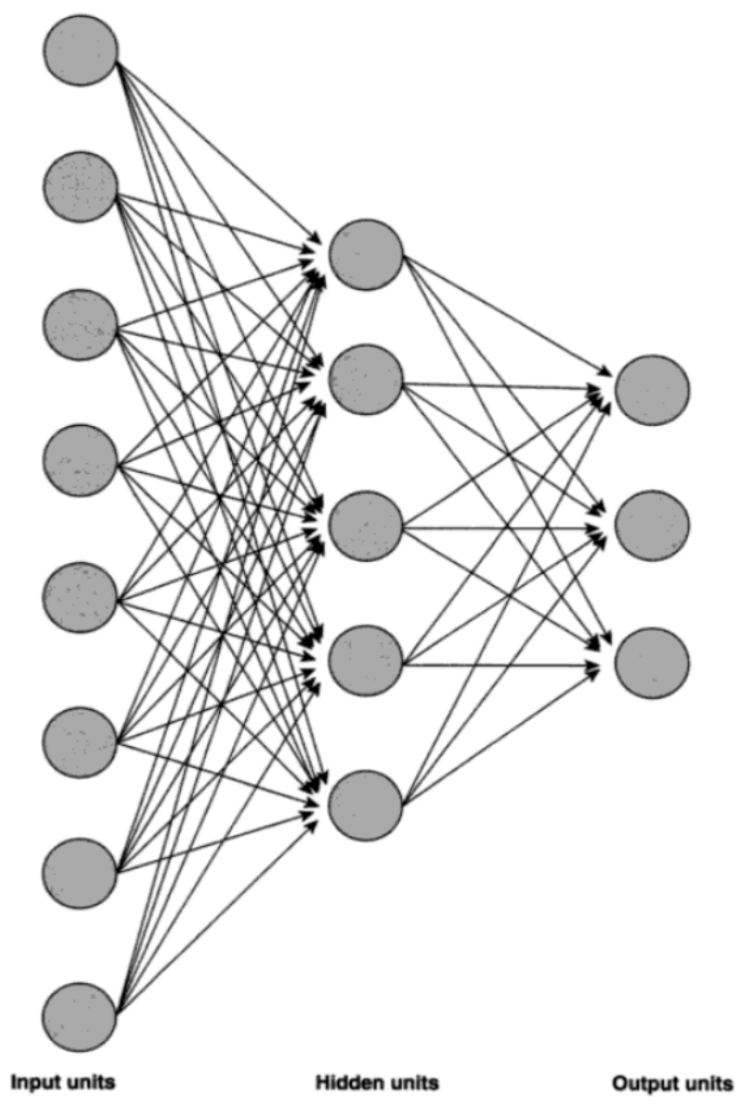
Farrukh Jabeen
Due Date: November 2, 2009.
Neural Networks: Backpropation

Assignment # 5

The "Backpropagation" method is one of the most popular methods of "learning" by a neural network. Read the class notes and then answer the following questions:

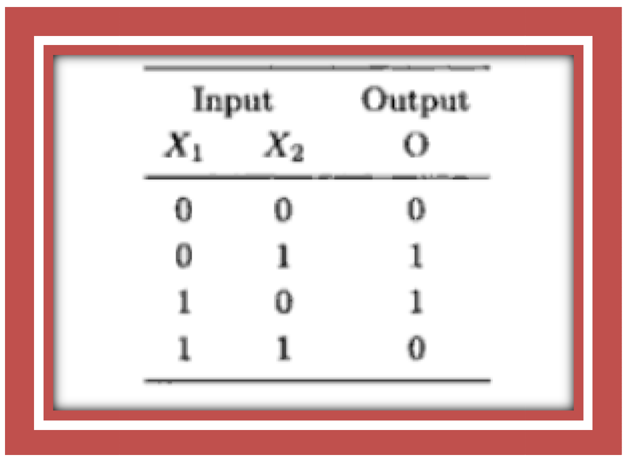
1. Define the terms: input layer, hidden layer, and output layer.

