Neuralyst™

*Neural Network Technology*

# Basic Concepts for Neural Networks

## Contents

- Real Neurons
- Neural Network Structure
- Neural Network Operation
- Neural Network Learning
- Appendix: Specific Formulae and Algorithms

**Note:** This document is an excerpt from the **Neuralyst$^{TM}$ User's Guide**, Chapter 3.

## Real Neurons

Let's start by taking a look at a biological neuron. Figure 1 shows such a neuron.

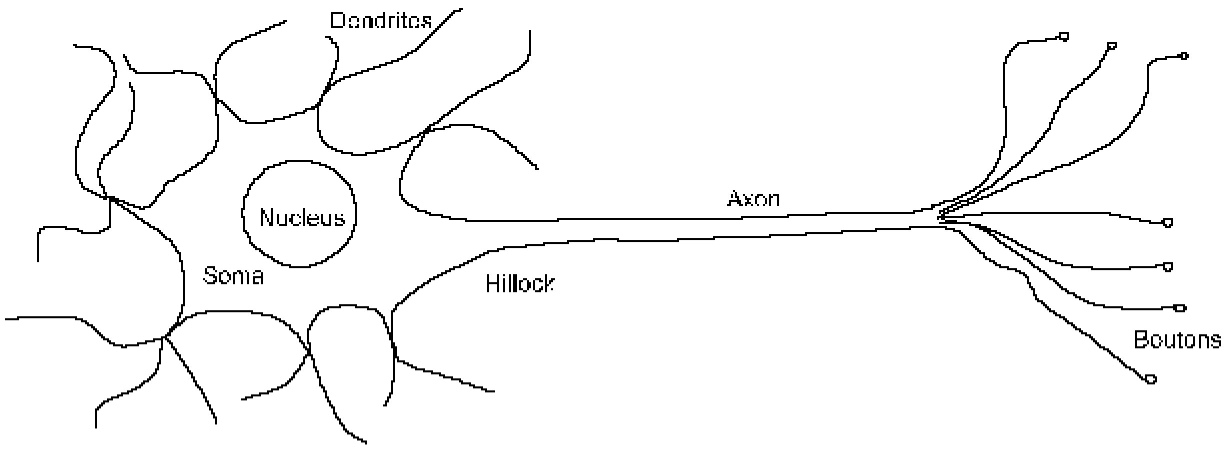

**Figure 1. A Biological Neuron**

A neuron operates by receiving signals from other neurons through connections, called *synapses*. The combination of these signals, in excess of a certain *threshold* or *activation* level, will result in the neuron *firing*, that is sending a signal on to other neurons connected to it. Some signals act as *excitations* and others as *inhibitions* to a neuron firing. *What we call thinking is believed to be the collective effect of the presence or absence of firings in the pattern of synaptic connections between neurons.*

This sounds very simplistic until we recognize that there are approximately one hundred billion (100,000,000,000) neurons each connected to as many as one thousand (1,000) others in the human brain. The massive number of neurons and the complexity of their interconnections results in a "thinking machine", your brain.

Each neuron has a body, called the *soma*. The soma is much like the body of any other cell. It contains the cell nucleus, various bio-chemical factories and other components that support ongoing activity.

Surrounding the soma are *dendrites*. The dendrites are receptors for signals generated by other neurons. These signals may be excitatory or inhibitory. All signals present at the dendrites of a neuron are combined and the result will determine whether or not that neuron will fire.

If a neuron fires, an electrical impulse is generated. This impulse starts at the base, called the *hillock*, of a long cellular extension, called the *axon*, and proceeds down the axon to its ends.

The end of the axon is actually split into multiple ends, called the *boutons*. The boutons are connected to the dendrites of other neurons and the resulting interconnections are the previously discussed synapses. (Actually, the boutons do not touch the dendrites; there is a small gap between them.) If a neuron has fired, the electrical impulse that has been generated stimulates the boutons and results in electrochemical activity which transmits the signal across the synapses to the receiving dendrites.

At rest, the neuron maintains an electrical potential of about 40-60 millivolts. When a neuron fires, an electrical impulse is created which is the result of a change in potential to about 90-100 millivolts. This impulse travels between 0.5 to 100 meters per second and lasts for about 1 millisecond. Once a neuron fires, it must rest for several milliseconds before it can fire again. In some circumstances, the repetition rate may be as fast as 100 times per second, equivalent to 10 milliseconds per firing.

