http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 5 (1): 45-50, 2006
ISSN 1812-5638

Enhanced Neural Networks Model Based on a Single Layer Linear Counterpropagation for Prediction and Function Approximation

[1]Sameh Ghwanmeh, [2]Riyad Al-Shalabi, [2]Ghassan Kana'n and [2]Luai Alnemi
[1]Department of Computer Engineering, Yarmouk University, Irbid, Jordan
[2]Faculty of Information Technology, Yarmouk University, Irbid, Jordan

Abstract: This study investigated the use of neural networks in function approximation, data fitting and prediction. Due to its superior performance, the counterpropagation network was considered and an attempt was made to enhance its performance. As a result of this research, we proposed a new neural network architecture named Single Layer Linear Counterpropagation (SLLIC) network. The SLLIC neural net has the following additional features: weight Initialization, automatic structure determination and higher order neural network concepts. The SLLIC network was tested and results show that the performance of the system in terms of good approximation or prediction is comparable to and some times better than other neural nets architecture's and traditional techniques.

Key words: Neural networks, function approximation, prediction, forecasting

INTRODUCTION

Function approximation is a central issue in subjects as diverse as pattern recognition, control theory and statistics. Predication (forecasting) is essential for planning and operation Control in a variety of areas such as production management, inventory systems, quality control, financial planning and investment analysis. A number of neural networks architectures were published in the literature fore use in the above mentioned areas. The most prominent of these architectures are[1-4] :

- Group Method of Data Handling.
- Back propagation Nets.
- Radial Basis Function Network
- Forward-only Countrpropagation Net.

The objectives of this study were to investigate the performance of a number of neural networks architectures in function approximation and prediction and also attempt to improve on these architectures and compare the performance of the resulting neural net with that of other neural nets and traditional methods of function approximation and prediction.

From these experimental tests, it was found that the counterpropagation neural net produce more accurate results than the other architectures. Consequently, our attention was focused on trying to find the modifications that might enhance the performance of the counterpropagation neural network.

The original forward-only Counterpropagation network shown in Fig. 1 has only one output unit per each output element. So the cluster has only one average value that represents the average of the correct output value y associated with the input vector value x that cause the cluster to win the competition. This will limit the net capabilities to produce specific output values instead of approximating a function that interacts with the input vectors to determine the out put values. The objective of the net is to approximate a continuous function f: $R^n \rightarrow R$ that maps a subset of n-dimensional Euclidean space into

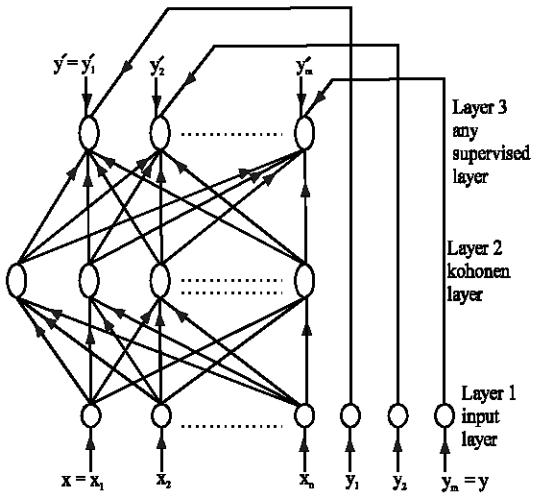

Fig. 1: The forward-only counterpropagation network

Corresponding Author: Sameh Ghwanmeh, Department of Computer Engineering , Yarmouk University, Irbid, Jordan
Tel: +962 2 721 1111/2551 Fax: +962 2 72 111 72