In a three layer network , the computing units or nodes are organized into three distinct groups. A representation of the stimulus is presented to the input layer. These units send signals to a hidden layer , which in turns feeds activation energy to an output layer. The output layer generates a representation of the response. The following figure shows three layer network.



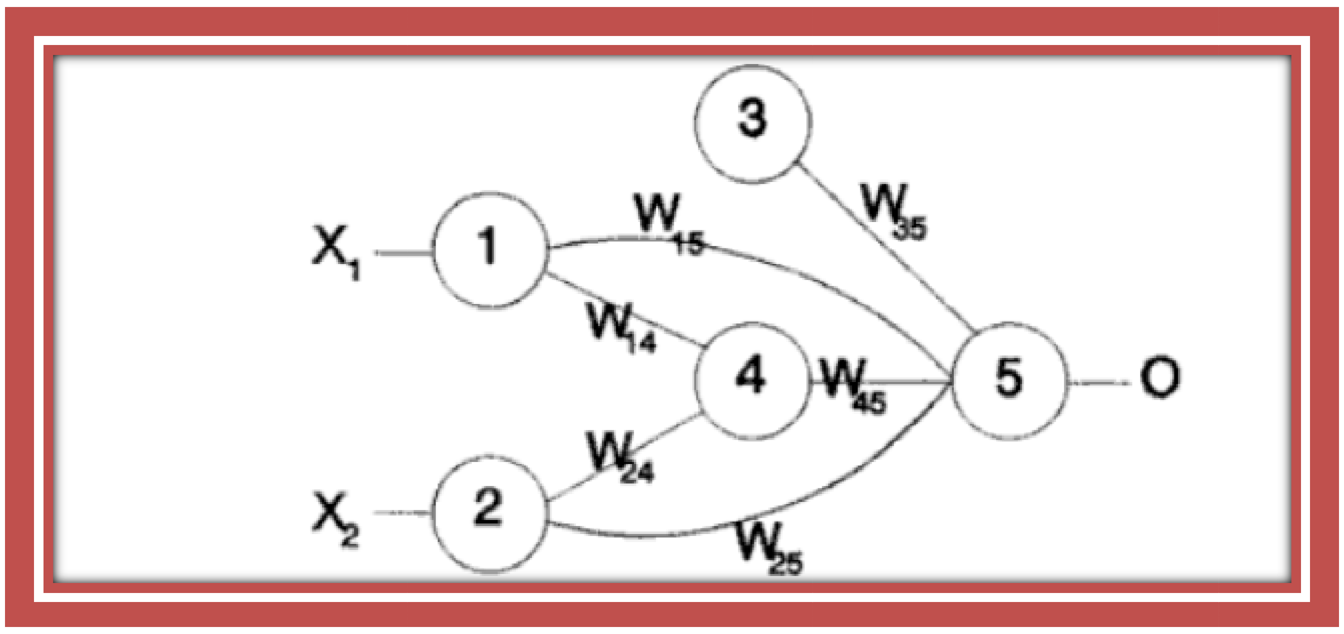
2. What is the XOR problem in this context?

The first useful artificial neural network was the perceptron. This simple two layered network was used to recognize characters of English alphabet, even those with imperfections. It was later proved that the perceptron was limited to making associations between similar pattern only. This means that if the outputs were not linearly separable , the problem could not be solved by perceptron. The classic example was eXclusive-OR(XOR) problem as shown in the following table