Compare this to a very fast electronic computer whose signals travel at about 200,000,000 meters per second (speed of light in a wire is 2/3 of that in free air), whose impulses last for 10 nanoseconds and may repeat such an impulse immediately in each succeeding 10 nanoseconds continuously. Electronic computers have at least a 2,000,000 times advantage in signal transmission speed and 1,000,000 times advantage in signal repetition rate.

It is clear that if signal speed or rate were the sole criteria for processing performance, electronic computers would win hands down. What the human brain lacks in these, it makes up in numbers of elements and interconnection complexity between those elements. This difference in structure manifests itself in at least one important way; the human brain is not as quick as an electronic computer at arithmetic, but it is many times faster and hugely more capable at recognition of patterns and perception of relationships.

The human brain differs in another, extremely important, respect beyond speed; it is capable of "self-programming" or adaptation in response to changing external stimuli. In other words, it can learn. The brain has developed ways for neurons to change their response to new stimulus patterns so that similar events may affect future responses. In particular, the sensitivity to new patterns seems more extensive in proportion to their importance to survival or if they are reinforced by repetition.

---

## Neural Network Structure

Neural networks are models of biological neural structures. The starting point for most neural networks is a model neuron, as in Figure 2. This neuron consists of multiple inputs and a single output. Each input is modified by a *weight*, which multiplies with the input value. The neuron will combine these weighted inputs and, with reference to a threshold value and activation function, use these to determine its output. This behavior follows closely our understanding of how real neurons work.

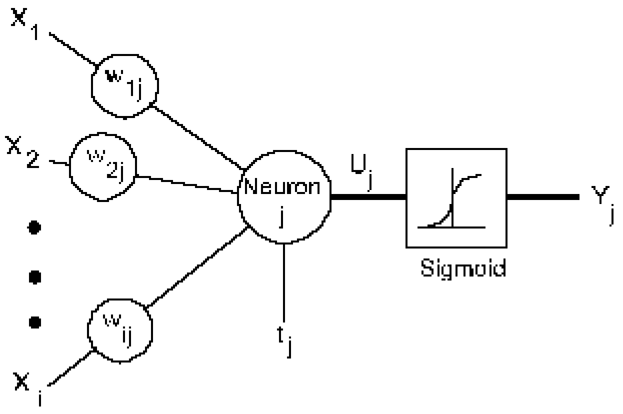

**FIgure 2. A Model Neuron**

While there is a fair understanding of how an individual neuron works, there is still a great deal of research and mostly conjecture regarding the way neurons organize themselves and the mechanisms used by arrays of neurons to adapt their behavior to external stimuli. There are a large number of experimental neural network structures currently in use reflecting this state of continuing research.

In our case, we will only describe the structure, mathematics and behavior of that structure known as the *backpropagation network*. This is the most prevalent and generalized neural network currently in use. If the reader is interested in finding out more about neural networks or other networks, please refer to the material listed in the bibliography.

To build a backpropagation network, proceed in the following fashion. First, take a number of neurons and array them to form a *layer*. A layer has all its inputs connected to either a preceding layer or the inputs from the external world, but not both within the same layer. A layer has all its outputs connected to either a succeeding layer or the outputs to the external world, but not both within the same layer.

Next, multiple layers are then arrayed one succeeding the other so that there is an input layer, multiple intermediate layers and finally an output layer, as in Figure 3. Intermediate layers, that is those that have no inputs or outputs to the external world, are called *>hidden layers*. Backpropagation neural networks are usually *fully connected*. This means that each neuron is connected to every output from the preceding layer or one input from the external world if the neuron is in the first layer and, correspondingly, each neuron has its output connected to every neuron in the succeeding layer.

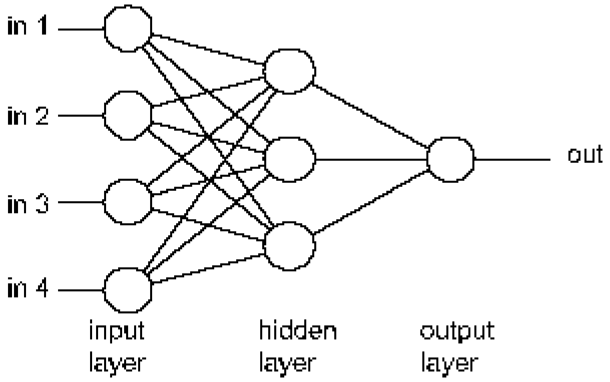

**Figure 3. Backpropagation Network**

Generally, the input layer is considered a distributor of the signals from the external world. Hidden layers are considered to be categorizers or feature detectors of such signals. The output layer is considered a collector of the features detected and producer of the response. While this view of the neural network may be helpful in conceptualizing the functions of the layers, you should not take this model too literally as the functions described may not be so specific or localized.