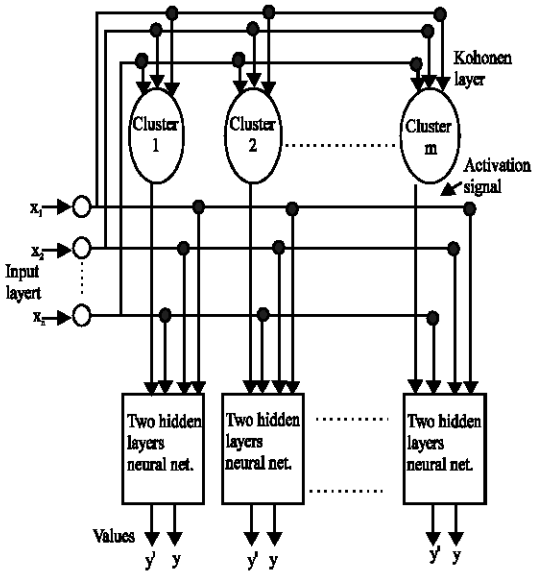

Fig. 2: Modified single layer linear counterpropagation network

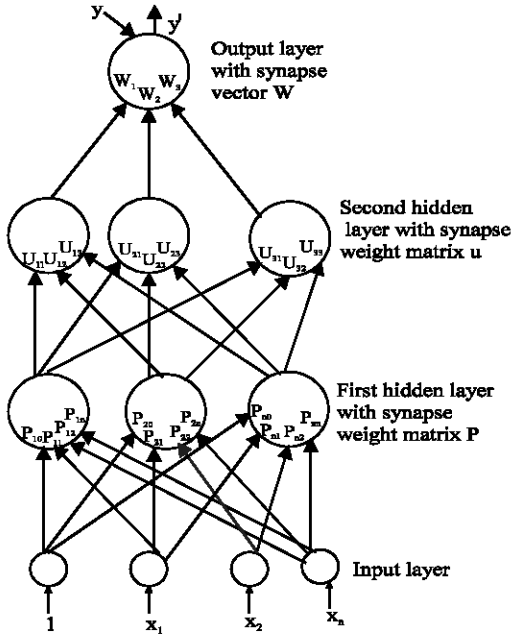

Fig. 3: Structure of the two hidden-layer net

the real numbers. Vector functions can be handled by creating one network per output coordinate.

FORWARD-ONLY COUNTERPROPAGATION MODIFICATION

The modification to the original counter-propagation network was accomplished by constructing a network for each cluster. Each cluster's network will approximate a function for the cluster subset of examples by mean of supervised learning. As first step, the constructed network will be multi-layered network with two hidden layers network with two hidden-layers since Cybenko[4] showed that two hidden layers network with continuous sigmoidal non linearity (i.e. a back propagation net) can approximate any continuous function arbitrarily well on a compact set. The resulting modified counterpropagation network is shown in Fig. 2. Figure 3 shows the structure of the two hidden layered networks referred to in Fig. 2.

As a further refinement, the two-hidden layers of the network of Fig. 3 is replaced by a single layer linear network. Although the former is more robust than the later, the single layer linear net is: simpler, requires less processing elements, does not need a normalization process and it is easily understood. In single layer linear network, robustness can be achieved by increasing number of clusters in the network.

The final resulting architecture of the single layer linear Counterpropagation (SLLIC) network, shown in Fig. 4, consists of a number of clusters, each of these clusters contains single processing element that approximates a function for the subset of examples associated with that cluster.

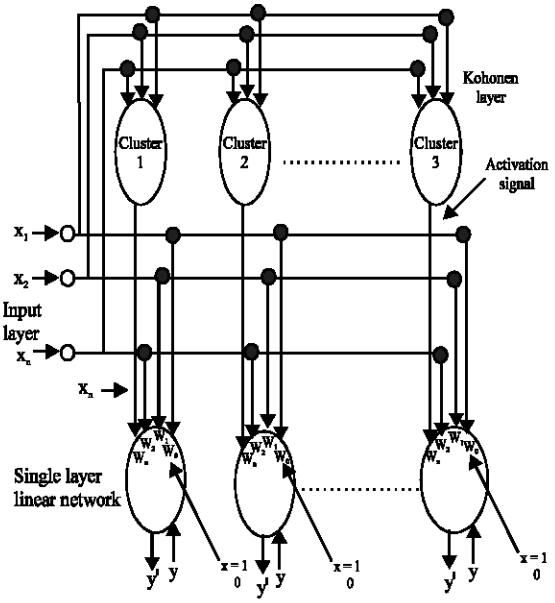

Fig. 4: The final structure of SLLIC network

LEARNING ALGORITHM FOR THE SLLIC NETWORK

The learning algorithm for the SLLIC net is based on the forward Counterpropagation net learning strategy[4,5].

