Survey on Artificial Neural Network Learning Technique Algorithms

K Ishthaq Ahamed[1], Dr. Shaheda Akthar[2]
[1] Faculty, Department of CSE, G. Pulla Reddy Engineering College, Kurnool, AP, India.
[2] Faculty of Computer Science, Govt. College for Women, Guntur, AP, India.

Abstract - ***Automation is now ubiquitous, and research for developing new ones is constantly growing. It plays an important role in decision-making, prediction, classification to provide a fast and accurate result. Due to their adaptive learning and nonlinear mapping properties, the artificial neural networks are widely used to support the human decision capabilities, avoiding inconsistency in practice and errors based on lack of experience. Neural networks is a mathematical model of neurons in human brain, possess all abilities mention above, also they provide parallel computations.***

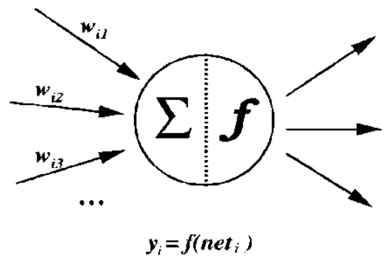
Learning rules are algorithms which direct changes in the weights of the connections in a network. They are incorporating an error reduction procedure by employing the difference between the desired output and an actual output to change its weights during training. The learning rule is typically applied repeatedly to the same set of training inputs across a large number of epochs with error gradually reduced across epochs as the weights are fine-tuned. In this paper error correction, memory based, hebbian and competition learning rules are explored for better predictions and learning.

Key Words: *Artificial neural networks, error correction, memory based, hebbian, and competition learning.*

1. Introduction

Models has constructed by computational neurobiologists by using neurons to simulate the behavior of the brain. As Computer Scientists, we are more interested in the general properties of neural networks, independent of how they are actually "implemented" in the brain. This means that we can use much simpler, abstract "neurons", which capture the essence of neural computation even if they leave out much of the details of how biological neurons work. People have implemented model neurons in hardware as electronic circuits, often integrated on VLSI chips. Remember though that computers run much faster than brains, therefore we can run fairly large networks of simple model neurons as software simulations in reasonable time [10]. This has obvious disadvantages over having to use special "neural" computer hardware.

The basic of neuron model is often known as node or unit. It receives input from some other units, or perhaps from an external source. Each input has an associated weight w, which can be modified so as to model synaptic learning. The unit computes some function f of the weighted sum of its inputs. Its output, in turn, can serve as input to other units. The weighted sum is called the net input to unit i, often written net_i. Weight from unit j to unit i is denoted as w_{ij}. The function f is the unit's activation function. In the simplest case, Fig.1, f is the identity function, and the unit's output is just its net input. This is called a linear unit.



$$y_i = f\left(\sum_j w_{ij} y_j\right)$$

Fig 1: Simple Artificial Neuron Model

Neural Network Applications can be grouped in following categories:

Prediction system
The task is to forecast some future values of a time-sequenced data. Prediction has a significant impact on decision support systems. Prediction differs from function approximation by considering time factor [1][2].

Clustering
A clustering algorithm explores the relationship between patterns and places related patterns in a cluster. Well known applications are data compression and data mining [3].

Classification recognition
Classification is to assign an input pattern (like handwritten symbol) to one of many classes. This category consists of algorithmic implementations like associative memory [5].

Function approximation
The main use of function approximation is to estimate of the unknown function f( ) subject to noise. In engineering various methods require function approximation.

Here the system is dynamic and may produce different results for the same input data based on system state (time).

The primary significance of a neural network is the ability to learn from its environment and to improve its performance through learning [4]. A neural network learns about its environment through an interactive process of adjustments applied to its synaptic weights and bias levels. Ideally, the network becomes more knowledge about its environment after each iteration of the learning process.

We define learning in the context of neural networks as: Learning is a process by which the free parameters of a neural network are adapted through a process of simulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place.

The definition of the learning process implies the following sequence of events [10]:

1. The neural network is simulated by an environment.
2. The neural network undergoes changes in its free parameters as a result of this simulation.
3. The neural network responds in a new way to the environment because of the changes that have occurred in its internal structure.

A prescribed set of well-defined rules for the solution of a learning problem is called a learning algorithm. In this paper, different learning algorithms are discussed and analyzed.

