International Journal of Engineering Research & Technology (IJERT)
NCAEM-2013 Conference Proceedings
ISBN: 978-93-83758-09-8

Application ANN: A Review

Anjali
M.tech Scholar
anjalisangwan15@gmail.com

Abstract-- Artificial Neural Networks (ANNs) are non-linear mapping structures based on the function of the human brain. They are powerful tools for modelling, especially when the underlying data relationship is unknown. ANNs can identify and learn correlated patterns between input data sets and corresponding target values. After training, ANNs can be used to predict the outcome of new independent input data. ANNs imitate the learning process of the human brain and can process problems involving non-linear and complex data even if the data are imprecise and noisy. These networks are "neural" in the sense that they may have been inspired by neuroscience but not necessarily because they are faithful models of biological neural or cognitive phenomena. In fact majority of the network are more closely related to traditional mathematical and/or statistical models such as non-parametric pattern classifiers, clustering algorithms, nonlinear filters, and statistical regression models than they are to neurobiology models. ANNs have been used for a wide variety of applications where statistical methods are traditionally employed. The problems which were normally solved through classical statistical methods, such as discriminant analysis, logistic regression, Bayes analysis, multiple regression, and ARIMA time-series models are being tackled by ANNs. It is, therefore, time to recognize ANN as a powerful tool for data analysis.

I. Introduction

ANNs imitate the learning process of the human brain and can process problems involving non-linear and complex data even if the data are imprecise and noisy. Thus they are ideally suited for the modeling of agricultural data which are known to be complex and often non-linear. ANNs has great capacity in predictive modeling i.e., all the characters describing the unknown situation can be presented to the trained ANNs, and then prediction of agricultural systems is guaranteed.

An ANN is a computational structure that is inspired by observed process in natural networks of biological neurons in the brain. It consists of simple computational units called neurons, which are highly interconnected. ANNs have become the focus of much attention, largely because of their wide range of applicability and the ease with which they can treat complicated problems. ANNs are parallel computational models comprised of densely interconnected adaptive processing units. These networks are fine-grained parallel implementations of nonlinear static or dynamic systems. A very important feature of these networks is their adaptive nature, where "learning by example" replaces "programming"in solving problems. This feature makes such computational models very appealing in application domains where one has little or incomplete understanding of the problem to be solved but where training data is readily available. ANNs are now being increasingly recognized in the area of classification and prediction, where regression model and other related statistical techniques have traditionally been employed. The most widely used learning algorithm in an ANN is the Backpropagation algorithm. There are various types of ANNs like Multilayered Perceptron, Radial Basis Function and Kohonen networks.

II. Technical Work Preparation

A. Networks

One efficient way of solving complex problems is following the lemma "divide and conquer". A complex system may be decomposed into simpler elements, in order to be able to understand it. Also simple elements may be gathered to produce a complex system (Bar Yam, 1997). Networks are one approach for achieving this. There are a large number of different types of networks, but they all are characterized by the following components: a set of nodes, and connections between nodes.The nodes can be seen as computational units. They receive inputs, and process them to obtain an output. This processing might be very simple (such as summing the inputs), or quite complex (a node might contain another network...). The connections determine the information flow between nodes. They can be unidirectional, when the information flows only in one sense, and bidirectional, when the information flows in either sense. The interactions of nodes though the connections lead to a global behaviour of the network, which cannot be observed in the elements of the network. This global behaviour is said to be *emergent*. This means that the abilities of the network supercede the ones of its elements, making networks a very powerful tool.

Networks are used to model a wide range of phenomena in physics, computer science, biochemistry, ethology, mathematics, sociology, economics, telecommunications, and many other areas. This is because many systems can be seen as a network: proteins, computers, communities, etc. Which other systems could you see as a network? Why?

B. Artificial neural networks

One type of network sees the nodes as 'artificial neurons'. These are called artificial neural networks (ANNs). An artificial neuron is a computational model inspired in the

ISBN: 978-93-83758-09-8

natural neurons. Natural neurons receive signals through *synapses* located on the dendrites or membrane of the neuron. When the signals received are strong enough (surpass a certain *threshold*), the neuron is *activated* and emits a signal though the *axon*. This signal might be sent to another synapse, and might activate other neurons.

