SUPERVISOR'S REPORT

The supervisor should complete the report below and then give this cover, enclosing the final version of the extended essay, to the Diploma Programme coordinator. The supervisor must sign this report; otherwise the extended essay will not be assessed and may be returned to the school.

Name of supervisor (CAPITAL letters)

Comments

If appropriate, please comment on the candidate's performance, the context in which the candidate undertook the research for the extended essay, any difficulties encountered and how these were overcome. These comments can help the examiner award a level for criterion H. Do not comment on any adverse personal circumstances that may have affected the candidate.

I have read the final version of the extended essay that will be submitted to the examiner.

To the best of my knowledge, the extended essay is the authentic work of the candidate.

I spent [4] hours with the candidate discussing the progress of the extended essay.

Supervisor's signature:

ASSESSMENT FORM (for examiner use only)

Candidate session number	

		ACHIEVEMENT LEVEL		
		First examiner	maximum	Second examiner
General assessment criteria *Refer to the general guidelines.*	**A** Research question		2	
	B Approach		3	
	C Analysis/interpretation		4	
	D Argument/evaluation		4	
	E Conclusion		2	
	F Abstract		2	
	G Formal presentation		3	
	H Holistic judgement		4	
Subject assessment criteria *Refer to the subject guidelines. Not all of the following criteria will apply to all subjects; use only the criteria which apply to the subject of the extended essay.*	**J**			
	K			
	L			
	M			
	TOTAL OUT OF 36			

Name of first examiner (CAPITAL letters): Examiner number:

Name of second examiner (CAPITAL letters): _____ Examiner number:

Using Neural Networks in Medical Diagnosis

Word Count: 4000

How do neural networks work, and in what ways can they be applied to the diagnosis of patients' illnesses?

Abstract

The research question of this essay was: **how do neural networks work, and in what ways can they be applied to the diagnosis of patients' illnesses**? Thus, the aim of this project was to understand what neural networks are, to mathematically model how they work, and to examine how they can be applied, with a focus on real-world applications in the field of medical diagnosis. The scope of the essay was a focus on basic neural networks known as feedforward networks, using the Perceptron Learning Algorithm. Firstly, I described the background of how neural networks were conceptualized and their relation to the brain. Secondly, I described the structure, components and processes which make up a neural network, comparing them with conventional algorithms. Thirdly, I described how they function by representing neural networks mathematically and showing how they perform computations. Fourthly, I made use of past research to examine the rationale and ways of applying neural networks to diagnosing a patient's illness, a task traditionally performed by doctors and for which many algorithms in conventional computing have been designed. Finally, I concluded that neural networks are appropriate for very different purposes from conventional algorithms, and are especially appropriate for medical diagnosis due to their unique characteristics and the complexity that is present in many areas of medical diagnosis.

[Word Count: 215 words]

Contents

Introduction to my Extended Essay

I have always been fascinated by the consistency and accuracy by which doctors diagnose diseases. The diagnosis of illnesses based on symptoms is complex problem for which doctors apply their reasoning and understanding of the human body and often rely on their intuition and experience as well.

Last year, I designed a simple algorithm which diagnosed illnesses based on probabilistic calculations. However, this has many limitations, such as requiring very detailed and difficult to acquire prior information. Furthermore, it cannot adapt to changing times, diseases and probabilities or learn from experience, both of which are critical in providing accurate diagnosis. Thus, the brain is presently much more useful for diagnosis than such algorithms.

Neural networks are based on the information processing structure of the brain. A program structured according to a neural network, as I will describe, is adaptive and learns from experience. Thus, it could provide a better solution to the problem of medical diagnosis.

While I had initially planned to create a neural network for medical diagnosis, I eventually realized that the task of applying neural networks to medical diagnosis has already received a great deal of research through the years. Thus, one of the goals of this essay is to understand the opinions of experts in the medical scientific community to evaluate the extent to which neural networks are applicable and have been applied to medical diagnosis.

1. Background of Neural Networks

In this section, I will give a brief description of how neural networks were developed in relation to the brain. The way they work will be described in greater detail in later sections.

Our brains contain billions of tiny units known as neurons. Biologically, each neuron receives signals as electric current from other neurons through channels known as dendrites, which form its input. If the total current received exceeds a certain threshold value, the neuron 'fires' and transmits output as electric current to other neurons through channels known as axons.

