FF NEURAL NETWORK

Rohit Kumar

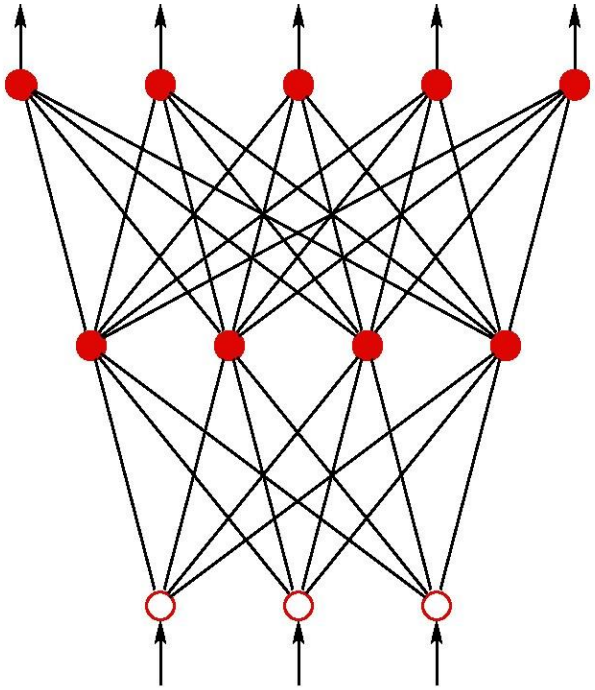
Abstract: *A feedforward neural network is an artificial neural network where connections between the units do not form a directed cycle. This is different from recurrent neural networks. The feedforward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.*

A feedforward neural network is a biologically inspired classification algorithm. It consist of a (possibly large) number of simple neuron-like processing units, organized in layers. Every unit in a layer is connected with all the units in the previous layer. These connections are not all equal, each connection may have a different strength or weight. The weights on these connections encode the knowledge of a network. Often the units in a neural network are also called nodes.

Data enters at the inputs and passes through the network, layer by layer, until it arrives at the outputs. During normal operation, that is when it acts as a classifier, there is no feedback between layers. This is why they are called feedforward neural networks.

In the following figure we see an example of a 2-layered network with, from top to bottom: an output layer with 5 units, a hidden layer with 4 units, respectively. The network has 3 input units.

*Student, CSE, Dronachraya College of Engineering, Gurgaon



The 3 inputs are shown as circles and these do not belong to any layer of the network (although the inputs sometimes are considered as a virtual layer with layer number 0). Any layer that is not an output layer is a hidden layer. This network therefore has 1 hidden layer and 1 output layer. The figure also shows all the connections between the units in different layers. A layer only connects to the previous layer.

1. INTRODUCTION

Feedforward neural networks (FF networks) are the most popular and most widely used models in many practical applications. They are known by many different names, such as "multi-layer perceptrons."

Figure 2.5 illustrates a one-hidden-layer FF network with inputs ,...,and output . Each arrow in the figure symbolizes a parameter in the network. The network is divided into *layers*. The input layer consists of just the inputs to the network. Then follows a *hidden layer,* which consists of any number of *neurons*, or *hidden units* placed in parallel. Each neuron performs a weighted summation of the inputs, which then passes a nonlinear *activation function*, also called the *neuron* function.

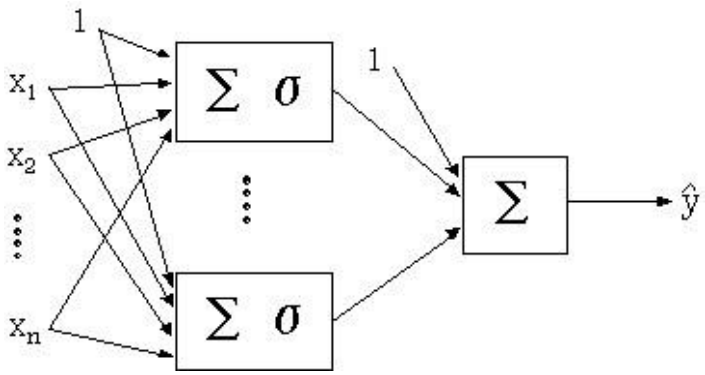


Figure 2.5. A feedforward network with one hidden layer and one output.

