ISSN 2229-5518

Designing Aspects of Artificial Neural Network Controller

Navita Sajwan, Kumar Rajesh

Abstract— In this paper important fundamental steps in applying artificial neural network in the design of intelligent control systems is discussed. Architecture including single layered and multi layered of neural networks are examined for controls applications. The importance of different learning algorithms for both linear and nonlinear neural networks is developed. The problem of generalization of the neural networks in control systems together with some possible solutions are also included.

Index Terms— Artificial, neural network, adaline algorithm, levenberg gradient, forward propagation, backward propagation, weight update algorithm.

1 INTRODUCTION

The field of intelligent controls has become important due to the development in computing speed, power and affordability. Neural network based control system design has become an important aspect of intelligent control as it can replace mathematical models. It is a distinctive computational paradigm to learn linear or nonlinear mapping from a priori data and knowledge. The models are developed using computer, the control design produces controllers,that can be implemented online.The paper includes both the nonlinear multi-layer feed-forward architecture and the linear single-layer architecture of artificial neural networks for application in control system design. In the nonlinear multi-layer feed-forward case, the two major problems are the long training process and the poor generalization. To overcome these problems, a number of data analysis strategies before training and several improvement generalization techniques are used.

2 ARCHITECTURE IN NEURAL NETWOKS

Depending upon the nature of the problems, design of neural network architecture is selected. There are many commonly used neural network architectures for control system applications such as Perceptron Network, Adaline network, feed forward neural network.

(a)ADALINE Architecture:

ADALINE(For ADAptive LINear combiner) is a device and a new, powerful learning rule called the widrow-Hoff learning rule this is shown in figure1.The rule minimized the summed square error during training involving pattern classification. Early applications of ADALINE and its extension to MADALINE(for many ADALINES) include pattern recognition, weather forecasting and adaptive control

ADALINE algorithm:

1. Randomly choose the value of weights in the range -1 to 1.
2. While stopping condition is false,follow steps 3.
3. For each bipolar training pair S:t, do step 4-7.
4. Select activations to the input units. $X_0=1$, $x_i=s_i(i=1,2.....n)$.
5. Calculate net input or y.
6. update the bias and weights.
   $W_0=w_0(old)+alpha(t-y)$
   $W_{new}=w_i(old)+alpha(t-y)x_i$.
7. If the largest weight change that occurred in step 3 is smaller than a specified value, stop else continue.

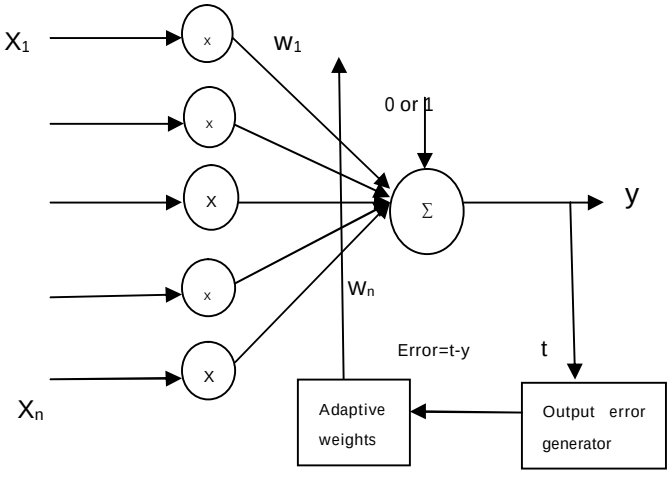

Figure 1:ADALINE Neuron Model

(b)Feed-forward Neural Network Architecture: It is a important architecture due to its non-parametric, non-linear mapping between input and output. Multilayer feed-forward neural networks employing sigoidal hidden unit activations are known as universal approximators.

ISSN 2229-5518

These function can approximate unknown function and its derivative. The feed-forward neural networks include one or more layers of hidden units between the input and output layers.The output of each node propagates from the input to the outside side. Nonlinear activation functions in multiple layers of neurons allows neural network to learn nonlinear and linear relationships between input and output vectors. Each input has an appropriate weighting W.The sum of W and the bias B form the input to the transfer function. Any differentiable activation function f may be used to generate the outputs. Most commonly used activation function are purelin $f(x)=x$, log-sigmoid $f(x)=(1+e^{-x})^{-1}$, and tan-sigmoid

$$f(x)=\tan(x/2)=(1-e^{-x}) / (1+e^{-x})$$

Hyperbolic tangent(Tan-sigmoid) and logistic(log-sigmoid) functions approximate the signum and step functions,respectively,and yet provide smooth, nonzero derivatives with respect to the input signals. These two activation function called sigmoid functions because there S-shaped curves exhibit smoothness and asymptotic properties. The activation function f_h of the hidden units have to be differentiable functions. If f_h is linear, one can always collapse the net to a single layer and thus lose the universal approximation / mapping capabilities. Each unit of the output layer is assumed to have the same activation function.

