Commun. Fac. Sci. Univ. Ank. Series A2-A3
V.50(1) pp 11-17 (2006)

A BRIEF REVIEW OF FEED-FORWARD NEURAL NETWORKS

MURAT H. SAZLI

Ankara University, Faculty of Engineering, Department of Electronics Engineering
06100 Tandoğan, Ankara, TURKEY
E-mail: sazli@eng.ankara.edu.tr
(Received Jan. 03, 200; Revised: Jan. 27, 2006 Accepted Feb. 06, 2006)

ABSTRACT

Artificial neural networks, or shortly neural networks, find applications in a very wide spectrum. In this paper, following a brief presentation of the basic aspects of feed-forward neural networks, their mostly used learning/training algorithm, the so-called back-propagation algorithm, have been described.

KEYWORDS: Artificial neural networks, feed-forward neural networks, back-propagation algorithm

1. INTRODUCTION

Artificial neural networks, or shortly *neural networks*, have been successfully applied to many diverse fields. Pattern classification/recognition; system modeling and identification; signal processing; image processing; control systems and stock market predictions are some of those main fields of engineering and science [1]. This, of course, can be attributed to the many useful aspects of neural networks, such as their parallel structure, learning and adaptive capabilities, Very Large Scale Integrated (VLSI) implementability, fault tolerance, to name a few [2], [3].

Outline of the paper is as follows. In the next section, a brief introduction on feed-forward neural networks is presented. Feed-forward neural networks are the mostly encountered and used in many diverse applications, therefore they are chosen to exemplify the artificial neural networks. Back-propagation algorithm is described in detail in Section 3. Back-propagation algorithm is the mostly used algorithm in the training of feed-forward neural networks, but it is also used, along with the modified versions of the algorithm, in the training of other types of neural networks.

2. FEED-FORWARD NEURAL NETWORKS

Artificial neural networks, as the name implies, are inspired from their biological counterparts, the biological brain and the nervous system. Biological brain is entirely different than the conventional digital computer in terms of its structure and the way it processes information. In many ways, biological brain (or

human brain as its most perfect example) is far more advanced and superior to conventional computers. The most important distinctive feature of a biological brain

is its ability to "learn" and "adapt", while a conventional computer does not have such abilities. Conventional computers accomplish specific tasks based upon the instructions loaded to them, the so-called "programs" or "software".

Basic building block of neural networks is a "*neuron*". A neuron can be perceived as a processing unit. In a neural network, neurons are connected with each other through "*synaptic weight*"s, or "*weight*"s in short. Each neuron in a network receives "weighted" information via these synaptic connections from the neurons that it is connected to and produces an output by passing the weighted sum of those input signals (either external inputs from the environment or the outputs of other neurons) through an "*activation function*".

There are two main categories of network architectures depending on the type of the connections between the neurons, "*feed-forward neural networks*" and "*recurrent neural networks*". If there is no "feedback" from the outputs of the neurons towards the inputs throughout the network, then the network is referred as a "feed-forward neural network". Otherwise, if there exists such a feedback, i.e. a synaptic connection from the outputs towards the inputs (either their own inputs or the inputs of other neurons), then the network is called a "recurrent neural network". Usually, neural networks are arranged in the form of "*layer*"s. Feed-forward neural networks fall into two categories depending on the number of the layers, either "*single layer*" or "*multi-layer*".

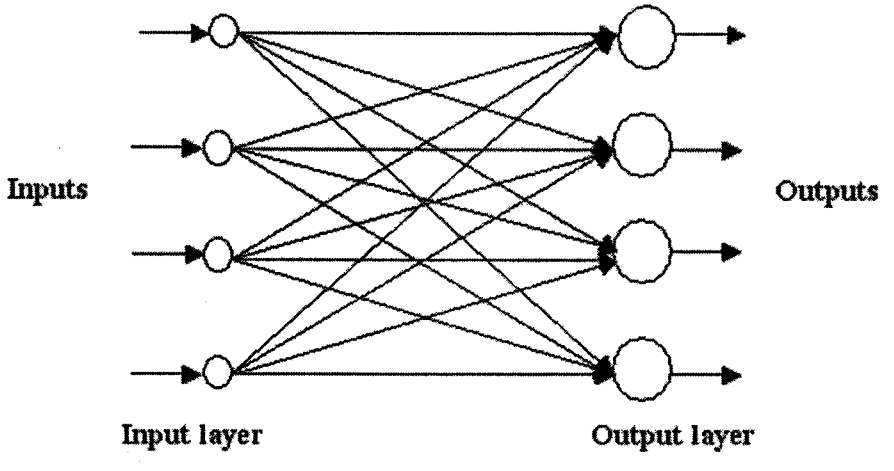

Figure 1. A single layer feed-forward neural network

In Figure 1, a single layer feed-forward neural network (fully connected) is shown. Including the input layer, there are two layers in this structure. However, input layer does not count because there is no computation performed in that layer. Input signals are passed on to the output layer via the weights and the neurons in the output layer compute the output signals.