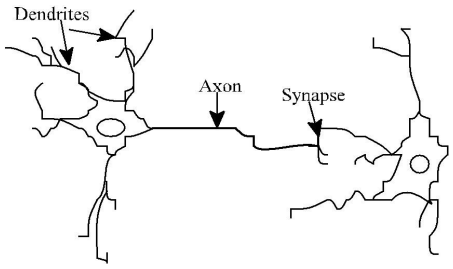


Figure1. Natural neurons (artist's conception).

The complexity of real neurons is highly abstracted when modelling artificial neurons. These basically consist of *inputs* (like synapses), which are multiplied by *weights* (strength of the respective signals), and then computed by a mathematical function which determines the *activation* of the neuron. Another function (which may be the identity) computes the *output* of the artificial neuron (sometimes in dependance of a certain *threshold*). ANNs combine artificial neurons in order to process information.

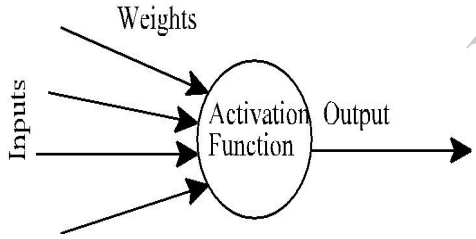


Figure2. An artificial neuron

The higher a weight of an artificial neuron is, the stronger the input which is multiplied by it will be. Weights can also be negative, so we can say that the signal is *inhibited* by the negative weight. Depending on the weights, the computation of the neuron will be different. By adjusting the weights of an artificial neuron we can obtain the output we want for specific inputs. But when we have an ANN of hundreds or thousands of neurons, it would be quite complicated to find by hand all the necessary weights. But we can find algorithms which can adjust the weights of the ANN in order to obtain the desired output from the network. This process of adjusting the weights is called *learning* or *training*.

The number of types of ANNs and their uses is very high. Since the first neural model by McCulloch and Pitts (1943) there have been developed hundreds of different models considered as ANNs. The differences in them might be the functions, the accepted values, the topology, the learning algorithms, etc. Also there are many hybrid models where each neuron has more properties than the ones we are reviewing here. Because of matters of space, we will present only an ANN which learns using the backpropagation algorithm (Rumelhart and McClelland, 1986) for learning the appropriate weights, since it is one of the most common models used in ANNs, and many others are based on it.

C. Development of an ANN Model

Development of ANN model is discussed here briefly. ANNs are constructed with layers of units, and thus are termed multilayer ANNs. A layer of units in such an ANN is composed of units that perform similar tasks. First layer of a multilayer ANN consists of input units. These units are known as independent variables in statistical literature. Lastlayer contains output units. In statistical nomenclature, these units are known as dependent or response variables. All other units in the model are called hidden units and constitute hidden layers. There are two functions governing the behaviour of a unit in a particular layer, which normally are the same for all units within the whole ANN, i.e.

- the input function, and
- the output/activation function.

Input into a node is a weighted sum of outputs from nodes connected to it. The input function is normally given by equation (1) as follows:

$$neti = \sum_{j} WijXj + \mu i$$

where neti describes the result of the net inputs xi (weighted by the weights Wij) impacting on unit i. Also, Wij are weights connecting neuron j to neuron i, Xj is output from unit j and μi is a threshold for neuron i. Threshold term is baseline input to a node in absence of any other inputs. If a weight Wij is negative, it is termed inhibitory because it decreases net input, otherwise it is called excitatory.

Each unit takes its net input and applies an activation function to it. For example, output of jth unit, also called activation value of the unit, is g(∑ WXji i , where g(.) is activation function and xi is output of ith unit connected to unit j. A number of nonlinear functions have been used in the literature as activation functions. The threshold function is useful in situations where the inputs and outputs are binary encoded. However, most common choice is sigmoid functions, such as

$$g(netinput) = [1+e^{-netinput}]^{-1}$$

or

$$g(netinput) = \tanh(netinput)$$

The activation function exhibits a great variety, and has the biggest impact on behavior and performance of the ANN. The main task of the activation function is to map the outlying values of the obtained neural input back to a bounded interval

ISBN: 978-93-83758-09-8

such as [0, 1] or [– 1, 1]. The sigmoid function has some advantages, due to its differentiability within the context of finding a steepest descent gradient for the backpropagation method and moreover maps a wide domain of values into the interval [0, 1].

