Neural Networks and Back Propagation Algorithm

Mirza Cilimkovic *

Institute of Technology Blanchardstown

Blanchardstown Road North

Dublin 15

Ireland

mirzac@gmail.com

Abstract

Neural Networks (NN) are important data mining tool used for classification and clustering. It is an attempt to build machine that will mimic brain activities and be able to learn. NN usually learns by examples. If NN is supplied with enough examples, it should be able to perform classification and even discover new trends or patterns in data. Basic NN is composed of three layers, input, output and hidden layer. Each layer can have number of nodes and nodes from input layer are connected to the nodes from hidden layer. Nodes from hidden layer are connected to the nodes from output layer. Those connections represent weights between nodes.

This paper describes one of most popular NN algorithms, Back Propagation (BP) Algorithm. The aim is to show the logic behind this algorithm. Idea behind BP algorithm is quite simple, output of NN is evaluated against desired output. If results are not satisfactory, connection (weights) between layers are modified and process is repeated again and again until error is small enough. Simple BP example is demonstrated in this paper with NN architecture also covered. New implementation of BP algorithm are emerging and there are few parameters that could be changed to improve performance of BP.

Keywords: Neural Networks, Artificial Neural Networks, Back Propagation algorithm

*Student Number B00000820.

1 Introduction

Classification is grouping of the objects or things that are similar. Examples of classifications are found everywhere, supermarkets put similar things together, there are shelf for meat, diary products, cleaning products. Classifications makes life easier, like in supermarket example, if things were put on shelf in random order, it would make shopping unpleasant and lengthy experience. Most financial institutions use classification for their credit ratings, medicine uses it extensively for diagnosing diseases.

There are many classification techniques used in data mining with NN being one of them. NN model is explained in this paper but distinction has to be made between NN in humans and animals and NN that are being used in industries. Former NN are often called Artificial NN and they will be discussed in paper but for simplicity they will be referred to as NN. There are two types of NN based on learning technique, they can be supervised where output values are known beforehand (back propagation algorithm) and unsupervised where output values are not known (clustering).

NN architecture, number of nodes to choose, how to set the weights between the nodes, training the network and evaluating the results are covered. Activation function gets mentioned together with learning rate, momentum and pruning. Back propagation algorithm, probably the most popular NN algorithm is demonstrated.

2 Neural Networks

'Neural networks have seen an explosion of interest over the last few years and are being successfully applied across an extraordinary range of problem domains, in areas as diverse as finance, medicine, engineering, geology and physics.' Statsoft.com [2010]

It all started way back in 1943 when McCullock and Pitts proved that neuron can have two states and that those states could be dependent on some threshold value. They presented first artificial neuron model according to Rojas [2005]. Many new and more sophisticated models have been presented since. McCullock and Pitts discovery opened door for intelligent machines. According to the Muller et al. [1995] there are two main reason for NN investigation, first is to try to get an understanding on how human brain function and second is desire to build machines that are capable for solving complex problems that sequentially operating computers were unable to solve.

If problem structure is well analyzed, traditional computers could still outperform NN but in cases where problem has not been analyzed in details, NN could be used to learn from large set of examples. NN can

handle errors better than traditional computers programs (imagine scenario where one faulty statement in program could halt everything while NN can handle errors in much better manner).

2.1 Neuron

Vaga [1994] Human brain has over 100 billion interconnected neurons. Most sophisticated application have only tiny fraction of that. It can only be imagined how powerful NN with this number of interconnected neurons would be. Neurons use this interconnected network to pass information's with each other using electric and chemical signals. Although it may seem that neurons are fully connected, two neurons actually do not touch each other. They are separated by tiny gap call Synapse. Each neuron process information and then it can "connect" to as many as 50 000 other neurons to exchange information. If connection between two neuron is strong enough (will be explained later) information will be passed from one neuron to another. On their own, each neuron is not very bright but put 100 billion of them together and let them talk to each other, then this system becomes very powerful. A typical neuron would have 4 components seen on Figure 1: Dendrites, Soma, Axon and Synapse. Dendrites gather inputs from other neurons and when a certain