2. Different learning algorithms

2.1 Error-Correction Learning Algorithm

Error-Correction Learning, used with supervised learning. In this the system output is compared with the desired output value, and using that error to direct the training. In the most direct route, the error values can be used to directly adjust the tap weights, using an algorithm such as the backpropagation algorithm. If the system output is *y*, and the desired system output is known to be *d*, the error signal can be defined as:

$$e = d - y$$

It attempts to minimize this error signal at each training iteration. The most popular learning algorithm for use with error-correction learning is the backpropagation algorithm.

Gradient Descent

The gradient descent algorithm is not specifically an ANN learning algorithm. It has a large variety of uses in various fields of science, engineering, and mathematics [8]. The gradient descent algorithm is used to minimize an error function *g(y)*, through the manipulation of a weight vector *w*. The cost function should be a linear combination of the weight vector and an input vector *x*. The algorithm is:

$$w_{ij}[n+1] = w_{ij}[n] + \eta g(w_{ij}[n])$$

Here, η is known as the step-size parameter, and affects the rate of convergence of the algorithm. If the step size is too small, the algorithm will take a long time to converge. If the step size is too large the algorithm might oscillate or diverge.

The gradient descent algorithm works by taking the gradient of the weight space to find the path of steepest descent. By following the path of steepest descent at each iteration, we will either find a minimum, or the algorithm could diverge if the weight space is infinitely decreasing. When a minimum is found, there is no guarantee that it is a global minimum, however.

2.2 Memory Based Learning

In memory-based learning, all (or most) of the past experiences are explicitly stored in a large memory of correctly classified input-output examples

$$\{(x_i,d_i)\}^{N}_{i=1}$$

where x_i denotes an input vector and d_i denotes the corresponding desired response.

When classification of a test vector x_{test} (not seen before) is required, the algorithm responds by retrieving and analyzing the training data in a “local neighbourhood” of x_{test}.

All memory-based learning algorithms [10] involve two factors i.e. criterion used for defining local neighbour of x_{test} and learning rule applied to the training examples in local neighbourhood of x_{test}. Nearest Neighbour Rule (NNR) the vector $X_N \in \{ X_1, X_2, \ldots, X_N\}$ is the nearest neighbour of X_{test} if min d (X_i, X_{test}) = d (X_N, X_{test}) where X_N is the class of X_{test}.

K-Nearest Neighbour rule is the variant of the NNR and it identifies the k classified patterns that lie nearest to X_{test} for some integer k.

2.3 Competitive Learning

It is a form of unsupervised learning in artificial neural networks, in which nodes compete for the right to respond to a subset of the input data. A variant of Hebbian learning [9], competitive learning works by increasing the specialization of each node in the network. It is well suited to finding clusters within data. Models and algorithms based on the principle of competitive learning include vector quantization and self-organizing maps (Kohonen maps).

There are three basic elements to a competitive learning rule:
A set of neurons that are all the same except for some randomly distributed synaptic weights, and which therefore respond differently to a given set of input patterns

- A limit imposed on the "strength" of each neuron
- A mechanism that permits the neurons to compete for the right to respond to a given subset of inputs, such that only one output neuron is active at a time. The neuron that wins the competition is called a "winner-take-all" neuron.

Accordingly the individual neurons of the network learn to specialize on ensemble of similar patterns and in so doing become 'feature detectors' for different classes of input patterns.

The fact that competitive networks Fig.2 record sets of correlated inputs to one of a few output neurons essentially removes the redundancy in representation which is essential part of processing in biological sensory systems.

Architecture

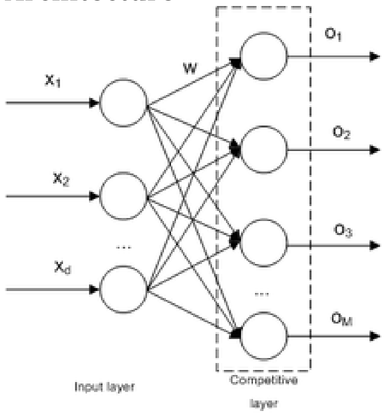


Fig 2: Competitive Neural Network Architecture

It is usually implemented with Neural Networks that contain a hidden layer which is commonly known as "competitive layer". Every competitive neuron i is described by a vector of weights

$\mathbf{w}_i = (w_{i1}, .., w_{id})^T, i = 1, .., M$
and calculates the similarity measure between the input data $\mathbf{x}^n = (x_{n1}, .., x_{nd})^T \in \mathbb{R}^d$ and the weight vector $\mathbf{w}_i$.
For every input vector, the competitive neurons "compete" with each other to see which one of them is the most similar to that particular input vector [8]. The winner neuron m sets its output $o_i = 1$ and all the other competitive neurons set their output $o_i = 0$, i=1…M, i≠m.
Usually, in order to measure similarity the inverse of the Euclidean distance is used: $\|\mathbf{x} - \mathbf{w}_i\|$ between the input vector $\mathbf{x}^n$ and the weight vector $\mathbf{w}_i$.