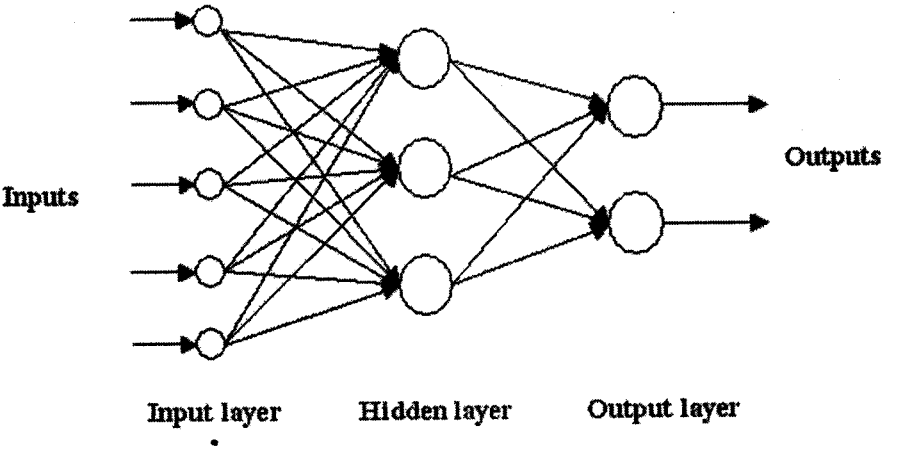

Figure 2. A multi-layer feed-forward neural network

In Figure 2, a multi-layer feed-forward neural network with one "***hidden layer***" is depicted. As opposed to a single-layer network, there is (at least) one layer of "***hidden neurons***" between the input and output layers. According to Haykin [1], the function of hidden neurons is to intervene between the external input and the network output in some useful manner. Existence of one or more hidden layers enable the network to extract higher-order statistics. For the example given in Figure 2, there is only one hidden layer and the network is referred as a 5-3-2 network because there are 5 input neurons, 3 hidden neurons, and 2 output neurons.

In both Figure 1 and Figure 2, networks are "*fully connected*" because every neuron in each layer is connected to every other neuron in the next forward layer. If some of the synaptic connections were missing, the network would be called as "*partially connected*".

The most important feature of a neural network that distinguishes it from a conventional computer is its "*learning*" capability. A neural network can *learn* from its environment and improve its performance through learning. Haykin defined learning in the context of neural networks in the literature [1] as follows:

"Learning is a process by which the free parameters of a neural network are adapted through a process of stimulation 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 [1]."

3. BACK-PROPAGATION ALGORITHM

Among many other learning algorithms, "***back-propagation algorithm***" is the most popular and the mostly used one for the training of feed-forward neural networks. It is, in essence, a means of updating networks synaptic weights by back propagating a gradient vector in which each element is defined as the derivative of an error measure with respect to a parameter. Error signals are usually defined as the difference of the actual network outputs and the desired outputs. Therefore, a set of desired outputs must be available for training. For that reason, back-propagation is a supervised learning rule. A brief explanation of the back-propagation algorithm to train a feedforward neural network is presented in the following.

Let us consider a multilayer feedforward neural network as shown in Figure 2. Let us take a neuron in output layer and call it neuron j. The error signal at the output of the neuron j for *n*th iteration is defined by Equation (1) in the literature [1]:

$$e_j(n) = d_j - y_j(n) \tag{1}$$

where d_j is the desired output for neuron j and $y_j(n)$ is the actual output for neuron j calculated by using the current weights of the network at iteration n. For a certain input there is a certain desired output, which the network is expected to produce. Presentation of each training example from the training set is defined as an "iteration".

Instantenous value of the error energy for the neuron j is given in Equation (2):

$$\varepsilon_j(n) = \frac{1}{2} e_j^2(n) \tag{2}$$

Since the only visible neurons are the ones in the output layer, error signals for those neurons can be directly calculated. Hence, the instantaneous value, $\varepsilon(n)$, of the total error energy is the sum of all $\varepsilon_j(n)$'s for all neurons in the output layer, as given in Equation (3):

$$\varepsilon(n) = \frac{1}{2} \sum_{j \in Q} e_j^2(n) \tag{3}$$

where Q is the set of all neurons in the output layer.