Input		Output
X_1	X_2	O
0	0	0
0	1	1
1	0	1
1	1	0

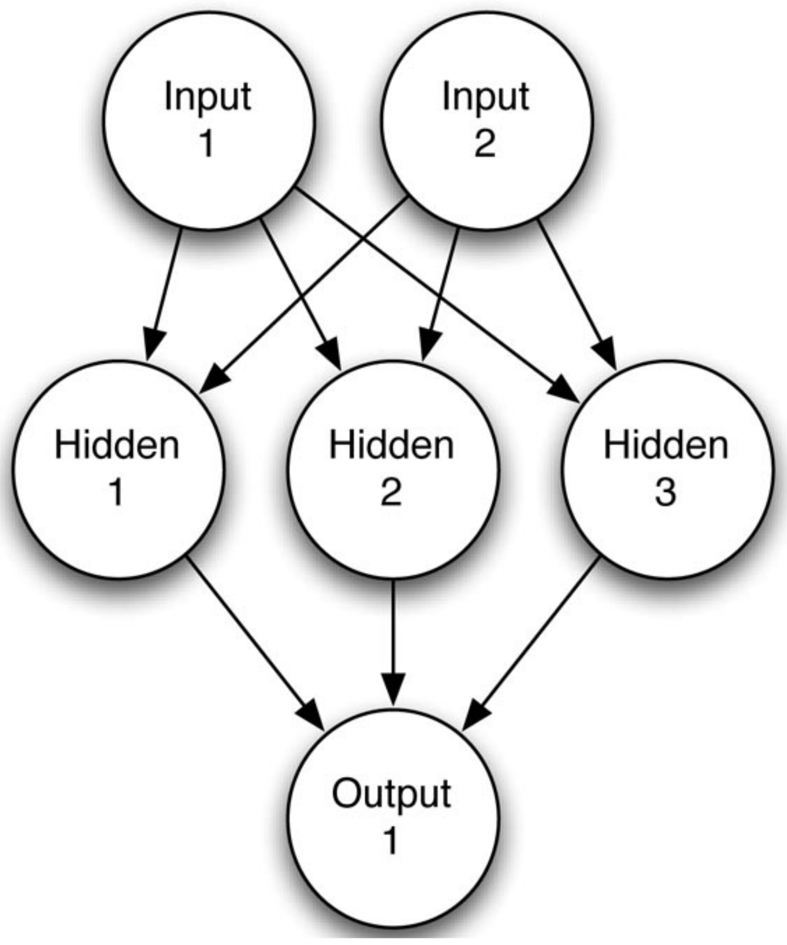
The back-propagation model has been used to solve a great variety of problems. The back-propagation neural network uses a two cycle procedure during the learning phase. The first cycle involves transforming the input pattern into an output pattern. The second involves calculating the error and propagating an error signal back throughout the network. The magnitude of an error signal at a particular node is proportional to how much that node influences the output pattern. The connection weights are then modified a small amount in a way that decreases the total error.



A back-propagating network for the XOR problem

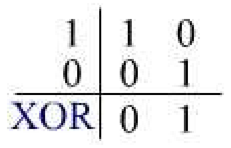
3. Explain why there is no way to solve the XOR problem with a feedforward backpropagation network with a single hidden layer.

The feedforward neural network begins with an input layer. The input layer may be connected to a hidden layer or directly to the output layer. If it is connected to a hidden layer, the hidden layer can then be connected to another hidden layer or directly to the output layer. There can be any number of hidden layers, as long as there is at least one hidden layer or output layer provided. In common use, most neural networks will have one hidden layer, and it is very rare for a neural network to have more than two hidden layers.



The earliest kind of neural network is a single-layer perceptron network, which consists of a single layer of output nodes; the inputs are fed directly to the outputs via a series of weights. In this way it can be considered the simplest kind of feed-forward network. The sum of the products of the weights and the inputs is calculated in each node, and if the value is above some threshold (typically 0) the neuron fires and takes the activated value (typically 1); otherwise it takes the deactivated value (typically -1). Neurons with this kind of activation function are also called McCulloch-Pitts neurons or threshold neurons. The term perceptron often refers to networks consisting of just one of these units.
Perceptrons can be trained by a simple learning algorithm that is usually called the delta rule. It calculates the errors between calculated output and sample output data, and uses this to create an adjustment to the weights, thus implementing a form of gradient descent. Single-unit perceptrons are only capable of learning linearly separable patterns; Hence for such a network it is impossible to learn an XOR function.
A single-layer neural network can compute a continuous output instead of a step function. A common choice is the so-called logistic function. With this choice, the single-layer network is identical to the logistic regression model, widely used in statistical modelling. The logistic function is also known as the sigmoid function. It has a continuous derivative, which allows it to be used in backpropagation which however is more common to be used to train multi-layered artificial networks (MLPs).