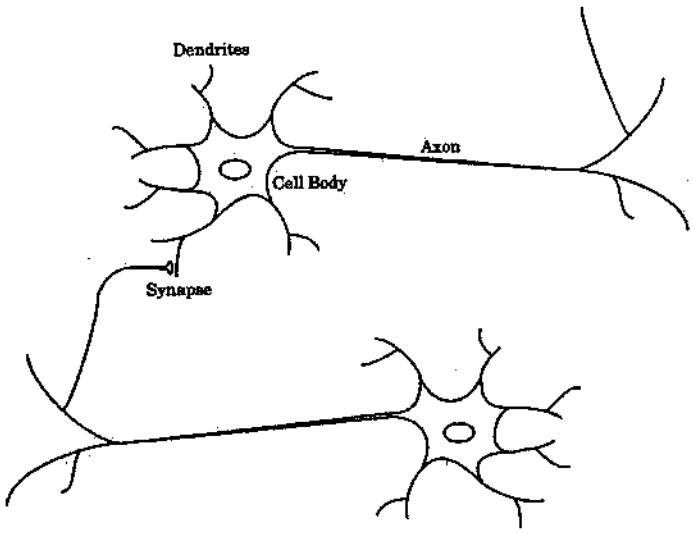


Figure 1.1 Schematic Drawing of Biological Neurons

Figure 1: Schematic Diagram of Biological Neurons

Source: Hagan, Demuth & Beale (1995)

The design of biological neurons led to computing algorithms known as 'artificial neural networks', commonly referred to as neural networks. A neural network consists of a network of units called neurons, which perform similar tasks to biological neurons: the neurons are individual programs which receive information, perform computations and then pass information through 'firing'.

Neural networks can be implemented as either hardware or software: each neuron can either be a separate processor, consisting of individual hardware forming a 'circuit board neural network', or the neural network can be a software simulation on a conventional computer, where the computer performs every neuron's tasks. At the current level of technology, the hardware required for circuit board neural networks is difficult to construct, so software simulations are more common. Thus, we will focus on software simulations of neural networks in this essay.

2. Structure of Neural Networks

The objective of both conventional algorithms and neural networks is to receive inputs, and after several steps, to arrive at the desired output. However, they perform this task using different methods. Thus, I will first describe how conventional algorithms process information.

2.1. Structure of Conventional Algorithms

Conventional algorithms receive a set of input, and repeatedly manipulate the data through several steps, until they arrive at the desired output.

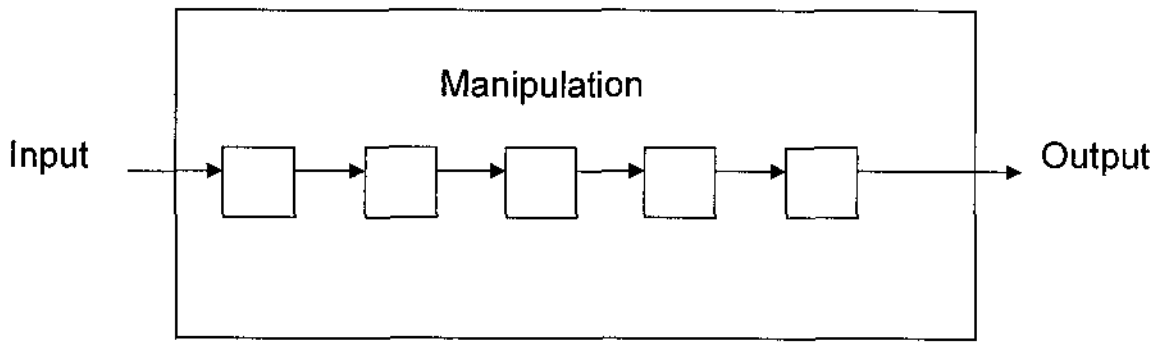


Figure 2: Information Processing in Conventional Algorithms

2.2 Structure of Neural Networks

We will refer to a simple but applicable form of neural networks known as 'feedforward' networks. Through the next few sections, we will illustrate concepts using an example of a text detection program that will detect the character "A": in fact, character detection is a common application of neural networks.

A neural network can contain a few to thousands of neurons, each of which is a program which perform certain tasks. These neurons are arranged in layers known as the 'input layer', the 'hidden layers' and the 'output layer'. In a feedforward network, information is passed in only one direction: from the input layer, to the hidden layers, to the output layer, or upwards in Figure 1.