Algorithm

Here is a simple competitive learning algorithm to find three clusters within some input data.
1. (Set-up.) Let a set of sensors all feed into three different nodes, so that every node is connected to every sensor. Let the weights that each node gives to its sensors be set randomly between 0.0 and 1.0. Let the output of each node be the sum of all its sensors, each sensor's signal strength being multiplied by its weight.
2. When the net is shown an input, the node with the highest output is deemed the winner. The input is classified as being within the cluster corresponding to that node.
3. The winner updates each of its weights, moving weight from the connections that gave it weaker signals to the connections that gave it stronger signals.
Thus, as more data are received, each node converges on the centre of the cluster that it has come to represent and activates more strongly for inputs in this cluster and more weakly for inputs in other clusters.

2.4 Hebbian Learning

Hebbian theory is a theory in neuroscience that proposes an explanation for the adaptation of neurons in the brain during the learning process. It describes a basic mechanism for synaptic plasticity, where an increase in synaptic efficacy arises from the presynaptic cell's repeated and persistent stimulation of the postsynaptic cell. It is also called Hebb's rule.

From the point of view of artificial neurons and artificial neural networks, Hebb's principle can be described as a method of determining how to alter the weights between model neurons. The weight between two neurons increases if the two neurons activate simultaneously, and reduces if they activate separately. Nodes that tend to be either both positive or both negative at the same time have strong positive weights, while those that tend to be opposite have strong negative weights[9].

Hebb's Rule is often generalized as $\Delta w_i = \eta x_i y,$ i,e, the change in the ith synaptic weight w_i is equal to a learning rate η times the ith input x_i times the postsynaptic response y. Often cited is the case of a linear neuron,

$$y = \sum_j w_j x_j,$$

3. CONCLUSIONS

In this paper, we present the objective of artificial neural network and its applications. In this we explore the error correction, memory based, hebbian and competition learning techniques. Each technique has its own advantages and disadvantages based on the type of neural network architecture and type of training. In future work, we design, enhance these techniques and test on real data.

REFERENCES

[1] W. Yan, "Toward automatic time-series forecasting using Neural networks," IEEE Transactions on Neural Networks and Learning Systems, vol.23, no. 7, pp. 1028-1039,2012.
[2] R. R. Andrawis and A. F. Atiya, "A new Bayesian formuation for holt's exponential smoothing," Journal of Foresting, vol.28, no. 3, pp. 218-234, 2009.
[3] I. Maqsood, M. Khan, and A. Abraham, "An ensemble of neural networks for weather forecasting," Neural Computing & Applications, vol. 13, no. 2, pp. 112–122, 2004.
[4] D. F. Specht, "Probabilistic neural networks", Neural Networks, vol.3, PP. 109-118, 1990.
[5] W. T. Miller III, F. H. Glanz and G.Kraft III, "CMAC: an associative neural network alternative to backprogation," Proc.
[6] Shantakumar B. Patil, S. Kumarswamy, Intelligent an and Effective Heart Attack Prediction System Using Data Mining and Artificial Neural Network, European Journal of Scientific Research, ISSN 1450-216X, Vol.31, No.4(2009), PP.642-656.
[7] S. Haykin, Neural Networks A Comprehensive Foundation, 2nd Edition, Prentice Hall, 2000.
[8] Q. Qiu and G. Sapiro. Learning transformations for classification forests. International Conference on Learning Representations, Banff, Canada, 2014.
[9] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets Neural Computation, 18:1527– 1554, 2006.
[10] Walter H. Delashmit and Michael T.Manry, "Recent Developments in Multilayer Perceptron Neural Networks", Proceedings of the 7th Annual Memphis Area Engineering and Science Conference, MAESC 2005, pp. 1-3.