With this picture of how a neural network is constructed, we can now proceed to describe the operation of the network in a meaningful fashion.

## Neural Network Operation

The output of each neuron is a function of its inputs. In particular, the output of the ***j***th neuron in any layer is described by two sets of equations:

$$U_j = \sum (X_i \cdot w_{ij}) \text{[Eqn 1]}$$

and

$$Y_j = F_{th}(U_j + t_j) \text{[Eqn 2]}$$

For every neuron, ***j***, in a layer, each of the ***i*** inputs, $X_i$, to that layer is multiplied by a previously established weight, $w_{ij}$. These are all summed together, resulting in the internal value of this operation, $U_j$. This value is then biased by a previously established threshold value, $t_j$, and sent through an activation function, $F_{th}$. This activation function is usually the sigmoid function, which has an input to output mapping as shown in Figure 4. The resulting output, $Y_j$, is an input to the next layer or it is a response of the neural network if it is the last layer. Neuralyst allows other threshold functions to be used in place of the sigmoid described here.

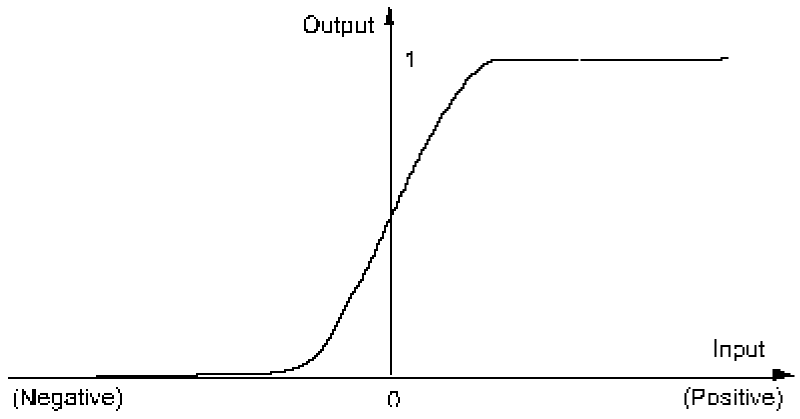

**Figure 4. Sigmoid Function**

In essence, Equation 1 implements the combination operation of the neuron and Equation 2 implements the firing of the neuron.

From these equations, a predetermined set of weights, a predetermined set of threshold values and a description of the network structure (that is the number of layers and the number of neurons in each layer), it is possible to compute the response of the neural network to any set of inputs. And this is just how Neuralyst goes about producing the response. But how does it learn?

## Neural Network Learning

Learning in a neural network is called *training*. Like training in athletics, training in a neural network requires a coach, someone that describes to the neural network what it should have produced as a response. From the difference between the desired response and the actual response, the *error* is determined and a portion of it is propagated backward through the network. At each neuron in the network the error is used to adjust the weights and threshold values of the neuron, so that the next time, the error in the network response will be less for the same inputs.

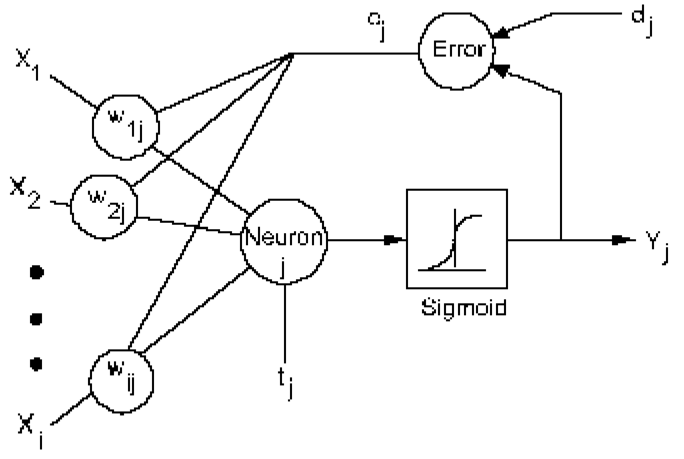

**Figure 5. Neuron Weight Adjustment**

This corrective procedure is called *backpropagation* (hence the name of the neural network) and it is applied continuously and repetitively for each set of inputs and corresponding set of outputs produced in response to the inputs. This procedure continues so long as the individual or total errors in the responses exceed a specified level or until there are no measurable errors. At this point, the neural network has learned the training material and you can stop the training process and use the neural network to produce responses to new input data.