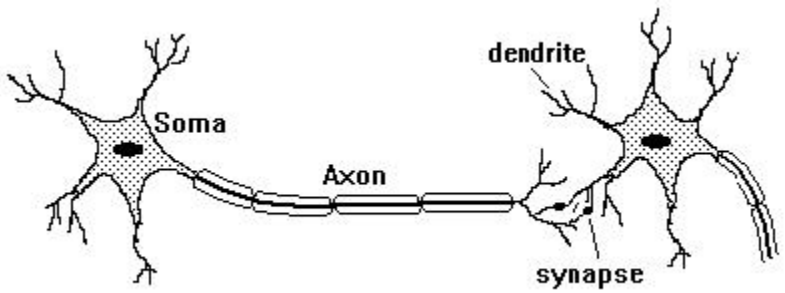


Figure 1: ComputingAssociate.com [2010]

threshold is reached they generate a non-linear response (signal to other neurons via the Axon).

2.2 Architecture

Figure 2 represent architecture of an simple NN. It is made up from an input, output and one or more hidden layers. Each node from input layer is connected to a node from hidden layer and every node from hidden layer is connected to a node in output layer. There is usually some weight associated with every connection. Input layer represents an the raw information that is fed into the network. This part of network is never changing its values. Every single input to the network is duplicated and send down to the nodes in hidden layer. Hidden Layer accepts data from the input layer. It uses input values and modifies them using some

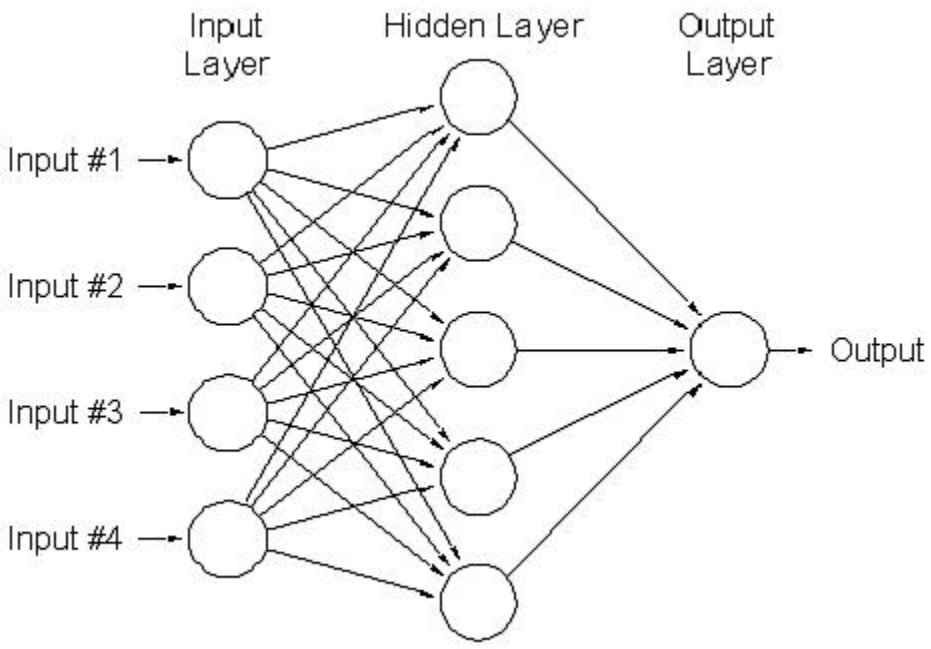


Figure 2: Simple Neural Network

weight value, this new value is than send to the output layer but it will also be modified by some weight from connection between hidden and output layer. Output layer process information received from the hidden layer and produces an output. This output is than processed by activation function.

