International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Advanced Neural Network Algorithms for Prediction Applications

Kanchan Bichkule

Pune University, J. S. P. M, Tathawade, Pune, India

Abstract: ***This paper discusses in detail the various advanced neural network algorithms, which are used to solve critical prediction applications such as, predict the traffic on highways or predict number of calls in the call center. Application of neural network for the various prediction application methods to the real life data is challenging due to very large size of the data, high dimensionality and presence of seasonal variations. Predicting the call center traffic or number of vehicles on highways for each time slot for forthcoming day(s) is of immense importance for planning sufficient resources to provide satisfactory and quick customer service. The paper discusses the use of back propagation algorithm for training the neural network using the historic data of the application. Then it discusses the need for improvement in the back propagation algorithm and the use of another algorithm, which is called as back propagation with momentum (BPM) algorithm, which converges faster than back propagation algorithm. The paper also discusses the use of another algorithm called as super self-adapting back propagation (SUPERSAB) as it shows more accurate results than other algorithms for the prediction applications.***

Keywords: Neural Network, Back Propagation, Momentum, Prediction, SuperSAB.

1. Introduction

An artificial Neural Network (ANN) [5] is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain process information. ANNs, like people learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification through learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well. Neural networks are suitable in problems such as pattern classification, associative memories, optimization, vector quantization and control applications, prediction or forecasting including sales forecasting, in speech and image processing applications like character recognition.

There are various advanced neural network algorithms, which are used to train the neural network and then can be used to various applications, like character recognition, predicting the traffic on highways or predicting the communication traffic in call center. These algorithms are Back Propagation, Back Propagation with Momentum (BPM), Quick Propagation, Resilient Propagation, SuperSAB and many more. The most complex problem across all the applications where NN is used is the Prediction, which can be solved with more accuracy using the neural networks than other techniques, let's discuss why Neural Network is widely used for the applications where prediction is required or needed.

2. Neural N/W in Prediction Applications

Neural Networks based prediction using the time series data has emerged as one of the most important and widely used branches of prediction for various applications. Application of neural network prediction methods to real life data is challenging due to very large size of the data, high dimensionality and presence of seasonal variations. Telecommunication is a field where such forecasting models are of great help. Traditional predicting techniques such as moving averages, exponential smoothing, auto regressive integrated moving average (ARIMA) are suitable for forecasting linear applications. Neural networks are useful for nonlinear applications. Prediction is nothing but the forecasting or sense the future. Many forecasting problems exhibit clearly rising trends and on the other hand they show very strong seasonal variations. In addition to these two main attributes, the time series is also affected by further smaller variations, which make it all the more difficult to model. Neural networks are particularly suitable for modeling these not immediately identifiable factors. The aim of the forecasting model is to spot the underlying structure in the data so as to predict future. A neural network is suitable to perform the task much better because of its ability to take into consideration external variables and their non-linear interactions, providing a solution to the solving of complex systems in economics and other fields. The advantage of the method is that neural networks teach themselves the effects of the different variables so that the user isn't required to know the interactions between them. It is for this reason that neural networks are gaining importance in the modeling of economics. A neural network is a computational model that is based on the neuron cell structure of the biological nervous system. Given a training set of data, the neural network can learn the data with a learning algorithm. The training process of the neural network involves adjusting the weights till a desired input/output relationship is obtained. The artificial neural network learns the input/output mapping by a stepwise change of the weights and minimizes the difference between the actual and desired output vector.

This paper presents an application of feed-forward neural networks and the most common algorithm, back propagation. Through back propagation, the neural network forms a mapping between inputs and desired outputs from the training set by altering weighted connections within the network. Artificial neural networks consist of many simple

processing devices (called processing elements or neurons) grouped in layers. Each layer is identified by an index as l=0, 1, . . . , L. The layer l=0 is called as 'input layer' and layer l=L is called as 'output layer'. And all other intermediate layers are called as 'hidden layers'. The processing elements are interconnected as follows: Communication between processing elements is only allowed for processing elements of neighboring layers. Neurons within a layer cannot communicate. Each neuron has a certain activation level 'a'. The network processes data by the exchange of activation levels between connected neurons. The size 's' of the input window corresponds to the number of input units of the neural network. In a forward path, these activation levels are propagated over one hidden layer to one output unit. The error used for the back propagation-learning algorithm is now computed by comparing the value of the output unit with the transformed value of the time series at time t+1. This error is propagated back to the connections between output and hidden layer and to those between hidden and output layer. After all weights have been updated accordingly, one presentation has been completed. Training a neural network with the back propagation algorithm usually requires that all representations of the input set are presented many times. For the learning of time series data, the representations were presented in a randomly manner. Choosing a random location for each representation's input window ensures better network performance and avoids local minima.