[There is some heavier going in the next few paragraphs. Skip ahead if you don't need to understand all the details of neural network learning.]

Backpropagation starts at the output layer with the following equations:

$w_{ij} = w'_{ij} + LR \cdot e_j \cdot X_i$ [Eqn 3]

and

$e_j = Y_j \cdot (1 - Y_j) \cdot (d_j - Y_j)$ [Eqn 4]

For the ***i***th input of the ***j***th neuron in the output layer, the weight ***$w_{ij}$*** is adjusted by adding to the previous weight value, ***$w'_{ij}$***, a term determined by the product of a *learning rate*, ***LR***, an error term, ***$e_j$***, and the value of the ***i***th input, ***$X_i$***. The error term, ***$e_j$***, for the jth neuron is determined by the product of the actual output, ***$Y_j$***, its complement, ***$1 - Y_j$***, and the difference between the desired output, ***$d_j$***, and the actual output.

Once the error terms are computed and weights are adjusted for the output layer, the values are recorded and the next layer back is adjusted. The same weight adjustment process, determined by Equation 3, is followed, but the error term is generated by a slightly modified version of Equation 4. This modification is:

$e_j = Y_j \cdot (1 - Y_j) \cdot \sum (e_k \cdot w'_{jk})$ [Eqn 5]

In this version, the difference between the desired output and the actual output is replaced by the sum of the error terms for each neuron, ***k***, in the layer immediately succeeding the layer being processed (remember, we are going backwards through the layers so these terms have already been computed) times the respective pre-adjustment weights.

The learning rate, ***LR***, applies a greater or lesser portion of the respective adjustment to the old weight. If the factor is set to a large value, then the neural network may learn more quickly, but if there is a large variability in the input set then the network may not learn very well or at all. In real terms, setting the learning rate to a large value is analogous to giving a child a spanking, but that is inappropriate and counter-productive to learning if the offense is so simple as forgetting to tie their shoelaces. Usually, it is better to set the factor to a small value and edge it upward if the learning rate seems slow.

In many cases, it is useful to use a revised weight adjustment process. This is described by the equation:

$$w_{ij} = w'_{ij} + (1 - M) \cdot LR \cdot e_j \cdot X_j + M \cdot (w'_{ij} - w''_{ij})$$ [Eqn 6]

This is similar to Equation 3, with a *momentum* factor, ***M***, the previous weight, ***$w'_{ij}$***, and the next to previous weight, ***$w''_{ij}$***, included in the last term. This extra term allows for momentum in weight adjustment. Momentum basically allows a change to the weights to persist for a number of adjustment cycles. The magnitude of the persistence is controlled by the momentum factor. If the momentum factor is set to 0, then the equation reduces to that of Equation 3. If the momentum factor is increased from 0, then increasingly greater persistence of previous adjustments is allowed in modifying the current adjustment. This can improve the learning rate in some situations, by helping to smooth out unusual conditions in the training set.

As you train the network, the total error, that is the sum of the errors over all the training sets, will become smaller and smaller. Once the network reduces the total error to the limit set, training may stop. You may then apply the network, using the weights and thresholds as trained.

It is a good idea to set aside some subset of all the inputs available and reserve them for *testing* the trained network. By comparing the output of a trained network on these test sets to the outputs you know to be correct, you can gain greater confidence in the validity of the training. If you are satisfied at this point, then the neural network is ready for *running*.

Usually, no backpropagation takes place in this running mode as was done in the training mode. This is because there is often no way to be immediately certain of the desired response. If there were, there would be no need for the processing capabilities of the neural network! Instead, as the validity of the neural network outputs or predictions are verified or contradicted over time, you will either be satisfied with the existing performance or determine a need for new training. In this case, the additional input sets collected since the last training session may be used to extend and improve the training data.

---

http://www.cheshireeng.com/Neuralyst/nnbg.htm

**APPLICATIONS**

## *5.1 Transfer Function*

The behaviour of an ANN (Artificial Neural Network) depends on both the weights and the input-output function (transfer function) that is specified for the units. This function typically falls into one of three categories:

- linear (or ramp)
- threshold
- sigmoid

For **linear units**, the output activity is proportional to the total weighted output.

For **threshold units**, the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value.

For **sigmoid units**, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurones than do linear or threshold units, but all three must be considered rough approximations.

To make a neural network that performs some specific task, we must choose how the units are connected to one another (see figure 4.1), and we must set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence.