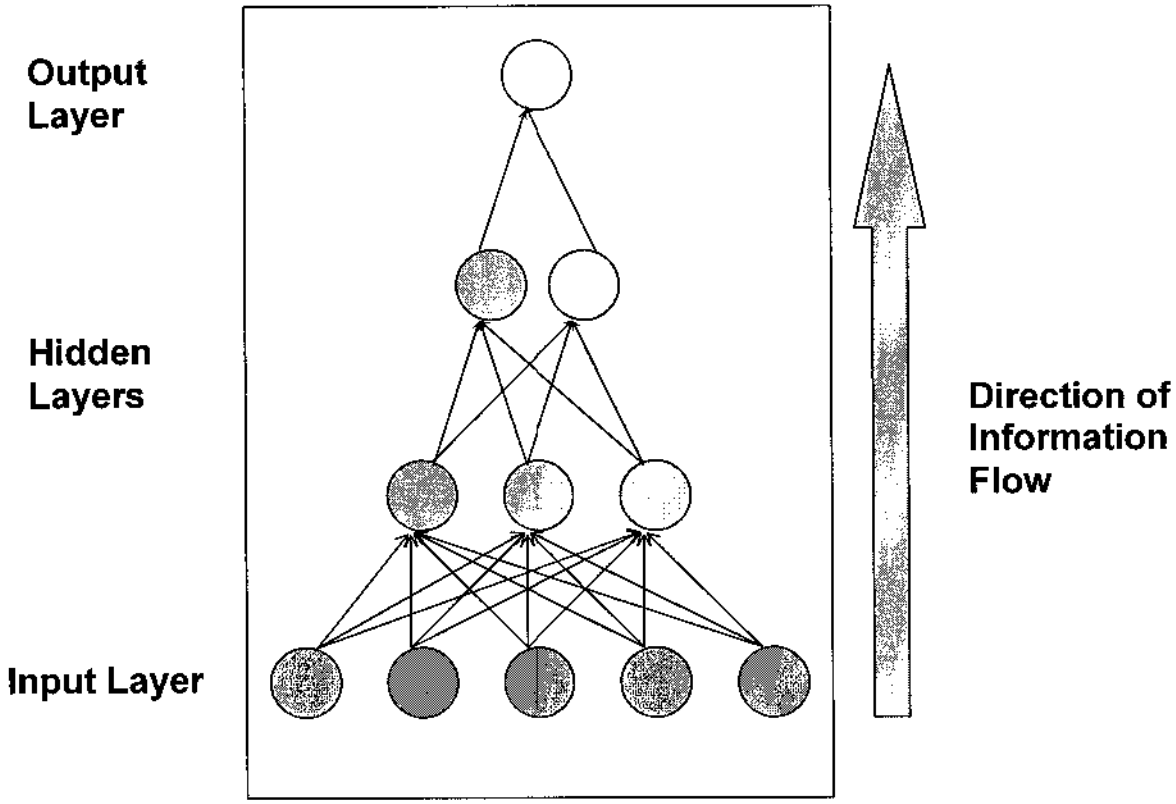


Figure 3: Information Processing in Neural Networks

The input layer neurons receive the input for a task. These neurons are programs which contain instructions for the computer to convert this input into signals: the neurons either fire or do not fire based on the input received.

In our example, the program receives an image and is to determine whether it represents an "A":

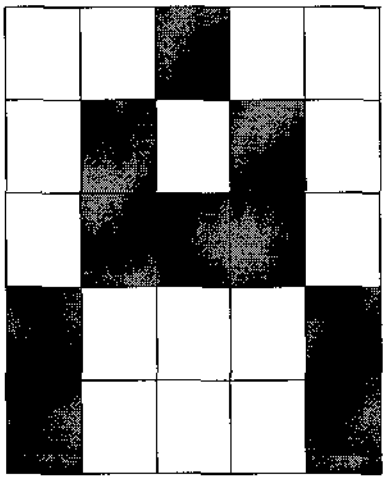


Figure 4: Example of Input Received by a Neural Network

To convert the image into a signal, we have 25 input neurons each corresponding to a single grid cell. To indicate the colour of a cell, the corresponding neuron fires exactly when the cell is red.

Next, the hidden layer neurons are each connected to a certain set of input layer neurons. They receive signals from input neurons they are connected to, combine these signals together in a certain way, and fire if the result is greater than their threshold value.

In our example, one hidden layer neuron searches for the horizontal line in the "A": it thus receives signals from input neurons corresponding to

the centre of the grid. It combines these together in a certain way, and if the result is large enough, it concludes that the horizontal line is present and fires. Other hidden layer neurons could search for other features of the "A".

Finally, the output layer neurons each correspond to an output that the network gives. They combine signals received from the hidden layer neurons, and either fire or do not fire based on these.

In our example, since the program only has to tell whether the character is an "A", there is only one output layer neuron which decides whether the features of the "A" present indicated by the hidden neurons are enough to conclude that the character is an "A", and fires to indicate an "A".