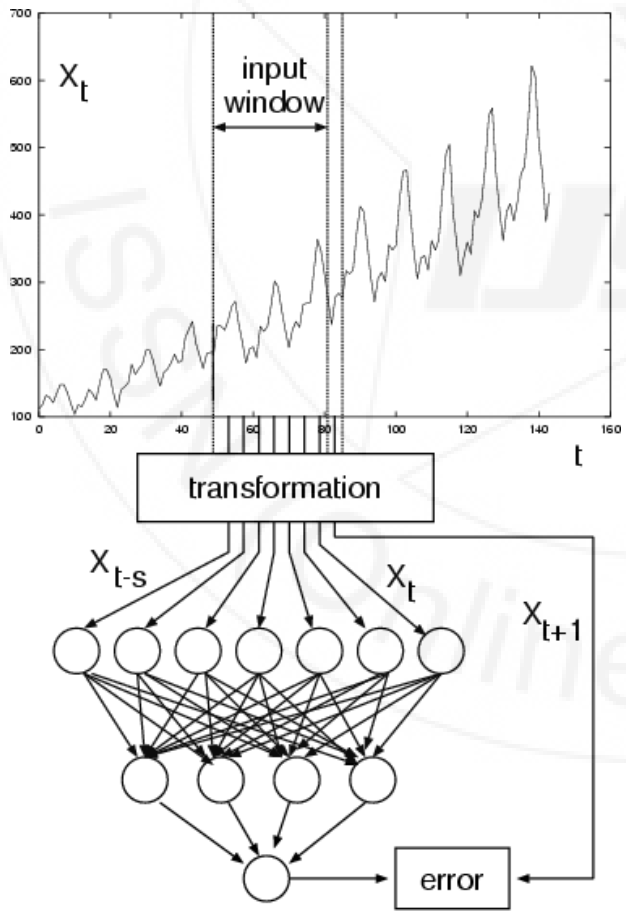

Figure 1: Neural Network for the Prediction Application

The number of input units determines the number of periods the neural network 'looks into the past' when predicting the future. The number of input units is equivalent to the size of the input window. Once the network is trained for given input-output sets, it can be used to forecast or predict the nature of the output for the future period. Figure 1 shows how the prediction data is feed in to the neural network.

One of the most popular neural net paradigms is the feed-forward neural network [1]. In a feed-forward neural network, the neurons are usually arranged in layers [1]. A feed-forward neural net is denoted as $N_I \times N_1 \times \ldots \times N_i \times \ldots N_L \times N_O$.

By convention, the input layer does not count, since the input units are not processing units, they simply pass on the input vector *x*. Units from the hidden layers and output layer are processing units. Figure 2 gives a typical fully connected 2-layer feed-forward network with a 3 × 4 × 3 structure.

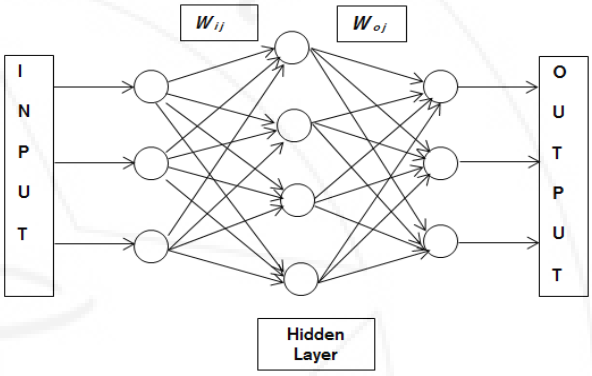

Figure 2: Architecture of Neural Network 3 × 4 × 3

Following are the learning techniques [2] that we discuss in detail in the following sections.
1. Back Propagation
2. Back Propagation with Standard momentum
3. SuperSAB

We will conclude that how Recurrent Neural networks can be used for the feedback connections techniques in the conclusion.

3. Algorithms

We will discuss in detail all the advanced neural network algorithms that can be used for the prediction-based applications. We will start with the basic Neural Network algorithm, i.e. Back Propagation and then how this can be modified in the subsections and which is better algorithm for the prediction based algorithms in this section.