2.3 Number of Nodes and Layers

Choosing number of nodes for each layer will depend on problem NN is trying to solve, types of data network is dealing with, quality of data and some other parameters. Number of input and output nodes depends on training set in hand. Larose [2005] argued that choosing number of nodes in hidden layer could be challenging task. If there are too many nodes in hidden layer, number of possible computations that algorithm has to deal with increases. Picking just few nodes in hidden layer can prevent the algorithm of its learning ability. Right balance needs to be picked. It is very important to monitor the progress of NN during its training, if results are not improving, some modification to the model might be needed.

2.4 Setting Weights

The way to control NN is by setting and adjusting weights between nodes. Initial weights are usually set at some random numbers and than they are adjusted during NN training.

According to Fogel [2002] focus should not be at changing one weight at time, changing all the weights should be attempted simultaneously. Some NN are dealing with thousands, even millions of nodes so changing one or two at time would not help in adjusting NN to get desired results in timely manner.
Logic behind weight updates is quite simple. During the NN training weights are updated after iterations. If results of NN after weights updates are better than previous set of weights, the new values of weights are kept and iteration goes on. Finding combination of weights that will help us minimize error should be main aim when setting weights. This will become bit more clear once the learning rate; momentum and training set are explained.

2.5 Running and training NN

Running the network consist of a forward pass and a backward pass. In the forward pass outputs are calculated and compared with desired outputs. Error from desired and actual output are calculated. In the backward pass this error is used to alter the weights in the network in order to reduce the size of the error. Forward and backward pass are repeated until the error is low enough (users usually set the value of accepted error).
Training NN could be separate topic but for the purpose of this paper, training will be explained briefly. When training NN, we are feeding network with set of examples that have inputs and desired outputs.
If we have some set of 1000 samples, we could use 100 of them to train the network and 900 to test our model. Choosing the learning rate and momentum will help with weight adjustment.

- Setting right learning rate could be difficult task, if learning rate is too small, algorithm might take long time to converges. On the other hand, choosing large learning rate could have opposite effect, algorithm could diverge. Sometimes in NN every weight has it's own learning rate. Learning rate of 0.35 proved to be popular choice when training NN. This paper will use rate of 0.45 but this value is used because of simple architecture of NN used in example.

- Larose [2005] claimed that momentum term represents inertia. Large values of momentum term will influence the adjustment in the current weight to move in same direction as previous adjustment.

2.6 Activation Function

According to Faqs.org [2010] activations function are needed for hidden layer of the NN to introduce nonlinearity. Without them NN would be same as plain perceptions. If linear function were used, NN would not

be as powerful as they are.

Activation function can be linear, threshold or sigmoid function. Sigmoid activation function is usually used for hidden layer because it combines nearly linear behavior, curvilinear behavior and nearly constant behavior depending on the input value Larose [2005]. To explain activation function figure 3 will be used.