Mathematically the functionality of a hidden neuron is described by

$$\sigma\left(\sum_{j=1}^{n} w_j x_j + b_j\right)$$

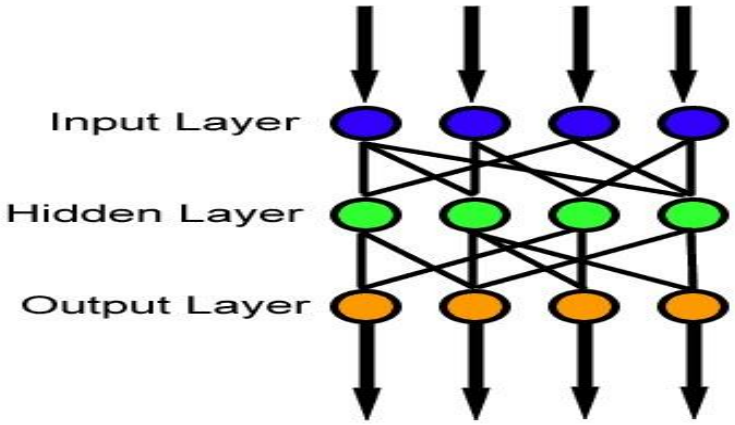
where the weights $\{w_j, b_j\}$ are symbolized with the arrows feeding into the neuron. The network output is formed by another weighted summation of the outputs of the neurons in the hidden layer. This summation on the output is called the *output layer*. In Figure 2.5 there is only one output in the output layer since it is a single-output problem. Generally, the number of output neurons equals the number of outputs of the approximation problem.

The neurons in the hidden layer of the network in Figure 2.5 are similar in structure to those of the perceptron, with the exception that their activation functions can be any differential function. The output of this network is given by

$$\hat{y}(\theta) = g(\theta, x) = \sum_{i=1}^{nh} w_i^2 \sigma\left(\sum_{j=1}^{n} w_{i,j}^1 x_j + b_{j,i}^1\right) + b^2$$

where *n* is the number of inputs and *nh* is the number of neurons in the hidden layer. The variables $\{w_{i,j}^1, b_{j,i}^1, w_i^2, b^2\}$ are the parameters of the network model that are represented collectively by the parameter vector θ. In general, the neural network model will be represented by the compact notation $g(\theta,x)$ whenever the exact structure of the neural network is not necessary in the context of a discussion.

2. BRIEF HISTORY:-



In a feed forward network information always moves one direction; it never goes backwards.

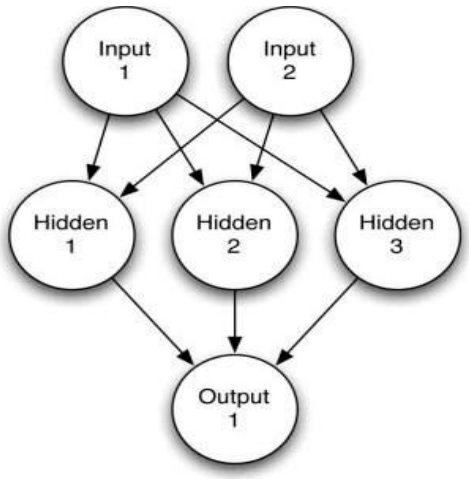
3. FEEDFORWARD NEURAL NETWORK

3.1 Definition

The term, "feed foreword" describes how this neural network processes the pattern and recalls patterns. When using a "feed forward neural network" neurons are only connected foreword. Each layer of the neural network contains connections to the next layer (for example from the input to the hidden layer), but there are no connections back. Feedforward neural network is an interconnection of perceptrons in which data and calculations flow in a single direction, from the input data to the outputs. The number of layers in a neural network is the number of layers of perceptrons.

4. STRUCTURE OF FEEDFORWARD NEURAL NETWORK

In a feedforward neural network, data enters at the inputs and passes through the network, layer by layer, until it arrives at the outputs. During normal operation, that is when it acts as a classifier, there is no feedback between layers. This is why they are called feedforward neural networks. Following figure shows a typical feed forward neural network with a single hidden layer.



4.1. Choosing the Network Structure

There are many ways that feedforward neural networks can be constructed. The user must decide how many neurons will be inside the input and output layers, and also decide how many hidden layers it is going to have, as well as how many neurons will be in each of these hidden layers. There are many techniques for choosing these parameters. There are some of the general "rules of thumb" that can be used to assist making these decisions. In nearly all cases some experimentation will be required to determine the optimal structure for feedforward neural network.

4.2. The Input Layer

The input layer to the neural network is the conduit through which the external environment presents a pattern to the neural network. Once a pattern is presented to the input later of the neural network the output layer will produce another pattern. In essence this is all the neural network does. The input layer should represent the condition for which the neural network is trained for. Every input neuron should represent some independent variable that has an influence over the output of the neural network.