3.1 Back Propagation

The Back propagation algorithm is the most popular supervised learning algorithm for feed-forward neural networks [2]. In this algorithm the minimization of the error function is carried out using a gradient-descent technique. The BP learning method is an iterative process used to train the feed forward neural network for minimal response error. An input pattern is applied to the network and forward propagated through the network using the initial node connection weights. Output error is determined then back

propagated to establish a new set of network connection weights. The process is continued until a prescribed minimum error is achieved. A typical topology for a feed forward network consists of an input layer, hidden layer and output layer. The input layer of the feed forward network is generally fully connected to all nodes in the subsequent hidden layer. Input vectors may be binary, integer or real and are usually normalized to values between -1 and. When inputs are applied to the network, they are scaled by the connecting weights between the input layer and the first hidden layer. These connection weights are denoted by w_{ji}, where i, represents the i^{th} input node and j represents the j^{th} node in the first hidden layer. Each j^{th} node of the hidden layer acts as a summing node for all scaled inputs and as an output activation function. The summing function is given following equation 1.

$$net_j = \sum_i w_{ji} * p_i + b_j \quad (1)$$

where,

- w_{ji} is the connection weights between the j^{th} node of the hidden layer and the i^{th} node of the input layer;
- p_i is the input value at the i^{th} node
- b_j is the bias at the j^{th} node in the hidden layer.

One of the more popular activation functions used in classification problems is the sigmoid function. This function is given in equation 2, where o_j is the output of the j^{th} node. The parameter j serves as a bias and is used to shift the activation function along the horizontal axis. The parameter o is used to modify the shape of the sigmoid function, where low values will make the function look like a threshold-logic unit and high values result in a more gradually sloped function.

$$O_j = \frac{1}{1+e^{-(net_j+\theta_j)/\theta_0}} \quad (2)$$

Subsequent layers work in the same manner. The output from the preceding layer is summed at each node using equation 1. The output of the node is determined using the activation function of equation 2. There is no limit on the number of nodes in a hidden layer, nor is there a limit on the number of hidden layers. However, practical experience shows that excessively large networks are difficult to train and require large amounts of memory and execution time.

The BP training process begins by selecting a set of training input vectors along with their corresponding output vectors. An input vector is applied to a network where some type of random value criteria has initialized connecting weights. The resulting output is determined by calculating the feed forward output at each node and forward propagating the layer results until the k^{th} output layer nodes are activated. The actual output o_k is then compared to the target output t_k and the error is calculated using the sum of square error (SSE), or other selected criteria. The relationship representing the SSE for an input pattern p is given in equation 3.

$$S \quad S_p = \frac{1}{2}\sum_k (t_k - o_k)^2 \quad (3)$$

Once the error has been determined the connection weights in the network are updated using a gradient descent approach, by back propagating change in the network weights starting at the output layer and working back toward the input layer. The incremental change in w_{kj} for a given pattern p is termedΔp w_{kj}, and is proportional to SSE_p. The relationship for calculating the incremental change in connection weights between the last hidden layer and the output layer is given by following equation 4.

$$\Delta_p w_k (m+1) = \eta \quad \delta_j \quad (4)$$

where,

m = Iteration step in the training process

η = Learning rate

and where the value at the η output layer is represented by following equation,

$$\delta_p \quad = (t_p \quad o_p \quad) \quad (1 - o_p \quad) \quad (5)$$

The subscript p represents the pattern number. The values of t_{pk} and o_{pk} represent the target output and the actual output, respectively, of the p_{th} pattern at the k_{th} output node. The incremental change in w_{ji} (between the hidden layer and the input layer) for a given pattern p is termed $\Delta_p w_{ji}$, and is identical to equation 4, with the exception that the delta functions is redefined as given in following equation.

$$\delta_p \quad = o_p \quad (1 - o_p \quad) \sum_k \delta_p \quad w_k \quad (6)$$

In general, the network connection weights are adjusted at the end of each training cycle. That is, after all training patterns have been presented to the network and the $\Delta_p w_{kj}$ calculated, a summation of all pattern changes is applied to the network. This relationship is represented in equation 7,

$$\Delta w_{k\ j} = \sum_p \Delta_p w_{k\ j} \quad (7)$$

Once the network has been updated, the process is repeated until either a specified error limit is achieved or the total numbers of training cycles (epochs) have been completed.

3.2 Back Propagation with Momentum

Back propagation networks are not without problems; however with the most serious being the slow speed of learning. Also, simple back propagation does not scale up very well. The number of training examples required is super linear in the size of the network. So momentum can be used with back propagation algorithm. When using momentum the update rule from Back Propagation is modified to be as below,

$$\Delta_\gamma w_{kj}(m+1) = \eta \delta_k o_j + \alpha \Delta w_{kj}(m) \quad (8)$$

where, α is the momentum.