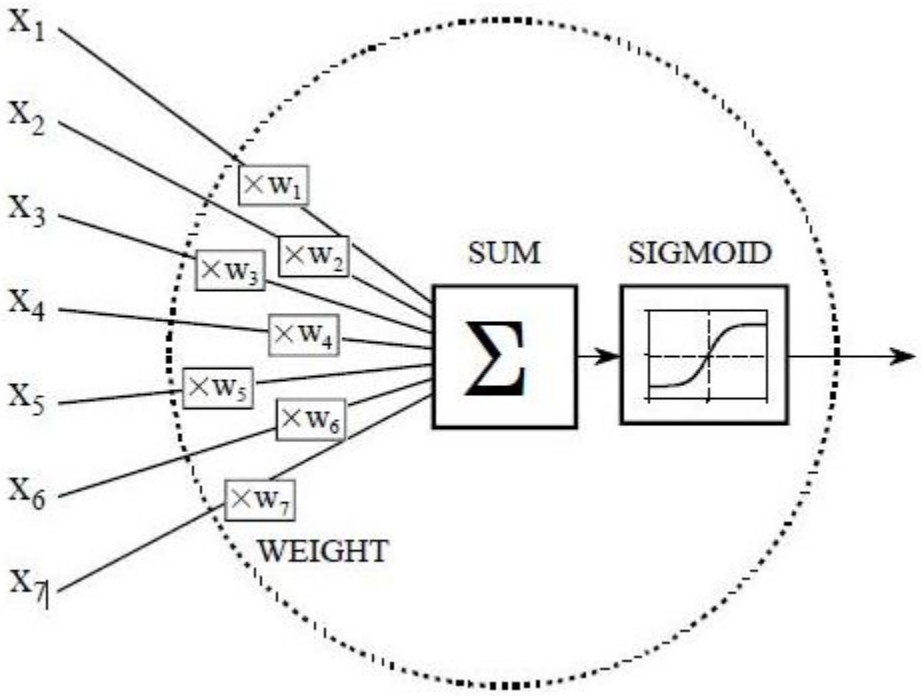


Figure 3: Dspguide.com [2010]

SUM is collection of the output nodes from hidden layer that have been multiplied by connection weights, added to get single number and put through sigmoid function (activation function). Input to sigmoid is any value between negative infinity and positive infinity number while the output can only be a number between 0 and 1. Dspguide.com [2010].

2.7 Pruning

Pruning could be useful technique when dealing with large NN. According to Michie et al. [1994] pruning useless nodes or weights have numerous advantages on NN. Smaller networks and faster training times on serial computers have some advantages. Smaller networks are easier to interpret but pruning should be considered carefully as it can impact network performance.

3 Back Propagation (BP) Algorithm

One of the most popular NN algorithms is back propagation algorithm. Rojas [2005] claimed that BP algorithm could be broken down to four main steps. After choosing the weights of the network randomly, the back propagation algorithm is used to compute the necessary corrections. The algorithm can be decomposed in the following four steps:

i) Feed-forward computation

ii) Back propagation to the output layer

iii) Back propagation to the hidden layer

iv) Weight updates

The algorithm is stopped when the value of the error function has become sufficiently small.

This is very rough and basic formula for BP algorithm. There are some variation proposed by other scientist but Rojas definition seem to be quite accurate and easy to follow. The last step, weight updates is happening through out the algorithm.

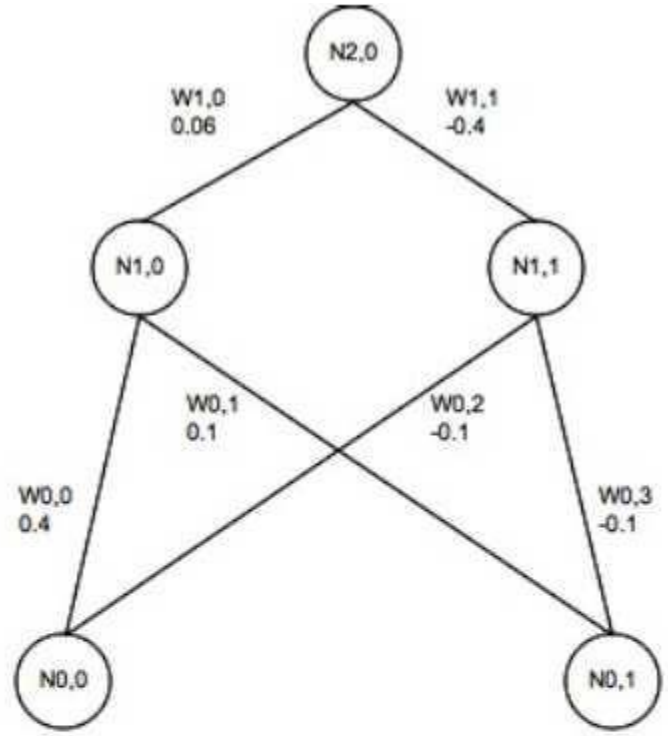
BP algorithm will be explained using exercise example from figure 4.