3. Characteristics of Neural Networks

Next, I will explain how the structures of neural networks lead to their unique characteristics.

Firstly, neural networks adapt and can learn from experience. This is because the structure of neural networks allows for an efficient algorithm of adaptation known as the Perceptron Learning Rule, which we will explain in a later section. In comparison, conventional algorithms are generally more suited to performing exactly the tasks they are programmed to.

Secondly, neural networks are suited to inputs affected by unwanted deviations from usual, also known as 'noise'. In a character recognition program, such deviations could be caused by dirt or specks on the paper before scanning, resulting in unwanted specks in the scanned image. Returning to our example, input noise could refer to a missing red square in the 'A':

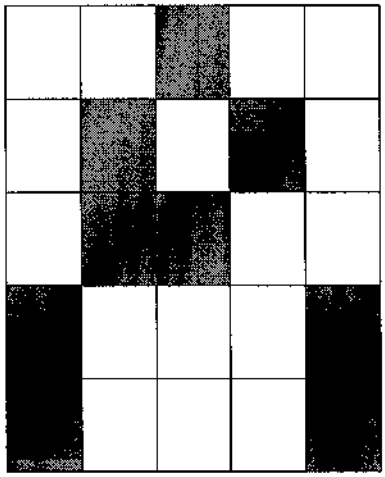


Figure 4: Example of 'Noisy' Input Received by a Neural Network

As the deviation is small, the features of the 'A' that are present make up for the deviation: the horizontal line through the "A" would probably be deemed present due to the other two present red squares in this line. Otherwise, due to the other present features of the 'A', the output neuron would still probably output a 'yes' answer. Thus, minor errors and faults are unlikely to affect the result.

We can summarize the differences between conventional computing and neural networks in this table:

Conventional Algorithm	**Neural network**
Well defined tasks	Adaptive, learn from experience
Suited to exact input	Work well despite messy, noisy input

4. Applications of Neural Networks

As we have seen, neural networks can generalize known examples to new cases. However, they are not suitable for straightforward computations and precise input. As a result, neural networks are fairly specialized. We focus on two of their common applications: character recognition and geological exploration.

4.1 Character recognition

As in our example, neural networks can be used to identify scanned handwritten or printed text. Neural networks are applied to this task because of their ability to learn as well as cope with noise.

Most neural networks work in a similar way to the "A" detection program in our example above. According to Houle et al. (1998), the programs convert a character image into a grid, which input neurons convert to signals. For increased accuracy, instead of just recognizing coloured and uncoloured squares, the neural networks assign to each coloured square a vector defining the direction that the character is being written at that point.

Using repeated examples and a learning algorithm similar to one we will introduce, the neural networks are trained such that hidden neurons recognize features of letters and output neurons decides which letter is present. An example of such a program, according to Mitek Systems (2005), is the 'QuickStrokes API' computer program, a software emulation of a neural network which recognizes characters from many languages and is used by financial institutions and government organizations worldwide.

In a recent study conducted by Chellapilla et al. (2005), neural network based algorithms performed better than humans at single character recognition.

4.2 Oil and Mineral Exploration

Neural networks are used to detect the locations of oil, mineral and natural gas beneath the sea bed. To determine locations to drill for oil, prospectors consider changes in the Earth's magnetic and gravitational fields, stratigraphy, optical noise and other factors. The exact relation between these factors and the probability of finding oil is complex and not fully

understood, which explains why a neural network, known as an 'expert system' is used for their adaptive nature.

Using the 'Artificial Foreteller of Oil and Mineral Deposits' neural network developed and detailed by Associative Neuron Systems (2005) as an example, the neural network first receives data on features near known oil and mineral sites. It then creates a function, or 'regression', representing the relationship between geological features and the probability of oil sites being present. It then predicts whether oil and minerals are present at a new site, given its geological features. According to Balch et al. (2002), neural networks generated maps of the Delaware Basin showing likely oil sites.