Phase one {Kohonen layer}

Step 0: Initialize the clusters weight vectors and the learning rate

Step 1: Repeat from step 2-7 until the weight vectors converge (remain stable)

Step 2: For each training input vector do steps 3-5
Step3: Set the input layer activation's to vector x
Step4: Find winning cluster unit using distance measure; call its index j.
Step5: Update weight vector v_j of unit j using:
V_{ij} (new) = V_{ij} (old) + a (x_i-V_{ij} (old)) i=1, 2…, n
Step 6: Reduce learning rate a.
Step 7: Goto step 1.

Phase two {single Layer Linear network}
Step 0: For each single layer linear net of a cluster do.
Step 1: Initialize weight vector w (randomly) and the learning rate a.
Step 2: set E =0 and k=0 (k is a counter for the number of associative examples to the current clusters net)
Step 3: k=k+1
$X=x_k$
$y=y_k$

$$y' = \sum_{m=0}^{n} W_m X_m$$

e= y-y`

$$Wj = Wj + a * \frac{eXj}{|x|^2}$$

$$y' = \sum_{m=0}^{n} W_M X_M$$

E= E+ abc(y-y`)
Step 4: if K< M then Go to 3
Step 5: if E >E_O then Go to 2
Where:
E: The absolute error of current cluster,
E_O: The previous absolute error of the current cluster.
W_J: The jth weight of the processing element.
y`: The actual output of the cluster,
y: The desired output,
X: The input vector.
a: The learning rate, a practical range for the a is 0.1= a =1.0.
M: The number of associated examples with the current clusters net.

PERFORMANCE CONSIDERATIONS

This section is concerned with discussing a number of considerations that were found useful in improving the performance and efficiency of the SLLIC net still further.

Weight initialization: Since a learning algorithm is essentially a search procedure, the success of the search depends to a great extent on its initial state[6]. In many learning algorithm, the initial magnitude of the weights is set randomly (as in the learning algorithm of the SLLIC), but in fact, random weight initialization may lead the algorithm to stuck in local optimum, instead of in global optimum of weight[7]. To overcome such problem, the initial weights magnitude should be limited to a certain rang to allow the SLLIC exhibits well learning behavior.

Linear regression is a statistical technique for investigating the relationship between variables, the parameter estimation problem can be solved by the regression method[8,9] and the initial weights values can be estimated using the following linear regression equation:
$W= (H^T H)^{-1} H^T Y$
Where:
W: is the weight vector
H: is the examples matrix, where each row x_k in H represent an example input vector.
Y: is the desired output vector, where Y_k represents desired output value of example x_k.

Automatic structure determination: A long-term goal of a neural network approach is the automatic optimal design of neural network architecture[3]. In the case of SLLIC net, the architecture specification limited to the problem determining the optimal number of clusters required to solve the problem defined by the training of examples.
In the weight initialization equation:

$$W= (H^T H)^{-1} H^T Y$$

The matrix H is composed of subset of the examples vectors x that are associated with the clusters processing element that we want to find its weight vector W and H is of size m* n: There are two cases to consider:

- m?n: number of examples in H is less than the input vector dimension. In this case, the generated weight vector produces great error ratio, i.e. Y=Hw+? where î is the error vector.
- m=n: number of examples in H equals to the input vector length .In this case, the generated weight vector produces exact solution(theoretically), i.e. Y=Hw and $w= (H^T H)^{-1} H^T Y=H^{-1} Y$.

Thus, is the case similar to case 2 can be reached for each cluster from a theoretically point of view, a SLLIC net with zero error vector can be produced (theoretically). This case can be achieved by clustering the examples vectors into number of clusters where:
If n=0 then # of clusters = (m div n) else # of clusters = (m div n) + 1.

Where, m represents number of examples in the training set and n represents the input vector dimension. The additional cluster in the case m mod n≠0 is for the

remaining examples plus a number of examples from the neighbor cluster that make the matrix H of the added cluster of size n * n.

From the above we can conclude that if the number of examples specified for each cluster equal to n, then a SLLIC neural network that automatically determine number of clusters (structure) with minimum error ratio can be specified.