XOR operator is not linearly separable and cannot be achieved by a single perceptron. Yet this problem could be overcome by using more than one perceptron arranged in feed-forward networks.

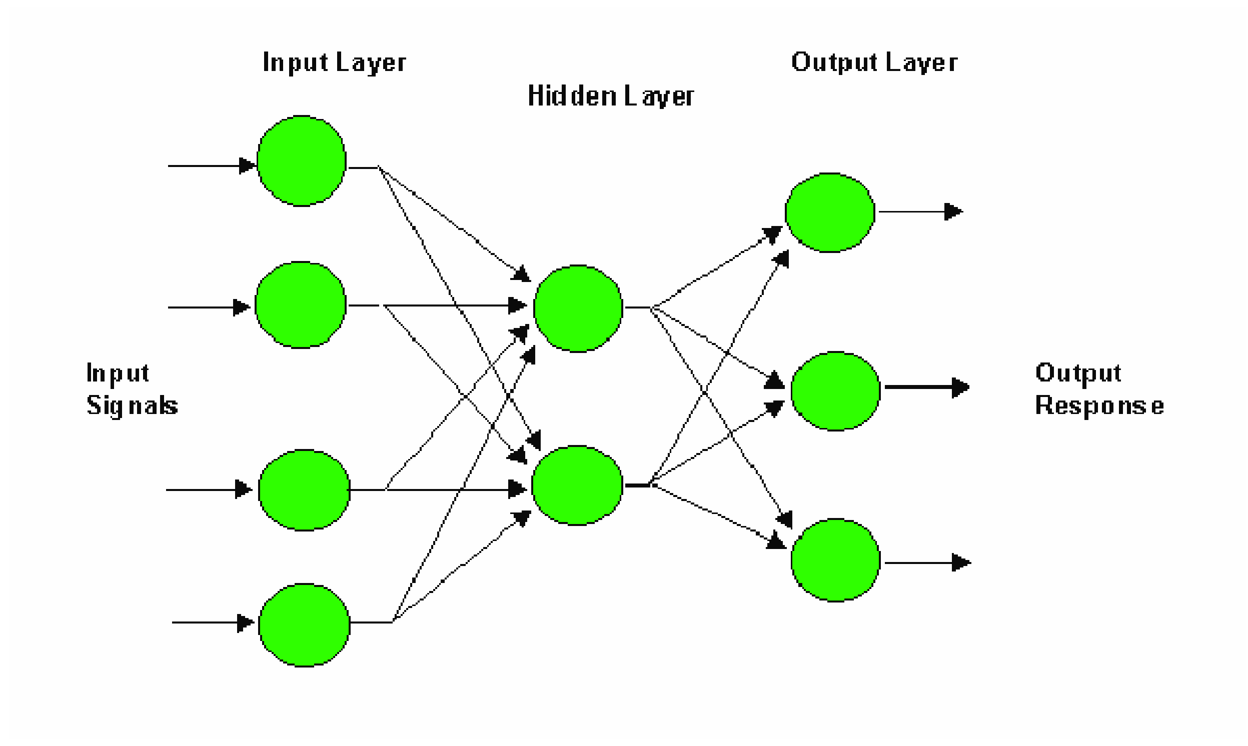


1	1	0
0	0	1
XOR	0	1

Since it is impossible to draw a line to divide the regions containing either 1 or 0, the XOR function is not linearly separable.

4. Explain the role of the external input. That is, what geometric significance does it have?

Back propagation involves two phases: a feed forward phase in which the **external input** information at the input nodes is propagated forward to compute the output information signal at the output unit, and a backward phase in which modifications to the connection strengths are made based on the differences between the computed and observed information signals at the output units (Eberhart and Dobbins, 1990). The neural network structure in this study possessed a three-layer learning network consisting of an input layer, a hidden layer and an output layer. The architecture of a typical layered forward neural network is shown in the following Figure.