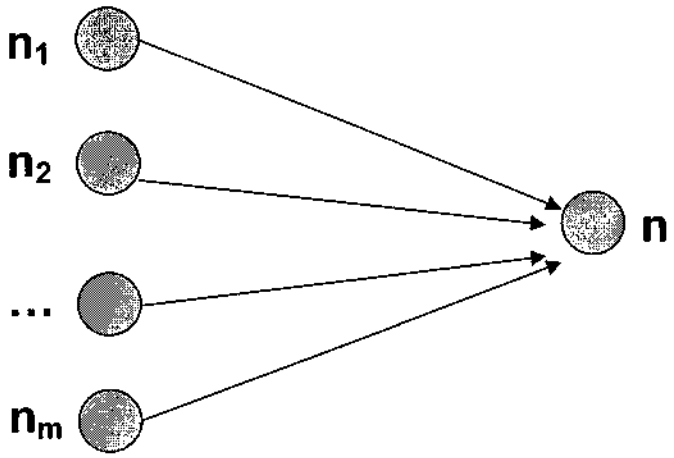
5. Functional Aspect of Neural Networks

5.1: Mathematical Model

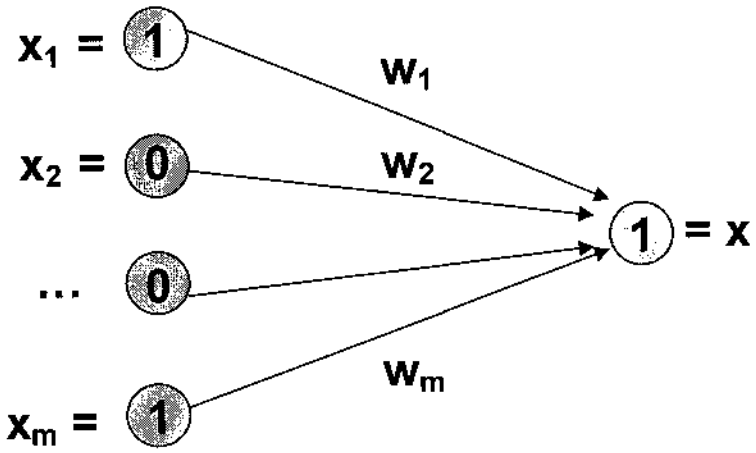
Neural networks were first mathematically modeled by McCulloch and Pitts (1943). While most real world neural network contain hidden layers, for simplicity, we use a neural network consisting of several input neurons connected to a single output neuron.

A neural network can be represented by a graph with directed, weighted edges, whose vertices represent neurons, edges represent connections between neurons, and weights represent the strength of a connection between two neurons. These weights are used to sum up inputs of firing input neurons. Each neuron has 2 possible states: 'firing' and 'not firing', which we denote using '1' and '0' respectively. The output neuron has a threshold, and if its sum computed exceeds its threshold, it fires. To represent this algebraically, we denote:

- The set of input neurons n_1, n_2, ... n_m, and n, the output neuron.



- x_k as the state of n_k: 1 if it is firing and 0 if not,
- x as the state of the output neuron n;
- θ, the threshold value of n;
- w_k as the weight assigned to the connection from n_k to n;



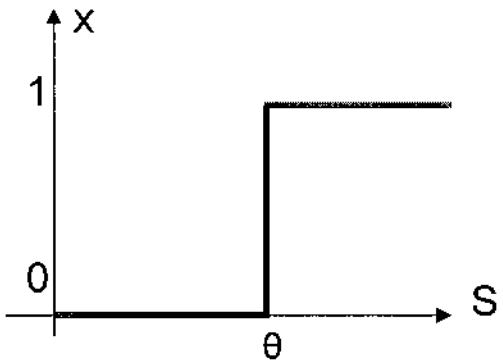
To decide whether to fire, n evaluates S, the sum of weights assigned to each connection between firing neurons and itself, and fires if S exceeds θ.

$$x = 1 \text{ if } S = w_1 x_1 + w_2 x_2 + \ldots + w_i x_i \geq \theta$$

$$x = 0 \text{ if } S = w_1 x_1 + w_2 x_2 + \ldots + w_i x_i < \theta$$

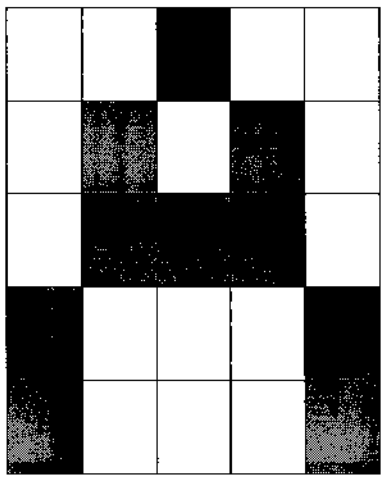
In effect, in our mathematical model, the output neuron x can be seen as a function, with inputs x_1, x_2, ..., x_m, and output x. Its weights w_i determine how it adds its inputs, while its threshold θ determines when it fires, so these are its 'parameters'. Finally, the sum obtained, S, determines a

yes or no answer, x, the final output of the function. The function relating S to x can be called a step function: the output is 0 for $S < \theta$ and 1 for $S \geq \theta$.