3 BACK PROPAGATION LEARNING

Error correction learning is most commonly used in neural networks. The technique of back propagation, apply error-correction learning to neural network with hidden layers.It also determine the value of the learning rate, η.Values for η is restricted such that $0<\eta<1$.Back propagation requires a perception neural network, (no interlayer or recurrent connection). Each layer must feed sequentially into the next layer.In this paper only the three-layer, A, B, and C are investigated. Feeding into layer a is the input vector I. Thus layer a has L nodes, a_i (i=1to L), one node for each input parameter. Layer B, the hidden layer, has m nodes, b_j (j =1 to m).L = m = 3; in practice L ≠ m. Each layer may have a different number of nodes. Layer C, the output layer, has n nodes, c_k (k = 1 to n), with one node for each output parameter. The interconnecting weight between the i^{th} node of layer A and the j^{th} node of layer B is denoted as v_{ij}, and that between the j^{th} node of layer B and the k^{th} node of layer C is w_{jk}.Each node has an internal threshold value. For layer A, the threshold is T_{Ai}, for layer B, $T_{Bi,}$ and for layer C, T_{ck}.The Back propagation neural network is shown in figure 2..

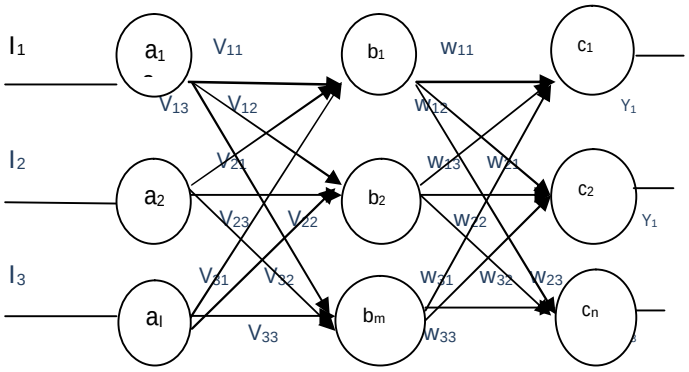

Figure 2: Three-layered artificial neural system

When the network has a group of inputs, the updating of activation values propagates forward from the input neurons, through the hidden layer of neurons to the output neurons that provide the network response. The outputs can be mathematically represented by:

$$Yp = f\left(\sum_{m=-1}^{M} \left((f\sum_{n=-1}^{N} XnWnm)\right)*Kmp\right)$$

Yp = The pth output of the network
Xn = The nth input to the network
Wnm = The mth weight factor applied to the nth input to the network
Kmp = The pth weight factor applied to the mth output of the hidden layer
F( ) = Transfer function (i.e., sigmoid, etc.)
The ANS becomes a powerful tool that can be used to solve difficult process control applications
Figure 3 depicts the designing procedure of Artificial Neural Network Controller.

4 LEARNING ALGORITHMS

A gradient-basedalgorithms necessary in the development of learning algorithms are presented in this section. Learning in neural network is known as learning rule, in which weights of the networks are incrementally adjusted so as to improve a predefined performance measure over time. Learning process is an optimization process, it is a search in the multidimensional parameter (weight) space for solution, Which gradually optimizes an objective(cost)function.

