Introduction

Neural-networks is one of those words that is getting *fashionable* in the new era of technology. Most people have heard of them, but very few actually know what they are. This write-up is designed to introduce you to all the basics of neural networks ? their function, generic structure, terminology, types and uses.

The term *'neural network'* is in fact a biological term, and what we refer to as neural networks should really be called Artificial Neural Networks (ANNs). The two terms will be used interchangeably throughout the write-up , though. a real neural network is a collection of neurons, the tiny cells our brains are comprised of. A network can consist of a few to a few billion neurons connected in an array of different methods. ANNs attempt to model these biological structures both in architecture and operation. There is a small problem: we don't quite know how biological NNs work! Therefore, the architecture of neural networks changes greatly from type to type. What we do know is the structure of the basic neuron.

The Neuron

Although it has been proposed that there are anything between 50 and 500 different types of neurons in our brain, they are mostly just specialized cells based upon the basic neuron. The basic neuron consists of synapses, the soma, the axon and dendrites. Synapses are connections between neurons - they are not physical connections, but miniscule gaps that allow electric signals to jump across from neuron to neuron. These electrical signals are then passed across to the soma which performs some operation and sends out its own electrical signal to the axon. The axon then distributes this signal to dendrites. Dendrites carry the signals out to the various synapses, and the cycle repeats.

Just as there is a basic biological neuron, there is basic artificial neuron. Each neuron has a certain number of inputs, each of which have a *weight* assigned to them. The weights simply are an indication of how 'important' the incoming signal for that input is. The *net value* of the neuron is then calculated - the *net* is simply the weighted sum, the sum of all the inputs multiplied by their specific weight. Each neuron has its own unique threshold value, and it the *net* is greater than the threshold, the neuron fires (or outputs a 1), otherwise it stays quiet (outputs a 0). The output is then fed into all the neurons it is connected to.

Learning

As this talk about weights and thresholds leads to an obvious question. How are all these values set? There are nearly as many training methods as there are network types (a lot!), but some of the more popular ones include back-propagation, the delta rule and Kohonen learning.

As architectures vary, so do the learning rules, but most rules can be categorized into two areas - supervised and unsupervised. Supervised learning rules require a 'teacher' to tell them what the desired output is given an input. The learning rules then adjusts all the necessary weights (this can be very complicated in networks), and the whole process starts again until the data can be correctly analyzed by the network. Supervised learning rules include back-propagation and the delta rule. Unsupervised rules do not require teachers because they produce their own output which is then further evaluated.

Architecture

This area of neural networking is the "fuzziest" in terms of a definite set of rules to abide by. There are many types of networks - ranging from simple boolean networks (Perceptrons), to complex self-organizing networks (Kohonen networks), to networks modelling thermodynamic properties (Boltzmann machines)! There is, though, a standard network architecture.

The network consists of several "layers" of neurons, an input layer, hidden layers, and output layers. Input layers take the input and distribute it to the hidden layers (so-called *hidden* because the user cannot see the inputs or outputs for those layers). These hidden layers do all the necessary computation and output the results to the output layer, which (surprisingly) outputs the data to the user. Now, to avoid confusion, I will not explore the architecture topic further. To read more about different neural nets, see the Generation5 essays.

Even after discussing neurons, learning and architecture we are still unsure about what exactly neural networks do!

The Function of ANNs

Neural networks are designed to work with patterns - they can be classified as pattern classifiers or pattern associators. The networks can takes a vector (series of numbers), then classify the vector. For example, my ONR program takes an image of a number and outputs the number itself. Or my PDA32 program takes a coordinate and can classify it as either class A or class B (classes are determined by learning from examples provided). More practical uses can be seen in military radars where radar returns can be classified as enemy vehicles or trees (read more in the Applications in the Military essay).

Pattern associators takes one vector and output another. For example, my HIR program takes a 'dirty' image and outputs the image that represents the one closest to the one it has learnt. Again, at a more practical level, associative networks can be used in more complex applications such as signature/face/fingerprint recognition.

The Ups and Downs of Neural Networks

There are many good points to neural-networks and advances in this field will increase their popularity. They are excellent as pattern classifiers/recognizors - and can be used where traditional techniques do not work. Neural-networks can handle exceptions and abnormal input data, very important for systems that handle a wide range of data (radar and sonar systems, for example). Many neural networks are biologically plausible, which means they may provide clues as to how the brain works as they progress. Advances in neuroscience will also help advance neural networks to the point where they will be able to classify objects with the accuracy of a human at the speed of a computer! The future is bright, the present however...