The various steps in developing a neural network forecasting model are:

Variable Selection
The input variables important for modeling/ forecasting variable(s) under study are selected by suitable variable selection procedures.

Formation of Training, Testing and Validation Sets
The data set is divided into three distinct sets called training, testing and validation sets.
The training set is the largest set and is used by neural network to learn patterns present in the data. The testing set is used to evaluate the generalization ability of a supposedly trained network. A final check on the performance of the trained network is made using validation set.

Neural Network Architecture
Neural network architecture defines its structure including number of hidden layers, number of hidden nodes and number of output nodes etc.

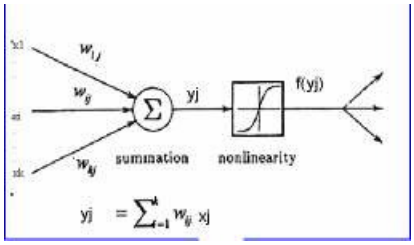
(i)Number of hidden layers: The hidden layer(s) provide the network with its ability to generalize. In theory, a neural network with one hidden layer with a sufficient number of hidden neurons is capable of approximating any continuous function. In practice, neural network with one and occasionally two hidden layers are widely used and have to perform very well.

(ii) Number of hidden nodes: There is no magic formula for selecting the optimum number of hidden neurons. However, some thumb rules are available for calculating number of hidden neurons. A rough approximation can be obtained by the geometric pyramid rule proposed by Masters (1993). For a three layer network with n input and m output neurons, the hidden layer would have sqrt(n*m) neurons.

(iii) Number of output nodes: Neural networks with multiple outputs, especiall y if these outputs are widely spaced, will produce inferior results as compared to a network with a single output.

(iv) Activation function: Activation functions are mathematical formulae that determine the output of a processing node. Each unit takes its net input and applies an activation function to it. Non linear functions have been used as activation functions such as logistic, tanh etc. The purpose of the transfer function is to prevent output from reaching very large value which can 'paralyze' neural networks and thereby inhibit training. Transfer functions such as sigmoid are commonly used because they are nonlinear and continuously differentiable which are desirable for network learning.

Model Building
Multilayer feed forward neural network or multi layer perceptron (MLP), is very popular
and is used more than other neural network type for a wide variety of tasks. Multilayer feed forward neural network learned by back propagation algorithm is based on supervised procedure, i.e., the network constructs a model based on examples of data with known output. It has to build the model up solely from the examples presented, which are together assumed to implicitly contain the information necessary to establish the relation.



An MLP is a powerful system, often capable of modeling complex, relationships between variables. It allows prediction of an output object for a given input object. The architecture of MLP is a layered feedforward neural network in which the non-linear elements (neurons) are arranged in successive layers, and the information flow uni-directionally from input layer to output layer through hidden layer(s).
The characteristics of Multilayer Perceptron are as follows:

(i) has any number of inputs

(ii) has one or more hidden layers with any number of nodes. The internal layers are called "hidden" because they only receive internal input (input from other processing units) and produce internal output (output to other processing units). Consequently, they are hidden from the output world.

(iii) uses linear combination function in the hidden and output layers

(iv) uses generally sigmoid activation function in the hidden layers

(v) has any number of outputs with any activation function.

(vi) has connections between the input layer and the first hidden layer, between the hidden layers, and between the last hidden layer and the output layer.

An MLP with just one hidden layer can learn to approximate virtually any function to any degree of accuracy. For this reason MLPs are known as universal approximates and can be used when we have litter prior knowledge of the relationship between input and targets. One hidden layer is always sufficient provided we have enough data. Schematic representation of neural network and mathematical representation of neural network .

ISBN: 978-93-83758-09-8

Each interconnection in an ANN has a strength that is expressed by a number referred to as weight. This is accomplished by adjusting the weights of given interconnection according to some learning algorithm. Learning methods in neural networks can be broadly classified into three basic types (i) supervised learning (ii) unsupervised learning and (iii) reinforced learning. In MLP, the supervised learning will be used for adjusting the weights. The graphic representation of this learning.