3.1 Worked example

NN on figure 4 has two nodes (N0,0 and N0,1) in input layer, two nodes in hidden layer (N1,0 and N1,1) and one node in output layer (N2,0). Input layer nodes are connected to hidden layer nodes with weights (W0,1-W0,4). Hidden layer nodes are connected with output layer node with weights (W1,0 and W1,1). The values that were given to weights are taken randomly and will be changed during BP iterations. Table with input node values and desired output with learning rate and momentum are also given in figure 5. There is also sigmoid function formula $f(x) = 1.0/(1.0 + exp(-x))$. Shown are calculations for this simple network (only calculation for example set 1 is going to be shown (input values of 1 and 1 with output value 1)). In NN training, all example sets are calculated but logic behind calculation is the same.

3.1.1 Feed-forward computation

Feed forward computation or forward pass is two step process. First part is getting the values of the hidden layer nodes and second part is using those values from hidden layer to compute value or values of output layer. Input values of nodes N0,0 and N0,1 are pushed up to the network towards nodes in hidden layer (N1,0 and N1,1). They are multiplied with weights of connecting nodes and values of hidden layer nodes are



Pattern data for AND

n0,0	*n0,1*	*Output n2,0*
1	1	1
1	0	0
0	1	0
0	0	0

β = Learning rate = 0.45
α = Momentum term = 0.9

$f(x) = 1.0 / (1.0 + \exp(-x))$

Figure 4: Example

calculated. Sigmoid function is used for calculations $f(x) = 1.0/(1.0 + exp(-x))$.

$N1,0 = f(x1) = f(w0,0 * n0,0 + w0,1 * n0,1) = f(0.4 + 0.1) = f(0.5) = 0.622459$

$N1,1 = f(x2) = f(w0,2 * n0,0 + w0,3 * n0,1) = f(-0.1 - 0.1) = f(-0.2) = 0.450166$

When hidden layer values are calculated, network propagates forward, it propagates values from hidden layer up to a output layer node (N2,0). This is second step of feed forward computation

$N2,0 = f(x3) = f(w1,0 * n1,0 + w1,1 * n1,1) = f(0.06 * 0.622459 + (-0.4) * 0.450166) = f(-0.1427188) = 0.464381$

Having calculated N2,0, forward pass is completed.

3.1.2 Back propagation to the output layer

Next step is to calculate error of N2,0 node. From the table in figure 4, output should be 1. Predicted value (N2,0) in our example is 0.464381. Error calculation is done the following way:

$N2,0_Error = n2,0 * (1 - n2,0) * (N2,0_Desired - N2,0) = 0.464381(1 - 0.464381) * (1 - 0.464381) = 0.133225$

Once error is known, it will be used for backward propagation and weights adjustment. It is two step process. Error is propagated from output layer to the hidden layer first. This is where learning rate and momentum are brought to equation. So weights W1,0 and W1,1 will be updated first. Before weights can be updated, rate of change needs to be found. This is done by multiplication of the learning rate, error value and node N1,0 value.

$\Delta W1,0 = \beta * N2,0_Error * n1,0 = 0.45 * 0.133225 * 0.622459 = 0.037317$

Now new weight for W1,0 can be calculated.

$W1,0_New = w1,0_Old + \Delta W1,0 + (\alpha * \Delta(t-1)) = 0.06 + 0.037317 + 0.9 * 0 = 0.097137$

$\Delta W1,1 = \beta * N2,0_Error * n1,1 = 0.45 * 0.133225 * 0.450166 = 0.026988$

$W1,1_New = w1,1_Old + \Delta W1,1 + (\alpha * \Delta(t-1)) = -0.4 + 0.026988 = -0.373012$

The value of $\Delta(t-1)$ is previous delta change of the weight. In our example, there is no previous delta change so it is always 0. If next iteration were to be calculated, this would have some value.