5.2 Algorithms for Learning

As we have seen, a property of neural networks is that they learn from examples. Returning to our text recognition example, the program starts with random weights, placing random importance on each cell. Given the following pattern, it may well output a "no" answer:



The correct answer would be "yes"; after making the mistake, the output neuron adjusts its parameters to place more importance on the red cells in this example. Thus, it gradually adjusts towards the optimal set of parameters.

The exact algorithm of adjustment is known as the Perceptron Learning Rule, and was developed by Rosenblatt (1962). Under this algorithm, if the student's answer is 0 while the teacher's answer is 1, this signals that the sum S for this set of input is too low and the threshold θ too high. Recalling that $S = w_1 x_1 + w_2 x_2 + \ldots + w_i x_i$, S is affected by only the weights w_i corresponding to firing input neurons, S being too low shows that these weights are too low: we adjust these weights by adding 1 to each,

while leaving the weights corresponding to the neurons which do not fire under this set of input unchanged. A convenient and compact way of representing this is to increase each weight w_i by x_i, ensuring that a weight w_i corresponding to a firing neuron ($x_i = 1$) increases by 1 while a weight w_i corresponding to a nonfiring neuron ($x_i = 0$) remains unchanged. Also, since the threshold θ is too high, we decrease it by 1.

Conversely, if the student's answer x is, instead, 1 when it should have been 0, we perform the reverse operation: decrease each weight w_i by x_i, and increase θ by 1. These operations have an exactly reverse effect of the previous case.

This algorithm can be described as follows:

If $x = t$,	do nothing;
If $x = 1, t = 0$,	change $w_i \rightarrow w_i - x_i$, $\theta \rightarrow \theta+1$
If $x = 0, t = 1$,	change $w_i \rightarrow w_i + x_i$, $\theta \rightarrow \theta-1$

Block (1962) first proved that given enough examples, a Perceptron system will eventually find the parameters w_i and θ if they exist, which

would give the correct answer in all cases. As this proof is complicated and is in the article itself, we instead give an outline of the standard proof of this result.

First, we represent the variables using vectors and the Perceptron algorithm using vector equations. We denote the vector containing the weights (w_1, w_2, …, w_m) by **w**; we additionally denote the threshold θ as w_0 to add it to this vector, so that

$$\mathbf{w} = (w_0, w_1, \ldots, w_m) = (\theta, w_1, \ldots, w_m).$$

We next denote the vector containing the values of x_i by **x**; we also add an additional value $x_0 = -1$, resulting in:

$$\mathbf{x} = (x_0, x_1, \ldots, x_m) = (-1, x_1, \ldots, x_m).$$

As a result, we have, using the rules of vector dot product:

$$\mathbf{w} \cdot \mathbf{x} = w_1x_1 + w_2x_2 + \ldots + w_mx_m - \theta$$

Recall that the neuron fires exactly when this expression, $\mathbf{w} \cdot \mathbf{x}$, is positive or zero.

For the Perceptron system to find a perfect set of parameters w_i, this perfect set of parameters must exist; we denote the vector for this perfect set of parameters by $\mathbf{w}^*$, including the threshold θ in the 0^{th} position. The teacher's answer must be 1 exactly when the sum computed using these parameters exceeds the threshold, or, for every x:

$$t = 1 \text{ if } \mathbf{w}^* \cdot \mathbf{x} \geq 0$$

$$t = 0 \text{ if } \mathbf{w}^* \cdot \mathbf{x} < 0$$

We can also represent the Perceptron Learning Rule using vectors, where we denote the new, modified vector of weights, modified from $\mathbf{w}$, by $\mathbf{w(1)}$:

$$\mathbf{w(1)} = \mathbf{w} + (t - x)\mathbf{x}.$$

This is equivalent to the previous formulation of the algorithm: when the neural network's answer is 0 when it should have been 1, we have t - x = 1, so the weights connected to input neurons which fired increase by 1, while the threshold $\theta = w_0$ decreases by 1; the opposite is true when the answer is 1 when it should have been 0. When the answers are the same, t - x = 0, so the parameters w_i and θ remain unchanged.

Let the result after k iterations by **w(k)**. We show that **k** has an upper limit, proving that the algorithm must end after a finite number of steps.

