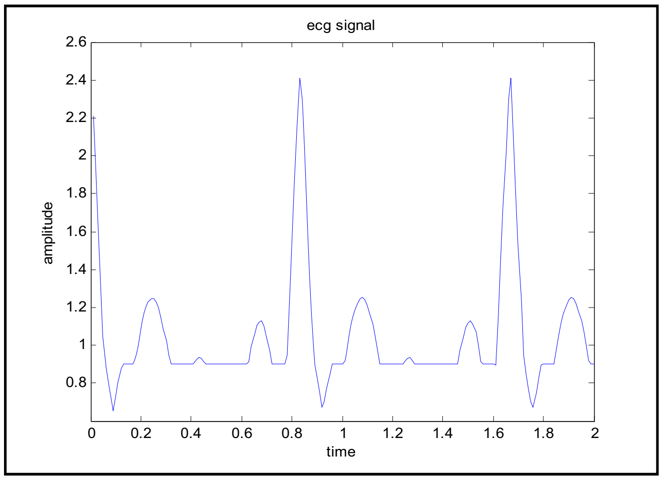

Figure 6. Generation Of ECG Signal

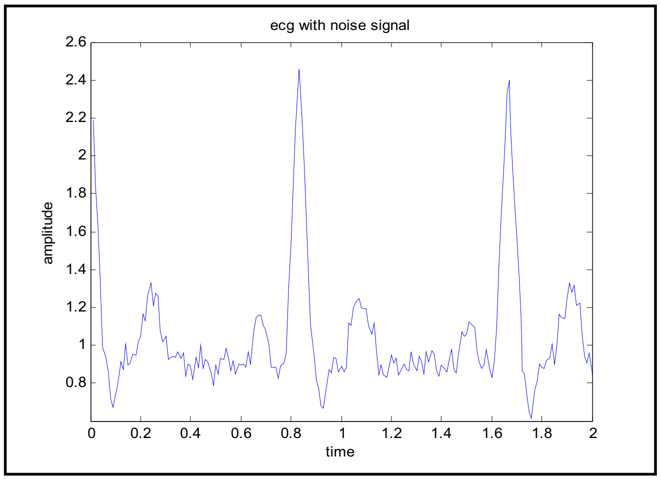

Figure 7. Adding noise to ECG signal

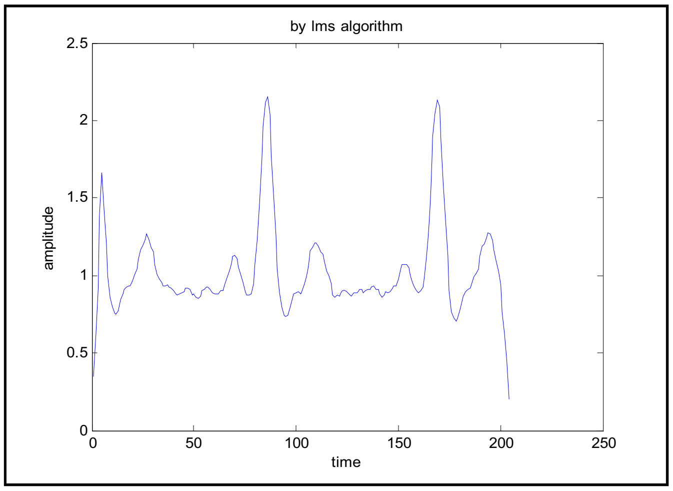

Figure 8. Enhanced ECG with mean square error obtained 0.0768 in LMS algorithm

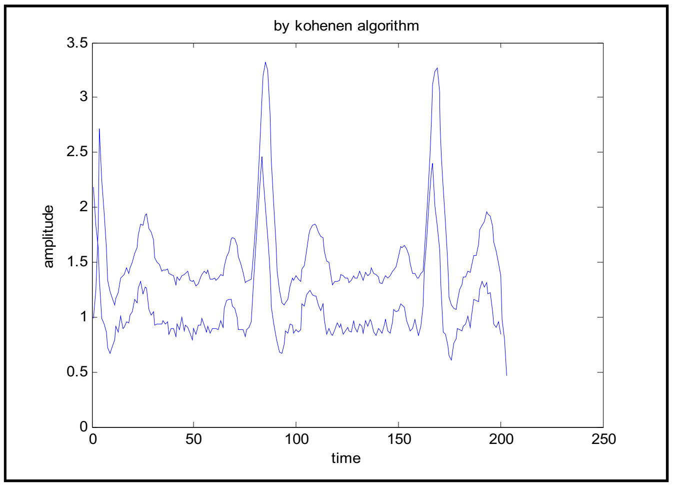

Figure 9. Enhanced ECG with mean square error obtained 0.3656 in Kohonen rule

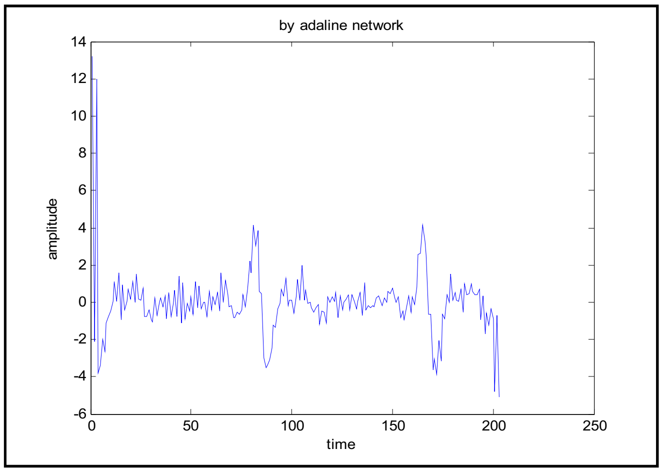

Figure 10. Enhanced ECG with mean square error obtained 3.9670 in adaline network

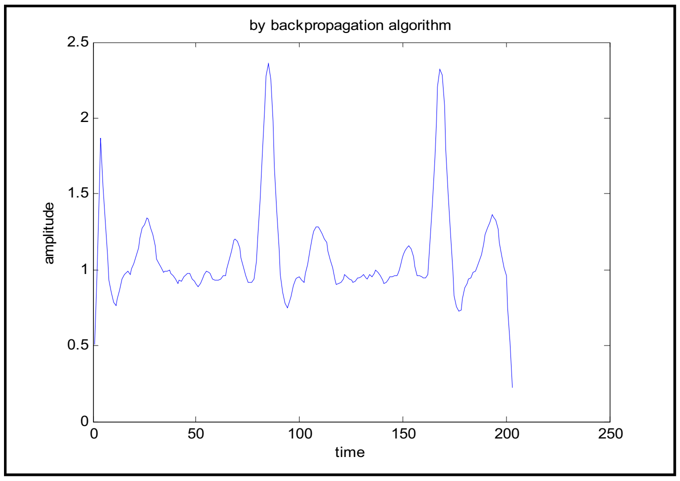

Figure 11. Enhanced ECG with mean square error obtained 0.0519 in backpropagation algorithm

VIII. CONCLUSION

In this paper, clean & noisy ECG signal has been trained with noise removing algorithms as: LMS adaptive filtering, kohonen rule, perceptron learning rule and finally backpropagation algorithm. Convolution technique has been used for filtering of signals and enhancement of images. Thus, it can be concluded that noise from ECG signal was eliminated to a large extent, with backpropagation algorithm giving best result with least error as compared to other network algorithms. The enhanced ECG signal can be used for automatic ECG interpretation to help reduce the burden of interpreting the ECG.

ACKNOWLEDGMENT

We express our deep sense of gratitude and thanks to Payal Samanta, P.Suranjika Samal, Bandita Banapriya and Stithaprajyna Dash for their sincere cooperation in the completion of the analysis and study.