3.1.3 Back propagation to the hidden layer

Now errors has to be propagated from hidden layer down to the input layer. This is bit more complicated than propagating error from output to hidden layer. In previous case, output from node N2,0 was known beforehand. Output of nodes N1,0 and N1,1 was unknown. Let's start with finding N1,0 error first. This will be calculated multiplying new weight W1,0 value with error for the node N2,0 value. Same way error

for N1,1 node will be found.

$N1,0_Error = N2,0_Error * W1,0_New = 0.133225 * 0.097317 = 0.012965$

$N1,1_Error = N2,0_Error * W1,1_New = 0.133225 * (-0.373012) = -0.049706$

Once error for hidden layer nodes is known, weights between input and hidden layer can be updated. Rate of change first needs to be calculated for every weight: $\Delta W0,0 = \beta * N1,0_Error * N0.0 = 0.45 * 0.012965 = 0.005834$

$\Delta W0,1 = \beta * N1,0_Error * n0,1 = 0.45 * 0.012965 * 1 = 0.005834$

$\Delta W0,2 = \beta * N1,1_Error * n0,0 = 0.45 * -0.049706 * 1 = -0.022368$

$\Delta W0,3 = \beta * N1,1_Error * n0,1 = 0.45 * -0.049706 * 1 = -0.022368$

Than we calculate new weights between input and hidden layer.

$W0,0_New = W0,0_Old + \Delta W0,0 + (\alpha * \Delta(t-1)) = 0.4 + 0.005834 + 0.9 * 0 = 0.405834$

$W0,1_New = w0,1_Old + \Delta W0,1 + (\alpha * \Delta(t-1)) = 0.1 + 0.005834 + 0 = 0.105384$

$W0,2_New = w0,2_Old + \Delta W0,2 + (\alpha * \Delta(t-1)) = -0.1 + -0.022368 + 0 = -0.122368$

$W0,3_New = w0,3_Old + *\Delta W0,3 + (\alpha * \Delta(t-1)) = -0.1 + -0.022368 + 0 = -0.122368$

3.1.4 Weight updates

Important thing is not to update any weights until all errors have been calculated. It is easy to forget this and if new weights were used while calculating errors, results would not be valid. Here is quick second pass using new weights to see if error has decreased.

$N1,0 = f(x1) = f(w0,0 * n0,0 + w0,1 * n0,1) = f(0.406 + 0.1) = f(0.506) = 0.623868314$

$N1,1 = f(x2) = f(w0,2 * n0,0 + w0,3 * n0,1) = f(-0.122 - 0.122) = f(-0.244) = 0.43930085$

$N2,0 = f(x3) = f(w1,0 * n1,0 + w1,1 * n1,1) = f(0.097 * 0.623868314 + (-0.373) * 0.43930085) = f(-0.103343991) = 0.474186972$

Having calculated N2,0, forward pass is completed.

Next step is to calculate error of N2,0 node. From the table in figure 4, output should be 1.

Predicted value (N2,0) in our example is 0.464381. Error calculation is done in following way.

$N2,0_Error = n2,0 * (1 - n2,0) * (N2,0_Desired - N2,0) = 0.474186972 * (1 - 0.474186972) * (1 - 0.474186972) = 0.131102901$

So after initial iteration, calculated error was 0.133225 and new calculated error is 0.131102. Our algorithm has improved, not by much but this should give good idea on how BP algorithm works. Although this was very simple example, it should help to understand basic operation of BP algorithm.

It can be said that algorithm learned through iterations. According to Dspguide.com [2010] number of iterations in typical NN would be any number from ten to ten thousands. This is only one example set pass that could be repeated many times until error is small enough.