Yes, there are quite a few down points to neural networks. Most of them, though, lie with our lack of hardware. The power of neural-networks lie in their ability to process information in a parallel fashion (that is, process multiple chunks of data simultaneously). Unfortunately, machines today are serial - they only execute one instruction at a time. Therefore, modelling parallel processing on serial machines can be a very time-consuming process. As with everything in this day and age, time is of the essence, which often leaves neural networks out of the list of viable solutions to a problem.

Other problems with neural networks are the lack of defining rules to help construct a network given a problem - there are many factors to take into consideration: the learning algorithm, architecture, number of neurons per layer, number of layers, data representation and much more. Again, with time being so important, companies cannot afford to invest to time to develop a network to solve the problem efficiently. This will all change as neural networking advances.

Conclusion

Hopefully, by now you have a good understanding of the basics of neural networks. Generation5 has recently had a lot of information added on neural networking, both in essays and in programs. We have examples of Hopfield networks, perceptrons (2 example programs), and even some case-studies on back-propagation. Please browse through the site to find out more!

A **neural network** is an interconnected group of artificial or biological neurons. It is possible to differentiate between two major groups of neural networks:

- Biological neural networks, for example the human brain or parts thereof.
- Artificial neural networks originally referred to electrical, mechanical or computational simulations or models of biological neural networks. The field has expanded so that some applications do not clearly resemble any existing biological counterpart.

In modern usage the term most often refers to artificial neural networks, especially in computer science and related fields. There exist hybrids, incorporating biological neurons as part of electronic circuits, so there is not always a clear delineation.

In general, a neural network is composed of a group of connected neurons. A single neuron can be connected to any other neurons so that the overall structure of the network can be very complex. Artificial intelligence and cognitive modeling try to simulate some properties of neural networks. Approaching human learning and memory is the main interest in these models. These artificial neural networks are advantageous, especially in in pattern recognition and classification tasks. They have found an application in the control of processes in the chemical industry, speech recognition, optical character recognition and adaptive software such as software agents (e.g. in computer games) and autonomous robots.

In some comparisons between the brain and computers the following calculation is made: There are billions of neurons in the human brain, estimates suggest roughly about $2 * 10^{12}$ neurons with individual differences. The relaxation time of these neurons is about 10 ms, this could amount to a processing speed of 100 Hz. The whole brain could therefore have a processing speed of roughly $2 * 10^{14}$ logical operations per second. These considerations are however very speculative. To compare, a PowerPC 970 at a frequency of 3 GHz and 64 bits (PowerPC) corresponds to $2 * 10^{11}$ logical operations per second in the case of the PowerPC. However, these comparison is very speculative, since the working of neural networks is not well understood. Perhaps the most fundamental difference between brain and computer is that today's computers operate sequentially (or occasionally with a small amount of parallelism possibly through such things as Hyper-threading and SIMD instructions like MMX & SSE2), while human brains are massively parallel.

Applications of neural networks

Character Recognition - The idea of character recognition has become very important as handheld devices like the Palm Pilot are becoming increasingly popular. Neural networks can be used to recognize handwritten characters.

Image Compression - Neural networks can receive and process vast amounts of information at once, making them useful in image compression. With the Internet explosion and more sites using more images on their sites, using neural networks for image compression is worth a look.

Stock Market Prediction - The day-to-day business of the stock market is extremely complicated. Many factors weigh in whether a given stock will go up or down on any given day. Since neural networks can examine a lot of information quickly and sort it all out, they can be used to predict stock prices.

Traveling Saleman's Problem - Interestingly enough, neural networks can solve the traveling salesman problem, but only to a certain degree of approximation.