We can teach a three-layer network to perform a particular task by using the following procedure:

1. We present the network with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.
2. We determine how closely the actual output of the network matches the desired output.
3. We change the weight of each connection so that the network produces a better approximation of the desired output.

## *5.2 An Example to illustrate the above teaching procedure:*

Assume that we want a network to recognise hand-written digits. We might use an array of, say, 256 sensors, each recording the presence or absence of ink in a small area of a single digit. The network would therefore need 256 input units (one for each sensor), 10 output units (one for each kind of digit) and a number of hidden units.

For each kind of digit recorded by the sensors, the network should produce high activity in the appropriate output unit and low activity in the other output units.

To train the network, we present an image of a digit and compare the actual activity of the 10 output units with the desired activity. We then calculate the error, which is defined as the square of the difference between the actual and the desired activities. Next we change the weight of each connection so as to reduce the error.We repeat this training process for many different images of each different images of each kind of digit until the network classifies every image correctly.

To implement this procedure we need to calculate the error derivative for the weight (EW) in order to change the weight by an amount that is proportional to the rate at which the error changes as the weight is changed. One way to calculate the EW is to perturb a weight slightly and observe how the error changes. But that method is inefficient because it requires a separate perturbation for each of the many weights.
Another way to calculate the EW is to use the Back-propagation algorithm which is described below, and has become nowadays one of the most important tools for training neural networks. It was developed independently by two teams, one (Fogelman-Soulie, Gallinari and Le Cun) in France, the other (Rumelhart, Hinton and Williams) in U.S.

## *5.3 The Back-Propagation Algorithm*

In order to train a neural network to perform some task, we must adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights (**EW**). In other words, it must calculate how the error changes as each weight is increased or decreased slightly. The back propagation algorithm is the most widely used method for determining the **EW**.

The back-propagation algorithm is easiest to understand if all the units in the network are linear. The algorithm computes each **EW** by first computing the **EA**, the rate at which the error changes as the activity level of a unit is changed. For output units, the **EA** is simply the difference between the actual and the desired output. To compute the **EA** for a hidden unit in the layer just before the output layer, we first identify all the weights between that hidden unit and the output units to which it is connected. We then multiply those weights by the **EA**s of those output units and add the products. This sum equals the **EA** for the chosen hidden unit. After calculating all the **EA**s in the hidden layer just before the output layer, we can compute in like fashion the **EA**s for other layers, moving from layer to layer in a direction opposite to the way activities propagate through the network. This is what gives back propagation its name. Once the **EA** has been computed for a unit, it is straight forward to compute the **EW** for each incoming connection of the unit. The **EW** is the product of the EA and the activity through the incoming connection.

Note that for non-linear units, (see Appendix C) the back-propagation algorithm includes an extra step. Before back-propagating, the **EA** must be converted into the **EI**, the rate at which the error changes as the total input received by a unit is changed.

## 6. Applications of neural networks

### 6.1 Neural Networks in Practice

Given this description of neural networks and how they work, what real world applications are they suited for? Neural networks have broad applicability to real world business problems. In fact, they have already been successfully applied in many industries.

Since neural networks are best at identifying patterns or trends in data, they are well suited for prediction or forecasting needs including:

- sales forecasting
- industrial process control
- customer research
- data validation
- risk management
- target marketing

But to give you some more specific examples; ANN are also used in the following specific paradigms: recognition of speakers in communications; diagnosis of hepatitis; recovery of telecommunications from faulty software; interpretation of multimeaning Chinese words; undersea mine detection; texture analysis; three-dimensional object recognition; hand-written word recognition; and facial recognition.

### 6.2 Neural networks in medicine

Artificial Neural Networks (ANN) are currently a 'hot' research area in medicine and it is believed that they will receive extensive application to biomedical systems in the next few years. At the moment, the research is mostly on modelling parts of the human body and recognising diseases from various scans (e.g. cardiograms, CAT scans, ultrasonic scans, etc.).

Neural networks are ideal in recognising diseases using scans since there is no need to provide a specific algorithm on how to identify the disease. Neural networks learn by example so the details of how to recognise 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.

#### *6.2.1 Modelling and Diagnosing the Cardiovascular System*

Neural Networks are used experimentally to model the human cardiovascular system. Diagnosis can be achieved by building a model of the cardiovascular system of an individual and comparing it with the real time physiological measurements taken from the patient. If this routine is carried out regularly, potential harmful medical conditions can be detected at an early stage and thus make the process of combating the disease much easier.