4.3. The Output Layer

The output layer of the neural network is what actually presents a pattern to the external environment. Whatever pattern is presented by the output layer can be directly traced back to the input layer. The number of output neurons should directly related to the type of work that the neural network is to perform. To consider the number of neurons to use in output layer one must consider the intended use of the neural network. If the neural network is to be used to classify items into groups, then it is often preferable to have one output neurons

for each group that the item is to be assigned into. If the neural network is to perform noise reduction on a signal then it is likely that the number of input neurons will match the number of output neurons.

4.4.The Number of Hidden Layers

There are really two decisions that must be made with regards to the hidden layers. The first is how many hidden layers to actually have in the neural network. Secondly, how many neurons will be in each of these layers. Neural networks with two hidden layers can represent functions with any kind of shape. There is currently no theoretical reason to use neural networks with any more than two hidden layers. Further for many practical problems there's no reason to use any more than one hidden layer. Problems that require two hidden layers are rarely encountered. Differences between the numbers of hidden layers are summarized in following table:

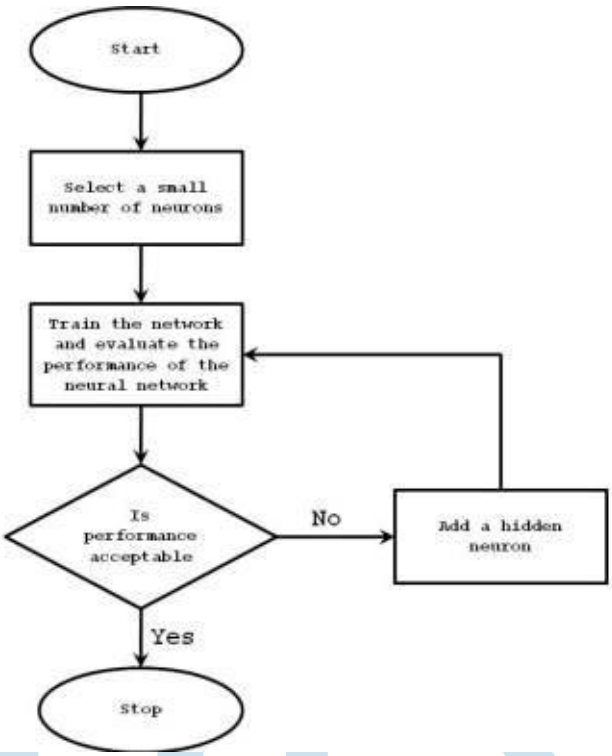
Number of Hidden Layers	**Result**
none	Only capable of representing linear separable functions or decisions.
1	Can approximate arbitrarily while any functions which contains a continuous mapping from one finite space to another.
2	Represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy.

4.5. The Number of Neurons in the Hidden Layers

Deciding the number of hidden neurons in layers is a very important part of deciding the overall neural network architecture. Though these layers do not directly interact with the external environment these layers have a tremendous influence on the final output. Both the number of hidden layers and number of neurons in each of these hidden layers must be considered. Using too few neurons in the hidden layers will result in something called underfitting. Underfitting occurs when there are too few neurons in the hidden layers to adequately detect the signals in a complicated data set. Using too many neurons in the hidden layers can result in several problems. First too many neurons in the hidden layers may result in overfitting. Overfitting occurs when the neural network has so much information processing capacity that the limited amount of information contained in the training set is not enough to train all of the neurons in the hidden layers. A second problem can occur even

when there is sufficient training data. An inordinately large number of neurons in the hidden layers can increase the time it takes to train the network. The amount of training time can increase enough so that it is impossible to adequately train the neural network. Obviously some compromise must be reached between too many and too few look neurons in the hidden layers. There are many rule-of-thumb methods for determining the correct number of neurons to use in the hidden layers. Some of them are summarized as follows.

- The number of hidden neurons should be in the range between the size of the input layer and the size of the output layer.
- The number of hidden neurons should be 2/3 of the input layer size, plus the size of the output layer.
- The number of hidden neurons should be less than twice the input layer size. These three rules are only starting points to consider. Ultimately the selection of the architecture of the neural network will come down to trial and error. But what exactly is meant by trial and error? Nobody wants to start throwing random layers and numbers of neurons at the network. To do so would be very time-consuming. There are two methods that can be used to organize the trial and error search for the optimum network architecture. There are two trial and error approaches that one may use in determining the number of hidden neurons are the "forward" and "backward" selection methods. The first method, the "forward selection method", begins by selecting a small number of hidden neurons. This method usually begins with only two hidden neurons. Then the neural network is trained and tested. The number of hidden neurons is then increased and the process is repeated so long as the overall results of the training and testing improved.The "forward selection method" is summarized in following figure.