Using the equation above for **w(1)**, we express **w(1) . w*** and $|\mathbf{w'}|^2$ by multiplying by **w*** and squaring respectively, in terms of **w**, **w***, t and **x**,. With the substitution x + t = 1, which applies whenever x and t differ, we can also eliminate x from the equation. After some computations, we find that **w(1) . w*** must always be greater than its predecessor **w . w***, and furthermore, greater than this by at least a fixed amount. Similarly, $|\mathbf{w'}|^2$ must be less than its predecessor $|\mathbf{w}|^2$, and also less by at least a fixed amount. Dividing, after the kth step, we get the following expression:

$$\frac{w(k).w^*}{|w(k)|^2}$$

As we have seen, the above expression increases with k by a fixed amount at each step. However, an inequality known as the Cauchy-Schwarz Inequality states that the magnitude of the dot product of two vectors is at most the product of their magnitudes, and after some computations, we eventually find that this condition places an upper limit on the expression above. This means that the process must terminate in a finite number of steps, and will terminate with $w(k) = w^*$ which means the neural network has obtained the optimal set of parameters **w**.

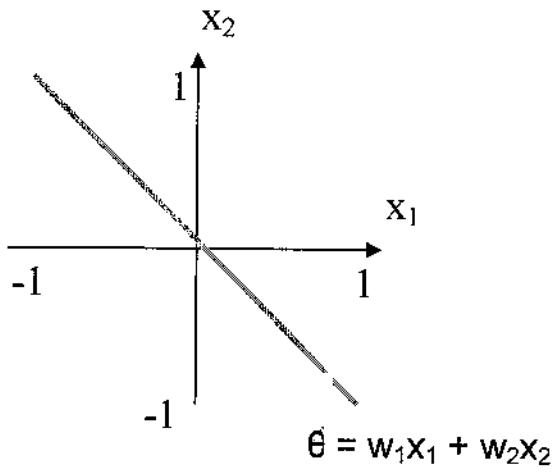
Next, we will discuss some limitations of the Perceptron Learning Rule. If only the output neurons adapt, the correct set of parameters may not exist to define certain functions. This was shown by Minsky and Papert (1969). An example is the Either/Or logical function or XOR function, which takes in 2 inputs and gives one output according to the table:

Output of XOR Function

Input 1

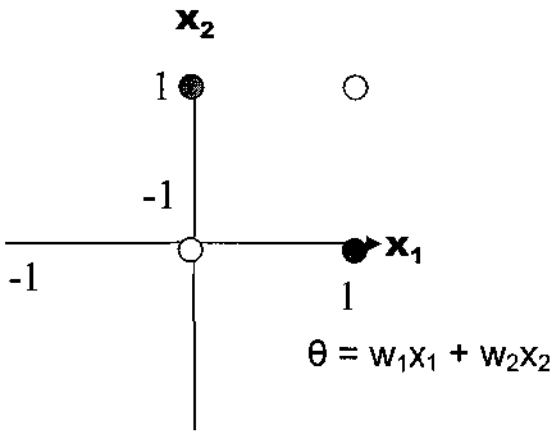
Input 2		**1**	**0**
	1	0	1
	0	1	0

We will show that there is no set of w_1, w_2, θ which gives output according to the XOR function. Assume instead that a set of w_1, w_2, θ actually exists. With 2 inputs x_1 and x_2, we can plot on the graph of x_1 against x_2 a straight line representing the equation $\theta = S = w_1x_1 + w_2x_2$. For example, if $w_1=w_2 = 1$, $\theta = 0$, the graph becomes:



On this graph, points on the line represent points at which $w_1x_1 + w_2x_2 = \theta$; points above or on the line represent points at which $w_1x_1 + w_2x_2 \geq \theta$. The neuron fires exactly when $w_1x_1 + w_2x_2 \geq \theta$, the threshold value, thus it fires exactly when x_1, x_2 define a point on or above the line. Note that this reasoning assumes that w_1 and w_2 are positive. The important point, however, is that in all cases the line divides the plane into 2 half-planes; the 2 half-planes contain points where the neuron fires and points where it does not.

However, to represent the XOR function, the output neuron must fire at the red points and not at the white points:



However, it is impossible to draw a straight line through the graph such that all red and white circles lie on opposite sides of the line. We conclude that it is impossible to represent the XOR function using any set of parameters (w_1, w_2, θ).

Thus, to represent the XOR function, a neural network must have multiple layers (including hidden layers), and to learn, it should adjust the parameters of neurons in multiple layers. While we have shown an algorithm for the output neuron to learn, constructing an algorithm to allow both the output neurons and the hidden neurons to learn is much more difficult - it was previously speculated by two researchers, Minsky and Papert, (1969) that for a neural network with multiple layers of neurons, no algorithm exists to allow both the output neuron and the hidden neurons to learn.