5. Explain why a network with a single hidden layer cannot separate non-convex regions.

Single-layer and multi-layer neural networks are organized in layers of neurons. The most simple layered neural network is constituted by the input nodes and a layer of output neurons. There are only connections between the inputs and the output layer (not the opposite), hence information is processed from the input to the output, reason for calling them feedforward neural networks.
One can have more complex layered neural networks by adding more layers of neurons building a multi-layer feedforward neural network also known as multilayer perceptron (MLP).
The output signals from the first layer are the inputs for the output layer. The layers between the input and output layers are known as hidden layers (neurons in these layers are called hidden neurons).
Multi-layer perceptrons are capable of more complex mappings than single layer perceptrons. A single-layer perceptron can only implement a linear discriminant producing an hyperplane as decision boundary. A multi-layer perceptron with one hidden layer is able to generate an open or closed convex region in the input space, whose boundary are segments of hyperplanes. This convex region is obtained with the ability of the hidden layer to perform an AND operation. To get non-convex and/or disjoint regions we must add a layer to perform the OR operation. Multi-layer perceptron with two hidden layers and threshold activation functions are capable of defining arbitrary regions.

6. Explain the process of *training* a typical feedforward backpropagation network.

We will consider the feedforword backpropagation neural network. This neural network architecture is very popular, because it can be applied to many different tasks. To understand this neural network architecture, we must examine how it is trained and how it processes a pattern.

The first term, “feedforward” describes how this neural network processes and recalls patterns. In a feedforward 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.

The term “backpropagation” describes how this type of neural network is trained. Backpropagation is a form of supervised training. When using a supervised training method, the network must be provided with both sample inputs and anticipated outputs. The anticipated outputs are compared against the actual outputs for given input. Using the anticipated outputs, the backpropagation training algorithm then takes a calculated error and adjusts the weights of the various layers backwards from the output layer to the input layer. Now we discuss the exact process by which backpropagation occurs.

The Backpropagation Algorithm

Multilayer feedforward networks normally consist of three or four layers, there is always one input layer and one output layer and usually one hidden layer, although in some classification problems two hidden layers may be necessary, this case is rare however. The term input layer neurons are a misnomer, no sigmoid unit is applied to the value of each of these neurons. Their raw values are fed into the layer downstream the input layer (the hidden layer). Once the neurons for the hidden layer are computed, their activations are then fed downstream to the next layer, until all the activations eventually reach the output layer, in which each output layer neuron is associated with a specific classification category. In a fully connected multilayer feedforward network, each neuron in one layer is connected by a weight to every neuron in the layer downstream it. A bias is also associated with each of these weighted sums. Thus in computing the value of each neuron in the hidden and output layers one must first take the sum of the weighted sums and the bias and then apply f(sum) (the sigmoid function) to calculate the neuron's activation.

How then does the network learn the problem at hand? By modifying the all the weights of course. According to Calculus by taking the partial derivative of the error of the network with respect to each weight we will learn a little about the direction the error of the network is moving. In fact, if we take negative this derivative (i.e. the rate change of the error as the value of the weight increases) and then proceed to add it to the weight, the error will decrease until it reaches a local minima. This makes sense because if the derivative is positive, this tells us that the error is increasing when the weight is increasing, the obvious thing to do then is to add a negative value to the weight and vice versa if the derivative is negative.
Because the taking of these partial derivatives and then applying them to each of the weights takes place starting from the output layer to hidden layer weights, then the hidden layer to input layer weights, (as it turns out this is necessary since changing these set of weights requires that we know the partial derivatives calculated in the layer downstream) this algorithm has been called the "**back propagation algorithm".**

Supervised Training

The Backpropagation works in two modes, a supervised training mode and a production mode.
The training can be summarized as follows:
Start by initializing the input weights for all neurons to some random numbers between 0 and 1, then:

1. Apply input to the network.
2. Calculate the output.
3. Compare the resulting output with the desired output for the given input. This is called the error.
4. Modify the weights and threshold θ for all neurons using the error.
5. Repeat the process until error reaches an acceptable value.
6. The challenge is to find a good algorithm for updating the weights and thresholds in each iteration (step 4) to minimize the error.