The second method, the "backward selection method", begins by using a large number of hidden neurons. Then the neural network is trained and tested. This process continues until about the performance improvement of the neural network is no longer significant. One additional method that can be used to reduce the number of hidden neurons is called pruning. In the simplest sense pruning involves evaluating the weighted connections between the layers. If the network contains any hidden neurons which contains only zero weighted connections, they can be removed. Pruning is a very important concept for neural networks.

5. OPERATION

The operation of this network can be divided into two phases:

5.1. The Learning Phase

The feedforward network uses a supervised learning algorithm: besides the input pattern, the neural net also needs to know to what category the pattern belongs. Learning proceeds as follows: a pattern is presented at the inputs. The pattern will be transformed in its passage through the layers of the network until it reaches the output layer. The units in the output layer all belong to a different category. The outputs of the network as they are now are compared with the outputs as they ideally would have been if this pattern were correctly classified: in the latter case the unit with the correct category would have had the largest output value and the output values of the other output units would have been very small. On

the basis of this comparison all the connection weights are modified a little bit to guarantee that, the next time this same pattern is presented at the inputs, the value of the output unit that corresponds with the correct category is a little bit higher than it is now and that, at the same time, the output values of all the other incorrect outputs are a little bit lower than they are now. (The differences between the actual outputs and the idealized outputs are propagated back from the top layer to lower layers to be used at these layers to modify connection weights. This is why the term backpropagation network is also often used to describe this type of neural network. The time for learning phase depends on the size of the neural network, the number of patterns to be learned, the number of epochs, the tolerance of the minimizer and the speed of your computer, how much computing time the learning phase may take.

5.2. Backpropagation

Backpropagation is the most commonly implemented training procedure for feedforward neural networks. Its primary objective is to provide a mechanism for updating connected neurons based upon minimization of error. To accomplish this, gradient descent is generally used to determine the steepest path toward the minimum of

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

.

where *d* is a training instance in *D*, t_d is the target value, o_d is the output value, and $\vec{w}$ is the weight vector.

Backpropagation requires determining an error by first feedfowarding inputs into the network and subtracting the result from some target output. This difference is then multiplied by the derivative of the neuron's activation function -- in the case of the sigmoid, this is $f'(net) = o(1 - o)$ -- and stored for reference by the update at the preceding layer. We can now proceed to make error calculations layer-by-layer by traversing backward through the network and performing the neuron's error computation, which is the derivative of the neuron activation function multiplied by the sum of each output weight's multiplication with the forward neuron's error term. After each error term is calculated, we update the weights by the multiplication of each branch's output with the forward node's error and the learning rate. [Charles Nugent]

5.3. The Classification Phase

In the classification phase, the weights of the network are fixed. A pattern, presented at the inputs, will be transformed from layer to layer until it reaches the output layer. The classification can occur by selecting the category associated with the output unit that has the largest output value.In contrast to the learning phase classification is very fast.

6. APPLICATIONS

Feedforward neural network, part of artificial neural network, has broad applicability to real world business problems. In fact, they have already been successfully applied in many industries. Since the network is best at identifying patterns or trends in data, it is well suited for prediction or forecasting needs including: sales forecasting, industrial process control, customer research, data validation, risk management, target marketing.

It is also used following specific paradigms: recognition of speakers in communications; diagnosis of hepatitis; recovery of telecommunications from faulty software; interpretation of multi meaning Chinese words; undersea mine detection; texture analysis; threedimensional object recognition; hand-written word recognition; and facial recognition.

It is a 'hot' research area in medicine and it is believed that it will receive extensive application to biomedical systems in the next few years. It is ideal in recognising diseases using scans since there is no need to provide a specific algorithm on how to identify the disease. It learns by example so the details of how to recognize the disease are not needed. What is needed is a set of examples that are representative of all the variations of the disease. The quantity of examples is not as important as the 'quantity'. The examples need to be selected very carefully if the system is to perform reliably and efficiently.

REFERENCES

[1]. http://www.wikipedia.org/neuralnetwork

[2]. http://www.studymode.com/neuralnetwork

[3]. http://www.fon.hum.uva.nl/praat/manual/Feedforward_neural_networks.html.Retrieved

[4]. Jeff Heaton,2008. Introduction to Neural Networks for Java, 2nd Edition,Heaton Research, Inc. ISBN:1604390085 9781604390087