REFERENCES

[1] Saad Alshaban and Rawaa Ali, "Using neural network and fuzzy software for the classification of ECG Signals". Research journal of applied sciences, Engineering and Technology 2(1): 5-10, 2010. ISSN: 2040-7467 Maxwell Scientific Organization, 2009.

[2] Neural Network Toolbox 6 User's Guide by Howard Demuth, Mark Beale and Martin Hagan.

[3] Jamshid Nazari and Okan K. Ersoy "Implementation of backpropagation neural networks with MATLAB". Electrical and Computer Engineering technical reports at Purdue Libraries in year 1992.

[4] Alfonso "Application of artificial neural network for ECG signal detection and classification", Journal of Electrocardiography, Vol 26 Supplement.

[5] Thomas Murphy. "Explaining convolution using MATLAB".

[6] MATLAB The Language of Technical Computing, The Mathworks.

[7] "Artificial Neural Networks" by Ajith Abraham. Chapter 29, pages 901 to 908.

[8] "Implementing Artificial Neural Networks in MATLAB" by Sivanand.

[9] G. Vijaya, V. Kumar and H.K. Verma, "ANN-based QRS complex analysis of ECG," J. Med. Eng. Technol., vol. 22, no. 4, pp. 160-167, 1998.

[10] "Kohonen T: Self-organization and associative memory". Springer-Verlag. Berlin, 1984.

[11] J.S.R.Jang, C.T.Sun and E.mizuatani, "Neuro-fuzzy and Soft computing". Prentice Hall International Inc., 1997.

[12] Martin T.Hagan, Howard B. Demuth, Mark Beale, "Neural network design".