Changing weights and threshold for neurons in the output layer is different from hidden layers.

Before we explain the training, let's define the following:

1. λ (Lambda) the Learning Rate: a real number constant, usually 0.2 for output layer neurons and 0.15 for hidden layer neurons.

2. Δ (Delta) the change: For example Δx is the change in x. Note that Δx is a single value and not Δ multiplied by x.

Output Layer Training

Let z be the output of an output layer neuron .
Let y be the desired output for the same neuron, it should be scaled to a value between 0 and 1. This is the ideal output which we like to get when applying a given set of input.
Then e (the error) will be:

$e = z * (1 – z) * (y – z)$

$\Delta\theta = \lambda * e$ ... *The change in* θ

$\Delta wi = \Delta\theta * xi$ ... *The change in weight at input i of the neuron*

In other words, for each output neuron, calculate its error e, and then modify its threshold and weights using the formulas above.

Hidden Layer Training

Let z be the output of the hidden layer neuron
Let mi be the weight at neuron Ni in the layer following the current layer. This is the weight for the input coming from the current hidden layer neuron.
Let ei be the error (e) at neuron Ni.
$g = \Sigma\, mi * ei$
$e = z * (1 - z) * g$ ... *Error at the hidden layer neuron*
$\Delta\theta = \lambda * e$ ... *The change in* θ
$\Delta wi = \Delta\theta * xi$ ... *The change in weight i*

Notice that in calculating g, we used the weight mi and error ei from the following layer, which means that the error and weights in this following layer should have already been calculated. This implies that during a training iteration of a Backpropagation, we start modifying the weights at the output layer, and then we proceed backwards on the hidden layers one by one until we reach the input layer. It is this method of proceeding backwards which gives this network its name *Backward Propagation*.

7. Explain how the training process induces a *gradient descent* on the error function.

Learning in a backpropagation network is in two steps. First each pattern $\mathbf{I_p}$ is presented to the network and propagated forward to the output. Second, a method called gradient descent is used to minimize the total error on the patterns in the training set. In gradient descent, weights are changed in proportion to the negative of an error derivative with respect to each weight:

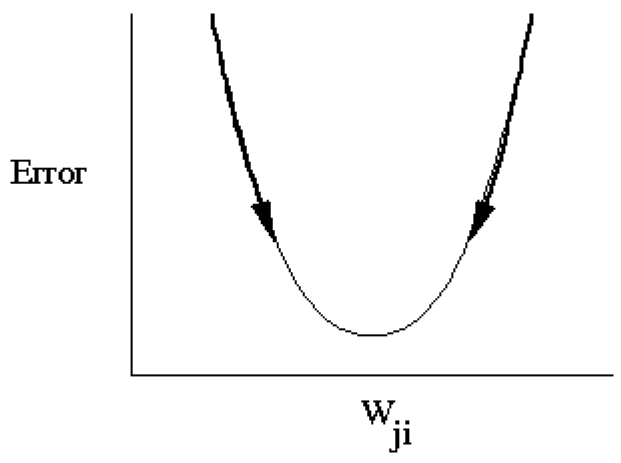
$$\Delta w_{ji} = -\varepsilon\, [\partial E/\partial w_{ji}]$$

Weights move in the direction of steepest descent on the error surface defined by the total error (summed across patterns):

$$E = 1/2\ \Sigma_p\ \Sigma_j(t_{pj}-o_{pj})^{2}$$

where $\mathbf{o_{pj}}$ be the activation of output unit $\mathbf{u_j}$ in response to pattern **p** and $\mathbf{t_{pj}}$ is the target output value for unit $\mathbf{u_j}$

following figure illustrates the concept of gradient descent using a single weight. After the error on each pattern is computed, each weight is adjusted in proportion to the calculated error gradient backpropagated from the outputs to the inputs. The changes in the weights reduce the overall error in the network.