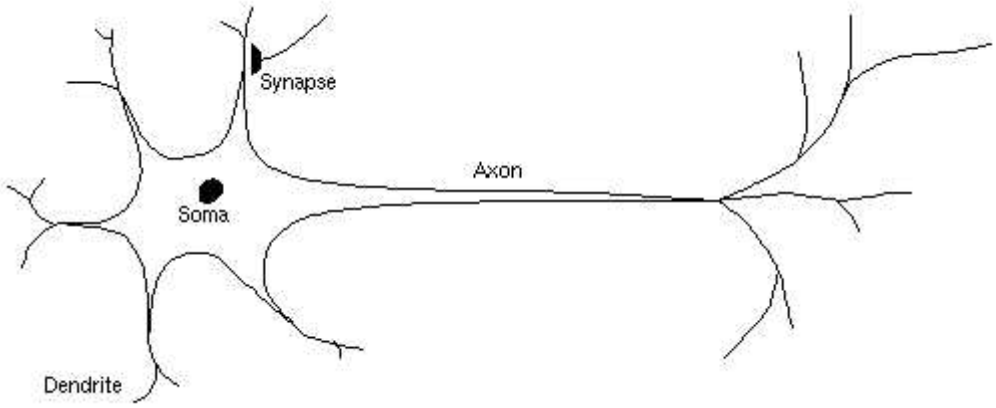
Medicine, Electronic Nose, Security, and Loan Applications - These are some applications that are in their proof-of-concept stage, with the acception of a

neural network that will decide whether or not to grant a loan, something that has already been used more successfully than many humans.

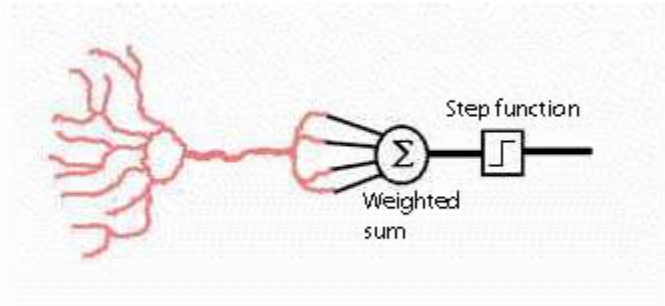
Miscellaneous Applications - These are some very interesting (albeit at times a little absurd) applications of neural networks.

The perceptron

The perceptron is a mathematical model of a biological neuron. While in actual neurons the dendrite receives electrical signals from the axons of other neurons, in the perceptron these electrical signals are represented as numerical values. At the synapses between the dendrite and axons, electrical signals are modulated in various amounts. This is also modeled in the perceptron by multiplying each input value by a value called the weight. An actual neuron fires an output signal only when the total strength of the input signals exceed a certain threshold. We model this phenomenon in a perceptron by calculating the weighted sum of the inputs to represent the total strength of the input signals, and applying a step function on the sum to determine its output. As in biological neural networks, this output is fed to other perceptrons.



(Fig. 1) A biological neuron



(Fig. 2) An artificial neuron (perceptron)

There are a number of terminology commonly used for describing neural networks. They are listed in the table below:

The input vector	All the input values of each perceptron are collectively called the input vector of that perceptron.
The weight vector	Similarly, all the weight values of each perceptron are collectively called the weight vector of that perceptron.

What can a perceptron do?

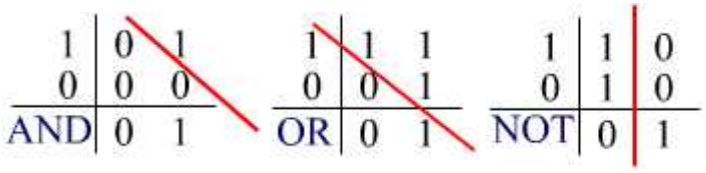
As mentioned above, a perceptron calculates the weighted sum of the input values. For simplicity, let us assume that there are two input values, x and y for a certain perceptron P. Let the weights for x and y be A and B for respectively, the weighted sum could be represented as: A x + B y.

Since the perceptron outputs an non-zero value only when the weighted sum exceeds a certain threshold C, one can write down the output of this perceptron as follows:

Output of P = {1 if A x + B y > C
{0 if A x + B y < = C

Recall that A x + B y > C and A x + B y < C are the two regions on the xy plane separated by the line A x + B y + C = 0. If we consider the input (x, y) as a point on a plane, then the perceptron actually tells us which region on the plane to which this point belongs. Such regions, since they are separated by a single line, are called linearly separable regions.

This result is useful because it turns out that some logic functions such as the boolean AND, OR and NOT operators are linearly separable Ð i.e. they can be performed using a single perceprton. We can illustrate (for the 2D case) why they are linearly separable by plotting each of them on a graph:



(Fig. 3) Graphs showing linearly separable logic functions

In the above graphs, the two axes are the inputs which can take the value of either 0 or 1, and the numbers on the graph are the expected output for a particular input. Using an appropriate weight vector for each case, a single perceptron can perform all of these functions.