High-order network capabilities: To make use of high-order neural net abilities in increasing the processing elements interpolation-capabilities, the same concept was applied to the SLLIC network. The application of the high-order neural network concept to SLLIC was accomplished by producing a high-order vector from the original input vector and then supplies the generated high order vector to the network processing elements. In this study, only the second and third orders were implemented. Higher orders can be implemented in the same manner (Fig. 5).

Second-Order Vector Generator Algorithm:
Step 0: c=0 (counter for second-order vector elements)
Step 1: for i=1 to N do
c=c+1
$h_c=x_i$
Step 2: For i=1 to N do
For j =i to N do
c=c+1
$hc=x_i*x_j$
Third-Order Vector Generator Algorithm:
Step 0: c=0(Counter for third-order vector elements)
Step 1: For i=1 to N do
c=c+1
$h_c=x_i$
Step 2: For i=1 to N do
For j =i to N do
c=c+1
$h_c=x_i*x_j$
Step 3: For i=1 to N do
For j=i to N do
For k =j to N DO
c=c+1
$h_c = x_i * x_j * x_k$
Where:
x_i: is the input vector i^{th} element of vector x
h_i: is the ith element of the high-order vector generated
N: is the input vector length.

The above two algorithms produce a high-order vector from the original input vector. The second-order represents the combination of the input vector that first a second-order polynomial, while the third-order vector represents the combinations of the input vector that fits a third-order polynomial.

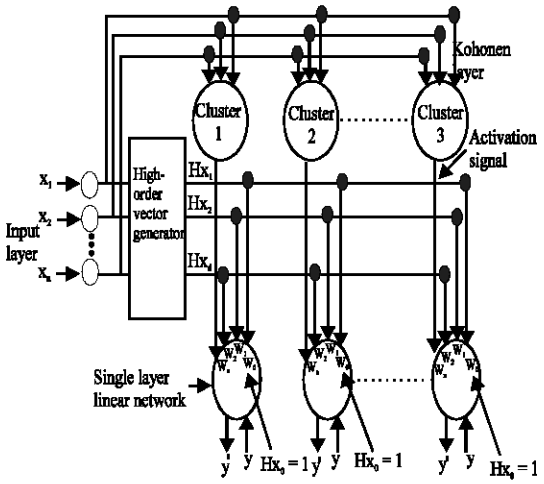

Fig. 5: SLLIC network with high-order vector generator

EVALUATION OF SLLIC NETWORK

To evaluate the performance of the SLLIC network in function approximation, data fitting and predication (forecasting), the network results are compared to the results obtained from the application of the traditional approach to some problem as well as other neural network model.

Comparison with other neural network model: There is a large number of neural network models that can be used to solve the real function approximation problems, in this section we compare the performance of the SLLIC network to: Forward-only countrpropagation, a Radial Basis Function (RBF) network and a back propagation networks.

Forward-only countrpropagation network: Fausett[2] used the function y=1/x for x in the range 0.1-1.0 with 1000 sample points in this range as training examples to construct a forward-only countrpropagation net.

The same function was used to train a SLLIC network with only 50 sample points as training examples. The two nets were then test with 50 values for x obtain the corresponding values of y. The results using mean absolute error criteria were:

- Mean absolute error of forward-only = $1/50\sum_{i=1}^{50}\text{abs}(y-y') = 0.080998$
- Mean absolute error of SLLIC network = $1/50\sum_{i=1}^{50}\text{abs}(y-y') = 0.004710$

Where, y' is the computed value and y is the actual value.

Radial basis function network: A Radial Basis Function network (RBF) is a two-layer network whose output nodes form a linear combination of the basis (or kernal) functions computed by the hidden layer nodes, the RBF network is capable of forming an arbitrarily close approximation to any continuous nonlinear mapping[10].

Sanchez[3] used RBF network in approximating the function:

$$Y=f(x)=\sum_{i=1}^{n} Wi\beta_{piqi}(x)$$

Where,

$$B_{p,p}(x)=\frac{\Gamma(p+q)}{\Gamma(p).\Gamma(q)}x^{p-1}.(1-x)^{q-1}$$

- Γ is the gamma function (i.e.Γ (x) =(x-1)!
- The specific value used is n=3 and
- The parameter values wi, pi, qi are given by Table 1.
- The training data (x_i, y_i), i = 1, 2 …., N where, N = 51 and
- The values x_i is equidistant in the range [0, 1].