8. Why is it important to have a representative training set? For example, suppose a neural network were trained on economic or stock market data from the 1990's and then expected to make predictions in today's market. Why might there be a problem?

Prediction of stock market returns is an important issue in finance. Nowadays artificial neural networks (ANNs) have been popularly applied to finance problems such as stock exchange index prediction, bankruptcy prediction and corporate bond classification.

The simplified ANN (supervised) training and prediction process can be illustrated by the following steps.

Step One:

Collect the training set, which includes the input data for the ANN to "see" and the known target data for ANN to learn to output. For stock price predictions, for example, the training set and target data would naturally be historical stock prices. A vector of 100 consecutive historical stock prices, for instance, can constitute training data and with the 101st stock price as a target datum.

Step Two:

Feed the input data to ANN; compare ANN output with the known target, and adjust ANN's internal parameters (weights and biases) so that ANN output and the known target are close to one another--more precisely, so that a certain error function is determined and further minimized.

Step Three:

Feed ANN some future input data (not seen by ANN); if ANN is well trained and if the input data are predictable, then ANN will give accurate predictions.

For example, suppose a neural network were trained on economic or stock market data from the 1990's and then expected to make predictions in today's market. Why might there be a problem?

A system designed to analyze IBM stock returns was trained on 1000 days of data and tested on 500 days of data. A similar system designed to predict Tokyo stocks learned daily data over 33 months. Some systems were even trained on data spanning over 50 years. It is desirable to have a lot of data available, as some patterns may not be detectable in small data sets. However, it is often difficult to obtain a lot of data with complete and correct values. As well, training on large volumes of historical data is computationally and time intensive and may result in the network learning undesirable information in the data set. For example, stock market data is time-dependent. Sufficient data should be presented so that the neural network can capture most of the trends, but very old data may lead the network to learn patterns or factors that are no longer important or valuable.

References

http://books.google.com/books?id=wGti6_4Qn_QC&pg=PA215&lpg=PA215&dq=input+layer+backpropagation&source=bl&ots=wqBvxc2R7s&sig=I1Q1RZeqyhbMcMyrh3jSul3ITqg&hl=en&ei=Auy0Str_Do-KsgP37MjRDA&sa=X&oi=book_result&ct=result&resnum=5#v=onepage&q=&f=false
http://books.google.com/books?id=pfEATHvL-uYC&pg=PA294&lpg=PA294&dq=XOR++problem++and+back+propagation&source=bl&ots=hcIjzRaphs&sig=GzMui2NxvO9WgFy9Wwf_4PuUQUM&hl=en&ei=UVO4SranFIrKsAOg_70j&sa=X&oi=book_result&ct=result&resnum=4#v=onepage&q=XOR%20%20problem%20%20and%20back%20propagation&f=false
http://www.heatonresearch.com/node/704
http://www.nullpointer.ch/machine-learning/supervised-learning/MLP/
http://cse.stanford.edu/class/sophomore-college/projects-00/neural-networks/Neuron/index.html
http://balwois.com/balwois/administration/full_paper/ffp-471.pdf
http://www.itee.uq.edu.au/~cogs2010/cmc/chapters/BackProp/index2.html
http://www.freewebs.com/farrukhjabeen/-%20Assignment4Backpropagation/Microsoft%20Word%20-%20BackpropReport.pdf
http://www.hicbusiness.org/biz2003proceedings/Birgul%20Egeli.pdf
Using Neural Networks to Forecast Stock Market Prices
Ramon Lawrence
Department of Computer Science
University of Manitoba
umlawren@cs.umanitoba.ca
December 12, 1997
http://gnomo.fe.up.pt/~nnig/papers/Thesis_Santos_Jorge_2007.pdf
http://www.patentstorm.us/patents/6735580/description.html