*D.*Architecture of Neural Network

There are several types of architecture of ANN. However, the two most widely used ANN are discussed below:

Feed forward Networks
Feedforward ANNs allow signals to travel one way only; from input to output. There is no
feedback (loops) i.e. the output of any layer does not affect that same layer. They are extensively used in pattern recognition.

Feedback/Recurrent Networks
Feedback networks can have signals traveling in both directions by introducing loops in the network. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found.

E. Supervised and Unsupervised Learning
Learning law describes the weight vector for the ith processing unit at time instant (t+1) in terms of the weight vector at time instant (t) as follows:
w (t + 1) = w (t) + Δwi(t)

where Δwi(t) is the change in the weight vector. The network adapts as follows: change the weight by an amount proportional to the difference between the desired output and the actual output. As an equation:
Δ Wi = η * (D-Y).Ii

where η is the learning rate, D is the desired output, Y is the actual output, and Ii is the ith input. This is called the Perceptron Learning Rule. The weights in an ANN, similar to coefficients in a regression model, are adjusted to solve the problem presented to ANN. Learning or training is term used to describe process of finding values of these weights. Two types of learning with ANN are supervised and unsupervised learning. Supervised learning which incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. An important issue concerning supervised learning is the problem of error convergence, i.e. the minimization of error between the desired and computed unit values. The aim is to determine a set of weights which minimizes the error. Unsupervised learning uses no external teacher and is based upon only local information. We say that a neural network learns off-line if the learning phase and the operation phase are distinct. A neural network learns on-line if it learns and operates at the same time. Usually, supervised learning is performed off-line, whereas unsupervised learning is performed on-line.

E. The Backpropagation Algorithm
The backpropagation algorithm (Rumelhart and McClelland, 1986) is used in layered feed-forward ANNs. This means that the artificial neurons are organized in layers, and send their signals "forward", and then the errors are propagated backwards. The network receives inputs by neurons in the *input layer*, and the output of the network is given by the neurons on an *output layer*. There may be one or more intermediate *hidden layers*. The backpropagation algorithm uses supervised learning, which means that we provide the algorithm with examples of the inputs and outputs we want the network to compute, and then the error (difference between actual and expected results) is calculated. The idea of the backpropagation algorithm is to reduce this error, until the ANN *learns* the training data. The training begins with random weights, and the goal is to adjust them so that the error will be minimal.

The activation function of the artificial neurons in ANNs implementing the backpropagation algorithm is a weighted sum (the sum of the inputs x_i multiplied by their respective weights w_{ji}):

$$A_j(\bar{x},\bar{w}) = \sum_{i=0}^{n} x_i w_{ji} \qquad (1)$$

We can see that the activation depends only on the inputs and the weights. If the output function would be the identity (output=activation), then the neuron would be called linear. But these have severe limitations. The most common output function is the sigmoidal function:

$$O_j(\bar{x},\bar{w}) = \frac{1}{1+e^{A(\bar{x},\bar{w})}} \qquad (2)$$

$$E_j(\bar{x},\bar{w},d) = \left(O_j(\bar{x},\bar{w}) - d_j\right)^2 \qquad (3)$$

We take the square of the difference between the output and the desired target because it will be always positive, and because it will be greater if the difference is big, and lesser if the difference is small. The error of the network will simply be the sum of the errors of all the neurons in the output layer:

$$E(\bar{x},\bar{w},\bar{d}) = \sum_j \left(O_j(\bar{x},\bar{w}) - d_j\right)^2 \qquad (4)$$

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}}$$

ISBN: 978-93-83758-09-8

This formula can be interpreted in the following way: the adjustment of each weight ()w_{ji}) will be the negative of a constant eta (0) multiplied by the dependance of the previous weight on the error of the network, which is the derivative of E in respect to w_i. The size of the adjustment will depend on 0, and on the contribution of the weight to the error of the function. This is, if the weight contributes a lot to the error, the adjustment will be greater than if it contributes in a smaller amount. (5) is used until we find appropriate weights (the error is minimal). If you do not know derivatives, don't worry, you can see them now as functions that we will replace right away with algebraic expressions. If you understand derivatives, derive the expressions yourself and compare your results with the ones presented here. If you are searching for a mathematical proof of the backpropagation algorithm, you are advised to check it in the suggested reading, since this is out of the scope of this material.