However, not all logic operators are linearly separable. For instance, the 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

(Fig. 4) Since it is impossible to draw a line to divide the regions containing either 1 or 0, the XOR function is not linearly separable.

Mindpixel

From Wikipedia, the free encyclopedia.

Mindpixel is a World Wide Web based collaborative artificial intelligence project led by Chris McKinstry, a computer scientist and former Very Large Telescope operator for the European Southern Observatory in Chile. The project enlists the efforts of thousands of participants over the www and offers them a share in its value if it is successful. Hence its claim to be “The planet's largest artificial intelligence effort".

The project aims to create a database of millions of human validated true/false statements or probabilistic propositions. Participants create one line statements which will be objectively true or false to 20 other anonymous participants. In order to submit their statement they must first check the true/false validity of 20 such statements submitted by others. Participants who reply consistently out of step with the majority have their status downgraded and are eventually excluded. Likewise participants who make contributions which others cannot agree are objectively true or false have their status downgraded.

A validated true/ false statement is called a **mindpixel**.

Chris McKinstry believes that the database so constructed could be used in conjunction with a neural net to produce a body of human common sense knowledge which would have market value. Participants are awarded shares in any future value according to the number of mindpixels they have successfully created.

The database and its software is known as GAC (pronounced Jak) which stands for Generic Artificial Consciousness.

The project started in 2000 and has 1.4 million mindpixels as of January 2004. It was developed out of the earlier MISTIC (1996) AI project also created by Chris McKinstry.

An Introduction to Neural Networks Printable Version
By James Matthews

Notes on Neural Network Learning and Training Printable Version
By Wan Hussain Wan Ishak

1.0 Introduction

Neural Network (NN) could be define as an interconnected of simple processing element whose functionality is based on the biological neuron. Biological neuron (Figure 1) is a unique piece of equipment that carries information or a bit of knowledge and transfers to other neuron in a chain of networks. Artificial Neuron imitates these functions and their unique process of learning. Basically, biological neuron has three types of components called dendrites, soma and axon. Dendrites are the sensitive part of neuron that receive signal from other neuron. Soma calculates and sums the signals and transmitted to other cells through axon.

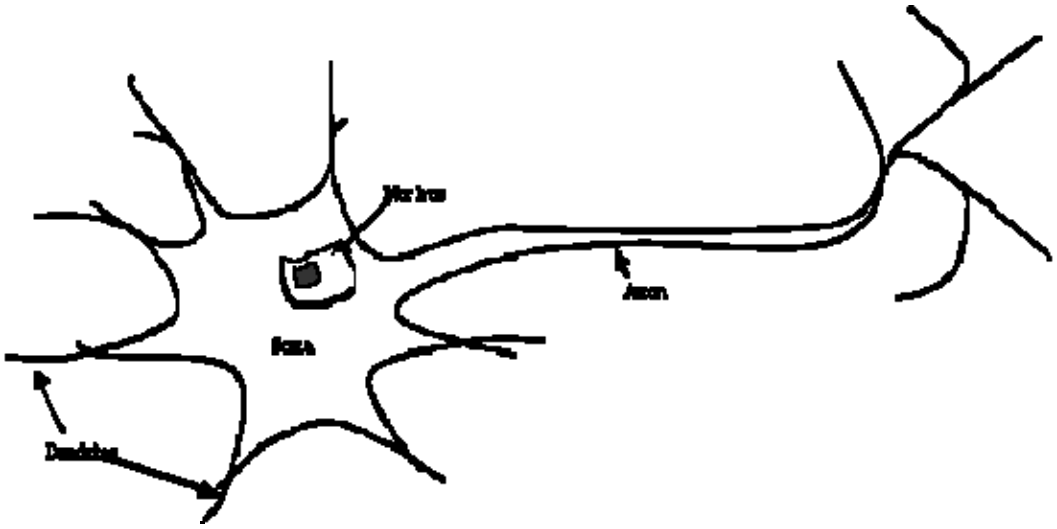


Figure 1: Biological Neuron

Simple neuron (Figure 2) introduced by McCulloch and Pitts in 1940s, consists of input layer, activation function, and output layer. Input layer receive input signal from external environment (or other neuron). Activation function is the neuron internal states that calculates and sum the input signals. The signals are then transmitted to output layer. The input layer, activation function and output layer in artificial neuron are similar to the function of dendrites, soma and axon in biological neuron.