Sanchez[3] in constructed an RBF network for the above example with 15 nodes. For comparison purposes, a second order SLLIC network was constructed and trained for the above example with 17 clusters. The mean absolute error of each network is given below:

- Mean Absolute Error of RBF =

$$\frac{1}{50}\sum_{i=1}^{50} abs(y-y')=0.011151$$

- Mean Absolute Error of SLLIC =

$$\frac{1}{50}\sum_{i=1}^{50} abs(y-y')=000002$$

The results clearly show that the overall performance of the SLLIC network is better than that of RBF.

Back propagation network: A back propagation Network was constructed and trained to approximate the following function:

$$Y=\sin(2\pi x_1)\sin(2\pi x_2)$$

A SLLIC network was constructed and trained also for the same function. The results of absolute mean error for test values are:

Table l: RBF parameter values

i	w_i	p_u	q_i
1	0.5	10	30
2	0.2	20	20
3	0.3	30	10

- Mean Absolute Error of Back propagation Net =

$$\frac{1}{36}\sum_{i=1}^{36} abs(y-y')=0.011151$$

- Mean Absolute Error of SLLIC =

$$\frac{1}{36}\sum_{i=1}^{36} abs(y-y')=0.000002$$

CONCLUSIONS

Using neural network approach to solve function approximation, data fitting and forecasting problems instead of traditional methods (i.e. mathematical and statistical methods) is simpler especially for users with poor statistical and mathematical knowledge since the functional form of the relationship between the independent variables and the dependent variable need not be known.

Initializing the weights of a neural network with specific values that are related to the problem instead of using random initial weights will accelerate the convergence of the neural network.

Using high order combinatory of the input values in the transfer function of the processing element (i.e. using high-order neural network concepts) to produce the non linearity is more powerful than using a nonlinear activation function (such as Bipolar, Binary sigmoid and hard limiter) because the high order transfer function represent the natural nonlinearity between the independent variables while the activation functions works as a function that compress the output of the processing element in specific range.

The suggested neural network allows the automatic optimal structure determination that is suitable for a given problem. This feature allows the system to specify the appropriate number of clusters and hence produce more accurate results.

From test experiments, the suggested neural model has comparable and some times better performance than that obtained from linear regression. However, the neural network model suffers from the disadvantage that the true functional relationship (mathematical formula) between the dependent and independent variables can not be explicitly obtained. Understanding how the neural network function may solve this problem.

REFERENCES

1. Fausett, L., 1994. Fundamental of Neural Networks. Prentice Hall International, Inc., pp: 35-46.
2. Hecht-Nielsen R., 1990. Neuro Computing Reading. Addison-Wesley, pp: 38-66.
3. Sanchez, D., 1999. Letters on the number and the distribution of RBF centers. Neuro Comp. Applic. Mag., 22: 52-58.
4. Uros, L. and D. Andrej, 2005. Predicting time series using neural networks with wavelet-based denoising layers. Neural Comp. Applic., 14: 18-24.
5. Widrow, B. and M. Lehr, 1998. Adaptive neural network and their application. Intl. J. Intel. Sys., 8: 34-42.
6. Gutierrez, J. and R. Groding, 1990. Estimating learning parameters for two-layer perception. Neural Networks Mag., 1: 31-35.
7. Kim, D., 2005. Improving prediction performance of neural networks in pattern classification. Intl. J. Comp. Math., 82: 391-399.
8. Cheng-Jian, L. and C. Cheng-Hung, 2005. Identification and prediction using recurrent compensatory neuro-fuzzy systems. Fuzzy Sets Sys., 150: 307-330.
9. Xie, N. and H. Leung, 2005. Blind Equalization using a predictive radial basis function neural network. IEEE Transactions on Neural Networks, 16: 709-720.
10. Hush, D. and B. Horne, 1997. Progress in supervised neural networks: What's New since Lippmann?. IEEE Signal Processing Mag., 12: 46-52.