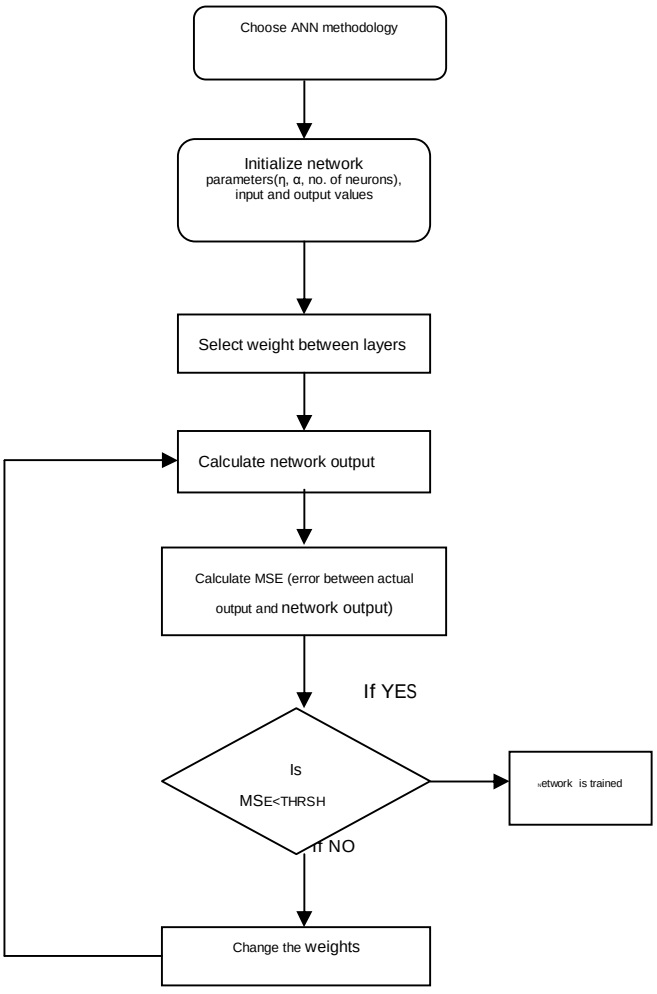

Figure 3: Flow chart for general neural network algorithm

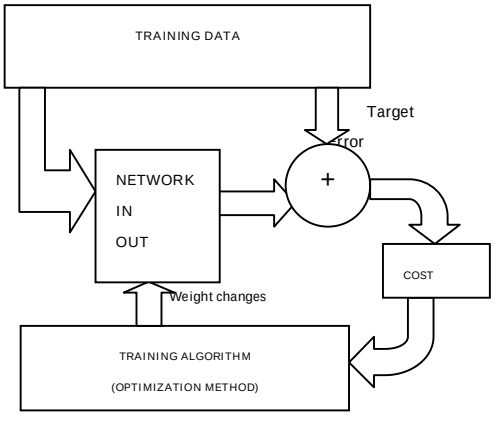

Figure 4: Flow diagram of learning process

5 Levenberg Gradient Based Methods

Gradient based methods searches for minima by comparing values of the objective function $E(\theta)$ at different points.The function is evaluated at points around the current search point and then look for lower values. Objective function $E(\theta)$ is minimized in the adjustable space $\theta = [\theta_1 \theta_{2,\ldots\ldots} \theta_n]$ and to find minimum point $\theta = \theta^*$.A given function E depends on an adjustable parameter θ with a nonlinear objective function. $E(\theta)=f(\theta_1 \theta_{2,\ldots\ldots} \theta_n)$ is so complex that an iterative algorithm is used to search the adjustable parameter space efficiently. The next point θ_{new} is determined by a step down from the current point θ_{now} in the direction vector d given as:

$$\theta_{next} = \theta_{now} + \eta d$$

Where η is some positive step size commonly referred to as the learning rate.

The principal difference between various descent algorithms lie in the first procedure for determining successive directions. After decision is reached, all algorithms call for movement to a minimum point on the line determined by the current point θ_{now} and the direction d.For the second procedure, the optimum step size can be determined by linear minimization as:

$$\eta^* = \arg(\min(\emptyset(\eta)))$$

$$\eta > 0$$

where

$$\emptyset(\eta) = E(\theta_{new} + \eta d).$$

6 Levenberg-Marquardt Method

The Levenberg_Marquardt algorithm can handle ill-conditioned matrices well, like nonquadratic objective functions. Also, if the Hessian matrix is not positive definite, the Newton direction may point towards a local maximum, or a saddle point. The Hessian can be changed by adding a positive definite matrix λI to H in order to make H positive definite.

Thus,

$$\theta_{next} = \theta_{now} - (H + \lambda I)^{-1} g ,$$

where I is the identity matrix and H is the Hessian matrix which is given in terms of Jacobian matrix J as $H = J^T J$. Levenberg-marquardt is the modification of the Gauss-Newton algorithm as

$$\theta_{next} = \theta_{now} - (J^T J)^{-1} J^T r = \theta now - (J^T J + \lambda I)^{-1} J^T r.$$

The Levenberg-Marquardt algorithm performs initially small, but robust steps along the steepest descent direction, and switches to more efficient quadratic Gauss-

Newton steps as the minimum is approached. This method combines the speed of Gauss-Newton with the everywhere convergence of gradient descent, and appears to be fastest for training moderate-sized feedforward neural networks.

7 FORWARD-PROPAGATION AND BACK-PROPAGATION

During training, a forward pass takes place.The network computes an output based on its current inputs.Each node i computes a weighted a_i of its inputs and passes this through a nonlinearity to obtain the node input y_i .The error between actual and desired network outputs is given by

$$E = \frac{1}{2} \sum_p \sum_i (d_{pi} - y_{pi})^2$$

where p indexes the pattern in the training set, I indexes the output nodes, and d_{pi} and y_{pi} are, respectively, the desired target and actual network output for the error with respect to the weights is the sum of the individual pattern errors and is given as

$$\frac{dE}{dW_{ij}} = \sum_p \frac{dE_p}{dW_{ij}} = \sum_{p,k} \frac{dE_p}{da_k} \frac{da_k}{dW_{ij}}$$

where the index k represent all outputs nodes. It is convenient to first calculate a value δ_i for each node i as

$$\delta_i = \frac{dE_p}{da_i} = \sum_k \frac{dE_p}{dy_k} \frac{dy_k}{da_i}$$

which measures the contribution of a_i to the error on the current pattern. For simplicity, pattern index p are omitted on y_i, a_i and other variables in the subsequent equations.