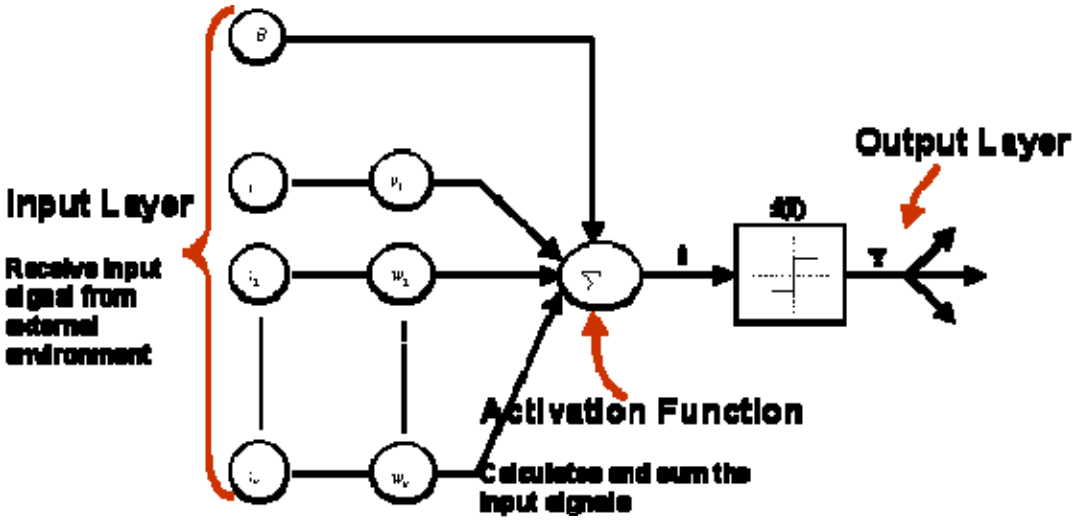


Figure 2: McCulloch-Pitts Neuron Model

2.0 Learning in Neural Network

Assume we have n input units, $X_i,?,X_n$ with input signals $x_1,?,x_n$. When the network receive the signals (x_i) from input units (X_i), the net input to output (y_in_j) is calculated by summing the weighted input signals ($\sum_{i=1}^{n} x_i w_{ij}$). The matrix multiplication method for calculating the net input is shown in the equation below.

$$y\_in_j = \sum_{i=1}^{n} x_i w_{ij}$$

where w_{ij} is the connection weights of input unit x_i and output unit y_j.

The network output (y_j) is calculated using the activation function $f(x)$. In which $y_j = f(x)$, where x is y_in_j. The computed weight from the training is stored and will become the information or knowledge for the future application.

NN can be divided into three architectures, namely single layer, multilayer network and competitive layer. The number of layers in a net is defined based on the number of interconnected weight in the neuron. Single layer network consists only one layer of connection weights. Whereas, multilayer networks consists of more than one layer of connection weights. The network also consists of additional layer called hidden layer. Multilayer networks can be used to solve more complicated problems compared to single layer network. Both of the network are also called feedforward network where the signal flows from the input units to the output units in a forward direction. The competitive layer network, for example the Recurrent Networks is a feedback network where there are closed-loop signal from a unit back to itself.

Learning Mechanisms

NNs learning algorithms can be divided into two main groups that are supervised (or Associative learning) and unsupervised (Self-Organisation) learning. Many supervised and unsupervised learning NN have been invented. Some are listed in NN FAQ (frequently-ask-question) and discussion group web page, but many other are not.

Supervised Learning

Supervised learning learns based on the target value or the desired outputs. During training the network tries to match the outputs with the desired target values. This method has two sub varieties called auto-associative and hetero-associative. In auto-associative learning, the target values are the same as the inputs, whereas in hetero-associative learning, the targets are generally different from the inputs.

One of the most commonly used supervised NN model is backpropagation network that uses backpropagation learning algorithm. Backpropagation (or backprop) algorithm is one of the well-known algorithms in neural networks. Backpropagation algorithm has been popularized by Rumelhart, Hinton, and Williams in 1980s as a euphemism for generalized delta rule. Backpropagation of errors or generalized delta rule is a decent method to minimize the total squared error of the output computed by the net (Fausett, 1994). The introduction of backprop algorithm has overcome the drawback of previous NN algorithm in 1970s where single layer perceptron fail to solve a simple XOR problem.