4 Advantages and Disadvantages

Negnevitsky [2005] argued that turning point in quest for intelligent machines was when Kasparov, world chess champion was defeated by computer in New York in May 1997.

Artificial intelligence and NN have been used more and more in recent decades. Potentials in this area are huge. Here are some NN advantages, disadvantages and industries where they are being used.

NN are used in cases where rules or criteria for searching an answer is not clear (that is why NN are often called black box, they can solve the problem but at times it is hard to explain how problem was solved). They found its way into broad spectrum of industries, from medicine to marketing and military just to name few. Financial sector has been known for using NN in classifying credit rating and market forecasts. Marketing is another field where NN has been used for customer classification (groups that will buy some product, identifying new markets for certain products, relationships between customer and company). Many companies use direct marketing (sending its offer by mails) to attract customers. If NN could be employed to up the percentage of the response to direct marketing, it could save companies lot's of their revenue. At the end of the day, it's all about the money. Post offices are known to use NN for sorting the post (based on postal code recognition). Those were just few examples of where NN are being used. NN advantages are that they can adapt to new scenarios, they are fault tolerant and can deal with noisy data.

Time to train NN is probably identified as biggest disadvantage. They also require very large sample sets to train model efficiently. It is hard to explain results and what is going on inside NN.

5 Conclusion

NN is interconnected network that resembles human brain. The most important characteristic of NN is its ability to learn. When presented with training set (form of supervised learning) where input and output values are known, NN model could be created to help with classifying new data. Results that are achieved by using NN are encouraging, especially in some fields like pattern recognition. NN is getting more and more attention in last two decades. BP algorithm is most popular algorithm used in NN. It is one of the main reasons why NN are becoming so popular.

References

ComputingAssociate.com. *Introduction to Neural Networks*. Retrived from internet on 26 Oct 2010 http://euclid.ii.metu.edu.tr/ ion526/demo/chapter1/section1.1/figure1$_1$*e.html*, 2010.

Dspguide.com. *Introduction to Neural Networks*. Retrived from internet on 30 Oct 2010 http://www.dspguide.com/CH26.PDF, 2010.

Faqs.org. *Why use activation functions*. Retrived from internet on 30 Oct 2010 http://www.faqs.org/faqs/ai-faq/neural-nets/part2/section-10.html, 2010.

D.B. Fogel. *Blondie24: playing at the edge of AI*. The Morgan Kaufmann Series in Evolutionary Computation. Morgan Kaufmann Publishers, 2002. ISBN 9781558607835. URL `http://books.google.ie/books?id=5vpuw8L0`$_C$*AC*.

D.T. Larose. *Discovering knowledge in data: an introduction to data mining*. Wiley-Interscience, 2005. ISBN 9780471666578. URL `http://books.google.ie/books?id=JbPMdPWQIOwC`.

D. Michie, D.J. Spiegelhalter, and C.C. Taylor. *Machine learning, neural and statistical classification*. Artificial intelligence. Ellis Horwood, 1994. ISBN 9780131063600. URL `http://books.google.ie/books?id=QPqCQgAACAAJ`.

B. Muller, J. Reinhardt, and M.T. Strickland. *Neural networks: an introduction*. Number v. 1 in Physics of neural networks. Springer, 1995. ISBN 9783540602071. URL `http://books.google.ie/books?id=EFUzMYjOXk8C`.

Michael Negnevitsky. *Artificial intelligence: a guide to intelligent systems*. Pearson Education Limited, 2005.

Raul Rojas. *Neural Networks:A Systematic Introduction*. Springer, 2005.

Statsoft.com. *Statsoft*. Retrieved October 30, 2010, from http://www.statsoft.com/textbook/neural-networks/apps, 2010.

T. Vaga. *Profiting from chaos: using chaos theory for market timing, stock selection, and option valuation*. McGraw-Hill, 1994. ISBN 9780070667860. URL `http://books.google.ie/books?id=hjUMHEHpp38C`.