A model of an individual's cardiovascular system must mimic the relationship among physiological variables (i.e., heart rate, systolic and diastolic blood pressures, and breathing rate) at different physical activity levels. If a model is adapted to an individual, then it becomes a model of the physical condition of that individual. The simulator will have to be able to adapt to the features of any individual without the supervision of an expert. This calls for a neural network.

Another reason that justifies the use of ANN technology, is the ability of ANNs to provide sensor fusion which is the combining of values from several different sensors. Sensor fusion enables the ANNs to learn complex relationships among the individual sensor values, which would otherwise be lost if the values were individually analysed. In medical modelling and diagnosis, this implies that even though each sensor in a set may be sensitive only to a specific physiological variable, ANNs are capable of detecting complex medical conditions by fusing the data from the individual biomedical sensors.

### *6.2.2 Electronic noses*

ANNs are used experimentally to implement electronic noses. Electronic noses have several potential applications in telemedicine. Telemedicine is the practice of medicine over long distances via a communication link. The electronic nose would identify odours in the remote surgical environment. These identified odours would then be electronically transmitted to another site where an door generation system would recreate them. Because the sense of smell can be an important sense to the surgeon, telesmell would enhance telepresent surgery.

### *6.2.3 Instant Physician*

An application developed in the mid-1980s called the "instant physician" trained an autoassociative memory neural network to store a large number of medical records, each of which includes information on symptoms, diagnosis, and treatment for a particular case. After training, the net can be presented with input consisting of a set of symptoms; it will then find the full stored pattern that represents the "best" diagnosis and treatment.

## 6.3 Neural Networks in business

Business is a diverted field with several general areas of specialization such as accounting or financial analysis. Almost any neural network application would fit into one business area or financial analysis.
There is some potential for using neural networks for business purposes, including resource allocation and scheduling. There is also a strong potential for using neural networks for database mining, that is, searching for patterns implicit within the explicitly stored information in databases. Most of the funded work in this area is classified as proprietary. Thus, it is not possible to report on the full extent of the work going on. Most work is applying neural networks, such as the Hopfield-Tank network for optimization and scheduling.

### *6.3.1 Marketing*

There is a marketing application which has been integrated with a neural network system. The Airline Marketing Tactician (a trademark abbreviated as AMT) is a computer system made of various intelligent technologies including expert systems. A feedforward neural network is integrated with the AMT and was trained using back-propagation to assist the marketing control of airline seat allocations. The adaptive neural approach was amenable to rule expression. Additionally, the application's environment changed rapidly and constantly, which required a continuously adaptive solution. The system is used to monitor and recommend booking advice for each departure. Such information has a direct impact on the profitability of an airline and can provide a technological advantage for users of the system. [Hutchison & Stephens, 1987]

While it is significant that neural networks have been applied to this problem, it is also important to see that this intelligent technology can be integrated with expert systems and other approaches to make a functional system. Neural networks were used to discover the influence of undefined interactions by the various variables. While these interactions were not defined, they were used by the neural system to develop useful conclusions. It is also noteworthy to see that neural networks can influence the bottom line.

### *6.3.2 Credit Evaluation*

The HNC company, founded by Robert Hecht-Nielsen, has developed several neural network applications. One of them is the Credit Scoring system which increase the profitability of the existing model up to 27%. The HNC neural systems were also applied to mortgage screening. A neural network automated mortgage insurance underwritting system was developed by the Nestor Company. This system was trained with 5048 applications of which 2597 were certified. The data related to property and borrower qualifications. In a conservative mode the system agreed on the underwriters on 97% of the cases. In the liberal model the system agreed 84% of the cases. This is system run on an Apollo DN3000 and used 250K memory while processing a case file in approximately 1 sec.

## 7. Conclusion

The computing world has a lot to gain from neural networks. Their ability to learn by example makes them very flexible and powerful. Furthermore there is no need to devise an algorithm in order to perform a specific task; i.e. there is no need to understand the internal mechanisms of that task. They are also very well suited for real time systems because of their fast response and computational times which are due to their parallel architecture.

Neural networks also contribute to other areas of research such as neurology and psychology. They are regularly used to model parts of living organisms and to investigate the internal mechanisms of the brain.

Perhaps the most exciting aspect of neural networks is the possibility that some day 'consious' networks might be produced. There is a number of scientists arguing that conciousness is a 'mechanical' property and that 'consious' neural networks are a realistic possibility.

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 intergrated with computing, AI, fuzzy logic and related subjects.

Source
http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html