Unsupervised Learning

Unsupervised learning method is not given any target value. A desired output of the network is unknown. During training the network performs some kind of data compression such as dimensionality reduction or clustering. The network learns the distribution of patterns and makes a classification of that pattern where, similar patterns are assigned to the same output cluster. Kohonen network is the best example of unsupervised learning network. According to Sarle (1997) Kohonen network refers to three types of networks that are Vector Quantization, Self-Organizing Map and Learning Vector Quantization.

3.0 Training the Network

Training the network is time consuming. It usually learns after several epochs, depending on how large the network is. Thus, large network required more training time compared to the smaller one. Basically, the network is trained for several epochs and stopped after reaching the maximum epoch. For the same reason minimum error tolerance is used provided that the differences between network output and known outcome is less than the specified value (see for example Pofahl et al., 1998). We could also stop the training after the network meet certain stopping criteria.

During training the network might learn too much. This problem is referred to as

overfitting. Overfitting is a critical problem in most all standard NNs architecture. Furthermore, NNs and other AI machine learning models are prone to overfitting (Lawrence et al., 1997). One of the solutions is early stopping (Sarle, 1995), but this approach need more critical intention as this problem is harder than expected (Lawrence et al., 1997). The stopping criteria is also another issue to consider in preventing overfitting (Prechelt, 1998). Hence, for this problem during training, validation set is used instead of training data set. After a few epochs the network is tested with the validation data. The training is stopped as soon as the error on validation set increases rapidly higher than the last time it was checked (Prechelt, 1998). Figure 3 shows that the training should stop at time t when validation error starts to increase.

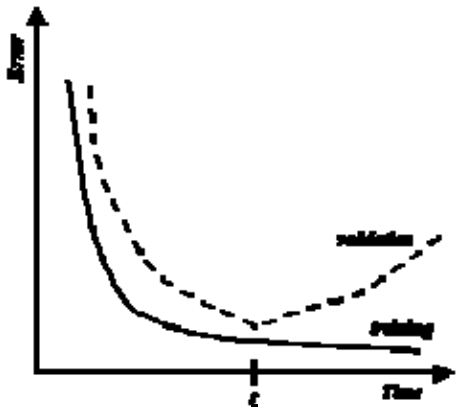


Figure 3: Training and validation curve

Discussion and Conclusion

Constructing a program for Neural Network is not a difficult task. Basically, it was only several steps of algorithms that are easily followed even by novice practitioners. However, preparing the network for training is a difficult task since the network dealing with a large amount of data. Another problem is when to stop the training? Over training could cause memorization where the network might simply memorize the data patterns and might fail to recognize other set of patterns. Thus, early stopping is recommended to ensure that the network learn accordingly.

References

Fausett, L. (1994). Fundamentals of Neural Network: Architectures, Algorithms and Applications. Englewood Cliffs: Prentice Hall.

Sarle, W. S. (1997). Neurak Network FAQ, part 1 of 7: Introduction. Periodic posting to the Usenet newsgroup comp.ai.neural-nets, URL: ftp://ftp.sas.com/pub/neural/FAQ.html Downloaded on 30 Nov. 1999.

Pofahl, W. E., Walczak, S. M., Rhone, E., and Izenberg, S. D. (1998). Use of an Artificial Neural Network to Predict Length of Stay in Acute Pancreatitis. American Surgeon, Sep98, Vol. 64 Issue 9, (pp: 868 ? 872)

Lawrence, S., Giles, C. L., and Tsoi, A. C. (1997). Lessons in Neural Network Training: Training May be Harder than Expected. Proceedings of the Fourteenth National Conference on Artificial Intelligence, AAAI-97, (pp. 540-545), Menlo Park, California: AAAI Press.

Sarle, W. (1995). Stopped Training and Other Remedies for Overfitting. Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics, (pp. 352-360). Retrieved March 18, 2002 from World Wide Web: ftp://ftp.sas.com/pub/neural/

Prechelt, L. (1998). Early Stopping-but when? Neural Networks: Tricks of the trade, (pp. 55-69). Retrieved March 28, 2002 from World Wide Web: http://wwwipd.ira.uka.de/~prechelt/Biblio/