The problem of allowing the hidden neurons to learn was later solved using the Back Propagation algorithm, which was developed by Werbos (1974) and is commonly used in neural networks today. In this algorithm, when an output neuron makes a mistake, both this neuron and the hidden neurons connected to it are adjusted. The Back Propagation algorithm allows for more powerful neural networks which can solve more problems: it was

proven by Hornik, Stinchcombe and White (1989) for continuous functions and by Sontag (1992) for discontinuous functions, that with 3 or more layers of learning neurons (i.e. a hidden layer and 2 output layer neurons learning) a neural network can approximate any desired relationship between input and output.

6. Applications of Neural Networks to Medical Diagnosis

As earlier mentioned, one of my interests in undertaking this project was to understand researchers' views of the ways in which neural networks can be applied to medical diagnosis.

In this section, I will examine how the features of neural networks, which arise from the structure of neural networks as we have explained, apply to medical diagnosis. In fact, neural networks are used and are well suited to diagnosis due to, as explained by Dybowski and Grant (2001), complexity arising from the uncertainty in the exact relation between diseases and symptoms.

Neural networks are used in medical diagnosis in two main ways: Interpretation of complex figures and diagrams, and the diagnosis of a patient's illness based on symptoms.

6.1 Interpretation of Complex Figures

Neural networks are used in this task due to their ability to extract patterns from complex figures. An example is the PAPNET screening system, which, according to Koss et al. (1994), analyzes "pap smear" test images to check for cervical cancer. They explain how the system works: cells with large and misshapen nuclei can signify cancer, but the difference between harmless or 'benign' and cancerous or 'malignant' cervical cells can be subtle and testing is usually performed by cell biologists or cytologists. The neural network is first trained by examining cancerous and benign cell samples. Then, it works alongside cytologists: after viewing cell images, the neural network converts these into signals, after which the hidden and output neurons analyze the data for the essential features of 'benign' and 'malignant' cells. In unassisted visual screening by cytologists, the error rate of diagnosis is up to one third, but the PAPNET system is able to identify abnormal smears with 97 percent accuracy.

6.2 Diagnosis of illnesses based on symptoms

By understanding which diseases are likely to result in which symptoms, neural networks are able to decide the most likely illness that a patient experiences based on their symptoms. Neural networks are used due to the uncertain relationships between illness and symptoms, resulting in a task suited to a neural network's ability to learn. Zhang and Lin (1999) described an experimental-stage system known as the Virtual Physician aimed at diagnosing thyroid illnesses based on symptoms. The Virtual Physician is a software simulation of a Back Propagation neural network system which was trained using sample symptoms sets as input, and the appropriate one of twelve thyroid illnesses as output. In tests, the neural network was shown to be significantly more accurate than human physicians in thyroid illness diagnosis.

Conclusion

In conclusion, neural networks allow for unique algorithms which have the special characteristics of learning from examples, recognizing patterns and applying these to new cases. As a result, neural networks are specialized and only suited for certain tasks. For tasks which require the analysis of complex and imprecise data in often non-straightforward ways, neural networks present the benefits that come from learning and improving themselves. On the other hand, conventional algorithms are more suitable for computational and arithmetic tasks. In this way, neural networks do not compete with conventional algorithms but complement them. In the area of medical diagnosis, neural networks are suitable for and have been applied in the interpretation of complex figures and to diagnose illnesses based on symptoms. However, due to the very different nature of neural networks and conventional algorithms, we have yet to see the extent to which neural networks can coordinate with conventional algorithms to produce even greater efficiency and accuracy in medical diagnosis.

10. McCulloch, W. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Cambridge: MIT Press.

11. Minsky, M. & Papert, S. (1969). An Introduction to Computational Geometry. Cambridge: MIT Press.

12. Mitek Systems Inc. (2005). Quickstrokes.. Available from World Wide Web: URL:http://www.miteksystems.com/content/products_quickstrokes.htm

13. Rosenblatt, F. (1962). Principles of Neurodynamics. New York: Spartan Books.

14. Sontag, E. (1992). Feedback Stabilization using Two-Hidden Layer Nets. *IEEE Transactions on Neural Networks* 3(6): pp. 981-990.

15. Werbos, P. (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph. D. thesis. Cambridge: Harvard U. Committee on Applied Mathematics.

16. Zhang, H. & Lin, F. (1999) Medical Diagnosis by the Virtual Physician. Washington, D.C.: IEEE Computer Society. *Proceedings of the 12th IEEE Symposium on Computer-Based Medical Systems:* pp. 296.