So, we "only" need to find the derivative of E in respect to w_{ji}. This is the goal of the backpropagation algorithm, since we need to achieve this backwards. First, we need to calculate how much the error depends on the output, which is the derivative of E in respect to O_j (from (3)).

$$\frac{\partial E}{\partial O_j} = 2\left(O_j - d_j\right) \tag{6}$$

And then, how much the output depends on the activation, which in turn depends on the weights (from (1) and (2)):

$$\frac{\partial O_j}{\partial w_{ji}} = \frac{\partial O_j}{\partial A_j}\frac{\partial A_j}{\partial w_{ji}} = O_j(1-O_j)x_i \tag{7}$$

And we can see that (from (6) and (7)):

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial O_j}\frac{\partial O_j}{\partial w_{ji}} = 2\left(O_j - d_j\right)O_j(1-O_j)x_i$$

And so, the adjustment to each weight will be (from (5) and (8)):

$$\Delta v_{ik} = -\eta\frac{\partial E}{\partial v_{ik}} = -\eta\frac{\partial E}{\partial x_i}\frac{\partial x_i}{\partial v_{ik}}$$

$$\Delta w_{ji} = -2\eta\left(O_j - d_j\right)O_j(1-O_j)x_i \tag{9}$$

We can use (9) as it is for training an ANN with two layers. Now, for training the network with one more layer we need to make some considerations. If we want to adjust the weights (let's call them v_{ik}) of a previous layer, we need first to calculate how the error depends not on the weight, but in the input from the previous layer. This is easy, we would just need to change x_i with w_{ji} in (7), (8), and (9). But we also need to see how the error of the network depends on the adjustment of v_{ik}.

Where:

$$\frac{\partial E}{\partial w_{ji}} = 2\left(O_j - d_j\right)O_j(1-O_j)w_{ji}$$

$$\Delta v_{ik} = -\eta\frac{\partial E}{\partial v_{ik}} = -\eta\frac{\partial E}{\partial x_i}\frac{\partial x_i}{\partial v_{ik}}$$

And, assuming that there are inputs u_k into the neuron with v_{ik} (from (7)):

$$\frac{\partial x_i}{\partial v_{ik}} = x_i(1-x_i)v_{ik} \tag{12}$$

If we want to add yet another layer, we can do the same, calculating how the error depends on the inputs and weights of the first layer. We should just be careful with the indexes, since each layer can have a different number of neurons, and we should not confuse them. For practical reasons, ANNs implementing the backpropagation algorithm do not have too many layers, since the time for training the networks grows exponentially. Also, there are refinements to the backpropagation algorithm which allow a faster learning.

III. CONCLUSION

The computing world has a lot to gain from neural networks. Their ability to learn by example V-47 Artificial Neural Networks and its Applications makes them very flexible and powerful. A large number of claims have been made about the modeling capabilities of neural networks, some exaggerated and some justified. Hence, to best utilize ANNs for different problems, it is essential to understand the potential as well as limitations of neural networks. For some tasks, neural networks will never replace conventional methods, but for a growing list of applications, the neural architecture will provide either an alternative or a complement to these existing techniques. Finally, I would like to state that even though neural networks have a huge potential we will only get the best of them when they are integrated with Artificial Intelligence, Fuzzy Logic and related subjects.

REFERENCES

[1]. Bar-Yam, Y. (1997). *Dynamics of Complex Systems*. Addison-Wesley. Kauffman, S. (1993) *Origins of Order*, Oxford University Press.
[2]. McCulloch, W. and W. Pitts (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133.
[3]. Rojas, R. (1996). *Neural Networks: A Systematic Introduction.* Springer, Berlin.
[4]. Rumelhart, D. and J. McClelland (1986). *Parallel Distributed Processing*. MIT Press, Cambridge, Mass.
[5]. Young, D. *Formal Computational Skills Course Notes*. Http://www.cogs.susx.ac.uk/users/davidy/fcs