For output nodes, dE_p / da_k, is obtained directly as

$\delta = -(d_{pk} - y_{pk}) f$ (for output node).

The first term in this equation is obtained from error equation, and the second term which is

$$\frac{dy_k}{da_k} = f'(a_k) = f'_k$$

is just the slope of the node nonlinearity as its current value. For hidden nodes, δ_i is obtain indirectly as

$$\delta_i = \frac{dE_p}{da_i} = \sum_k \frac{dE_p}{da_k} \frac{da_k}{da_i} = \sum_p \delta k \frac{da_k}{da_i}$$

where the second factor is obtained by noting that if the node I connects directly to node k then $da_k / da_i = f'_i w_{ki}$, otherwise it is zero. Thus,

$$\delta_i = f'_i \sum_k w_k \delta_k$$

for hidden nodes.δ_i is a weighted sum of the δ_k values of nodes k to which it has connections w_{ki}. The way the nodes are indexed, all delta values can be updated through the nodes in the reverse order. In layered networks, all delta values are first evaluated at the output nodes based on the current pattern errors, the hidden values is then evaluated based on the output delta values, and so on backwards to the input layer. Having obtained the node deltas, it is an easy step to find the partial derivatives dE_p / dW_{ij} with respect to the weights. The second factor is da_k / dw_{ij} because a_k is a linear sum, this is zero if k = i; otherwise

$$\frac{da_i}{dw_{ij}} = X_j$$

The derivative of pattern error E_p with respect to weight w_{ij} is then

$$\frac{dE_p}{dw_{ij}} = \delta_i x_j$$

First the derivative of the network training error with respect to the weights are calculated. Then a training algorithm is performed. This procedure is called back-propagation since the error signals are obtained sequentially from the output layer back to the input layer.

8 WEIGHT UPDATE ALGORITHM

The reason for updating the weights is to decrease the error. The weight update relationship is

$$\Delta wij = \eta dE_p = \eta (d_{pi} - y_{pi}) f_i x_j$$

$$\frac{}{dw_{ij}}$$

where the learning rate η>0 is a small positive constant. Sometimes η is also called the step size parameter.

The Delta Rule is weight update algorithm in the training of neural networks. The algorithm progresses sequentially layer by layer, updating weights as it goes. The update equation is provided by the gradient descent method as

$$\nabla w_{ij} = w_{ij}(k+1) - w_{ij}(k) = -\eta \frac{dE_p}{dw_{ij}}$$

$$\frac{dE_p}{dw_{ij}} = -(d_{ij} - y_{pi}) \frac{dy_{pi}}{dw_{ij}}$$

for linear output unit, where

$$y_{pi} = \sum_i w_{ij} x_i$$

and

$$\frac{dy_{pi}}{dw_{ij}} = x_i$$

so,

$$\nabla w_{ij} = w_{ij}(k+1) - w_{ij}(k) = \eta (d_{pi} - y_{pi}) x_i$$

The adaptation of those weights which connects the input units and the i th output unit is determined by the corresponding error $e_i = 1 \sum (d_{pi} - y_{pi})^2$.

Training Data Analysis
Two training data analysis methods are:
(1)normalizing training set and initializing weights
(2)Principal components analysis(speed up the learning process).

9 CONCLUSION

The fundamentals of neural network based control system design are developed in this paper and are applied to intelligent control of the advanced process. Intelligent control can also be used for fast and complex process control problems.

REFERENCES

[1] Rajesh Kumar, Application of artificial neural network in paper industry, A Ph.D thesis, I.I.T.Roorkee, 2009.
[2] S.I.Amari,N.Murata,K.R.Mullar,M.Fincke,H.H.Yang(1997),Asyptotic Statistical Theory of overtraining and cross validation,IEEE Trans.Neural Networks,8(5),985-993.
[3] C.H.Dagli,M.Akay,O.Ersoy,B.R.Fernandez,A.Smith(1997),Intelligent Engineering Systems Through Artificial Neural Networks,vol.7 of Neural Networks fuzzy logic Data mining evolutionary Programming.
[4] H.Demuth and M.Beale(1997), Neural Networks toolbox user guide,mathworks.
[5] L.Fu, Neural Networks in Computer Intelligence(1994).
[6] M.T.Hang.andM.B.Menhaj(1994),Training feedforward Networks with the Marquardt Algorithm,IEEE Trans. Neural Networks,5(6),989-993.
[7] Hong HelenaMu,Y.P.Kakad,B.G.Sherlock,Application of artificial Neural Networks in the design of control systems.