Momentum in neural networks behaves similar to momentum in physics. In Machine Learning by Tom

Mitchell, momentum is described. To see the effect of this momentum term, consider that the gradient descent Search trajectory is analogous to that of a (momentum less) ball rolling down the error surface. The effect of α is to add momentum that tends to keep the ball rolling down the error surface. In order to speed up training, many researchers augment each weight update based on the previous weight update. This effectively increases the learning rate. Because many updates to a particular weight are in the same direction, adding momentum will typically result in a speedup of training time for many applications. This speedup has been shown to be several orders of magnitude in simpler applications with lower complexity decision boundaries. Even though momentum is a well-known algorithm [2] in the neural network community, there are certain criteria that have been considered when extending momentum. Jacobs enumerates these criteria and analyzes the problems with basic back propagation by giving several heuristics that should be used in neural network design:

- Each weight should have a local learning rate
- Each learning rate should be allowed to vary over time
- Consecutive updates to a weight should increase the weight's learning rate
- Alternating updates to a weight should decrease the weights' learning rate

3.3 SuperSAB (Super Self-Adapting Back propagation)

The SuperSAB algorithm is a local adaptive technique introduced by T. Tollenaere in 1990 [1]. This algorithm uses different learning rate for every connection. The value of each learning rate depends on the value of the gradient of the error function in relation to that weight in the current and past instant of time. For an arbitrary weight Wji the learning rate is defined as,

$$\eta_{ij}(k) = \begin{matrix} \eta_{ij}(k-1) * u, if \frac{\partial E_{(k)}}{\partial W} * \frac{\partial E_{(k-1)}}{\partial W} >= 0 \\ \eta_{ij}(k-1), else \end{matrix}$$

where $u \cong \frac{1}{d}$

SuperSAB [2] combines the momentum and adaptive methods. It uses adaptive method and momentum so long as the sign of the gradient does not change. This is an additive effect of both methods resulting in a faster traversal of gradual slopes. When the sign of the gradient does change the momentum will cancel the drastic drop in learning rate. This allows for the function to roll up the other side of the minimum possibly escaping local minima. These two cases are show by following mathematical formula.

$$W_{ji(k+1)} = \{ \begin{matrix} -\eta_{ij(k)} * \frac{\partial E(k)}{\partial W_{ij}} + \mu * W_{ji(k-1)} if \frac{\partial E(k)}{\partial W} * \frac{\partial E(k-1)}{\partial W} >= 0 \\ W_{ji(k)}, otherwise \end{matrix}$$

SuperSAB has shown to be a fast converging algorithm, which is often considerably faster than ordinary gradient descent. One possible problem of SuperSAB is the large number of parameters that need to be determined in order to achieve good convergence times, namely the initial learning rate, the momentum factor, and the increase (decrease) factors. Another drawback, inherent to all learning-rate adaptation algorithms, is the remaining influence of the size of partial derivative on the weight-step.

4. Conclusion

Back Propagation with momentum (BPM) algorithm converges faster than other algorithms. Also the SUPERSAB algorithm shows more accurate results than other algorithms for the prediction problems. So both the algorithms can be effectively used in forecasting in call center traffic or predicting number of vehicles on highway. In this paper and [1], we have used feed-forward neural networks. Another idea is to use recurrent neural networks, which has feedback connections. These are very interesting for a number of reasons. Biological neural networks are highly recurrently connected, and many authors have studied recurrent network models of various types of perceptual and memory processes. The general property making such networks interesting and potentially useful is that they manifest highly nonlinear dynamical behavior. Hence, using recurrent neural networks could improve our results.

5. Acknowledgments

I would like to thank my guide Dr. P. K. Deshmukh and Prof .S. B. Jahveri for their help and guidance throughout this project and the semester, without them this would not have been possible. I would also like to give special thank to Abhay Khoje, for his constant input to my project.

References

[1] Laurenţiu Leuştean , "*Liquid flow time series prediction using feed-forward neural networks and SuperSAB learning algorithm*", National Institute for Research and Development in Informatics-ICI, Romania.
[2] Christos stergiou, Dimitrios Siganos *Neural Networks, http://www.doc.ic.ac.uk/~nd/surprise-96/journal/vol4/cs11/report.html.*
[3] B. Yegnanarayana, "*Artificial Neural Networks*", PHI, pages 278-280, 1999
[4] Kishan Mehrotra, Chilukari Mohan, Sanjay Ranka, "*Elements of Artificial Neural Networks*", Penram International Publishing (India), pages 70-88, 1997.
[5] Tao Liu, Lieli Liu,"*Research on Forecasting Call Center Traffic through PCA and BP Neural Network*", 2012 Fifth International Symposium on Computational Intelligence and Design.

Author Profile

Kanchan Bichkule, have completed Bachelor of Engineering in Computer Engineering from Maharashtra Institute of Technology, Pune in 2013. She has passed Bachelor of Engineering with first class. Currently, she is pursuing Masters of Engineering in Computer Science and Engineering from J.S.P.M`s Rajarshi Shahu College Of Engineering, Tathawade, Pune. She is doing research on "How to use advanced Neural Network algorithms for prediction applications and compare the performances of the algorithms on these applications.