Suppose there are N patterns (examples) in the training set. The average squared energy for the network is found by Equation (4):

$$\varepsilon_{av} = \frac{1}{N}\sum_{n=1}^{N}\varepsilon(n) \tag{4}$$

It is important to note that the instantaneous error energy $\varepsilon(n)$ and therefore the average error energy, ε_{av}, is a function of all the free parameters (i.e., synaptic weights and bias levels) of the network. Back-propagation algorithm, as explained in the following, provides the means to adjust the free parameters of the network to minimize the average error energy, ε_{av}. There are two different modes of back-propagation algorithm: "***sequential mode***" and "***batch mode***". In sequential mode, weight updates are performed after the presentation of each training example. One complete presentation of the training set is called an "*epoch*". In the batch mode, weight updates are performed after the presentation of all training examples, i.e. after an epoch is completed. Sequential mode is also referred as *on-line, pattern or stochastic mode*. This is the most frequently used mode of operation and explained in the following.
Let us start by giving the output expression for the neuron j in Equation (5):

$$y_j(n) = f\left(\sum_{i=0}^{m} w_{ji}(n)y_i(n)\right) \tag{5}$$

where m is the total number of inputs to the neuron j (excluding the bias) from the previous layer and f is the activation function used in the neuron j, which is some nonlinear function. Here w_{j0} equals the bias b_j applied to the neuron j and it corresponds to the fixed input $y_0 = +1$.

The weight updates to be applied to the weights of the neuron j is proportional to the partial derivative of the instantaneous error energy $\varepsilon(n)$ with respect to the corresponding weight, i.e. $\partial\varepsilon(n)/\partial w_{ji}(n)$, and using the chain rule of calculus it can be expressed in Equation (6):

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = \frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial w_{ji}(n)} \tag{6}$$

From Equations (2), (1) and (5) respectively, Equation (7) is obtained.

$$\frac{\partial \varepsilon(n)}{\partial e_j(n)} = e_j(n) \tag{7}$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{8}$$

$$\begin{aligned} \frac{\partial y_j(n)}{\partial w_{ji}(n)} &= f'\left(\sum_{i=0}^{m} w_{ji}(n) y_i(n)\right) \frac{\partial\left(\sum_{i=0}^{m} w_{ji}(n) y_i(n)\right)}{\partial w_{ji}(n)} \\ &= f'\left(\sum_{i=0}^{m} w_{ji}(n) y_i(n)\right) y_i(n) \end{aligned} \tag{9}$$

where

$$f'\left(\sum_{i=0}^{m} w_{ji}(n) y_i(n)\right) = \frac{\partial f\left(\sum_{i=0}^{m} w_{ji}(n) y_i(n)\right)}{\partial\left(\sum_{i=0}^{m} w_{ji}(n) y_i(n)\right)} \tag{10}$$

Substituting Equations (7), (8) and (9) in Equation (6) yields Equation (11).

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = -e_j(n) f'\left(\sum_{i=0}^{m} w_{ji}(n) y_i(n)\right) y_j(n) \tag{11}$$

The correction $\Delta w_{ji}(n)$ applied to $w_{ji}(n)$ is defined by the delta rule, given in Equation (12).

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} \tag{12}$$

In Equation (12), η corresponds to the learning-rate parameter of the back-propagation algorithm, which is usually set to a pre-determined value and kept constant during the operation of the algorithm.

4. CONCLUSIONS

Feed-forward neural networks are the mostly encountered type of artifical neural networks and applied to many diverse fields. In this paper, a brief introduction on artificial neural networks, following a presentation on feed-forward neural networks, have been given. A detailed description of back-propagation algorithm, the mostly used learning/training algorithm of feed-forward neural networks, have been presented in the paper, as well. Interested reader is referred to the literature [1] for a thorough discussion of the artificial neural networks and the back-propagation algorithm.

ÖZET

Yapay sinir ağları, kısaca sinir ağları, çok geniş bir spektrumda uygulama alanları bulmaktadır. Bu makalede, ileri-beslemeli sinir ağlarının temel özelliklerinin kısa bir tanıtımını takiben, bu tür sinir ağlarında en yaygın kullanılan öğrenme/eğitme algoritması olan geri-yayınım algoritması tarif edilmektedir.

ANAHTAR KELİMELER: Yapay sinir ağları, ileri-beslemeli sinir ağları, geri-yayınım algoritması

REFERENCES

[1] S. Haykin. "Neural Networks, A Comprehensive foundation", 2nd edition. Prentice Hall, 1999.

[2] M. H. Sazlı. "Neural Network Applications to Turbo Decoding". Ph.D. Dissertation, Syracuse University, 2003.

[3] M. H. Sazlı, C. Işık. "Neural Network Implementation of the BCJR Algorithm". accepted for publication in Digital Signal Processing Journal, Elsevier, 2005.