CHAPTER 1

OVERVIEW OF NEURAL NETWORKS

Introduction

Throughout the years, the computational changes have brought growth to new technologies. Such is the case of artificial neural networks, that over the years, they have given various solutions to the industry.

Designing and implementing intelligent systems **has become a crucial factor for the innovation and development of better products for society**. Such is the case of the implementation of **artificial life** as well as giving solution to interrogatives that linear systems are not able resolve.

A neural network is a parallel system, capable of resolving paradigms that linear computing cannot. A particular case is for considering which I will cite. During summer of 2006, an intelligent crop protection system was required by the government. This system would protect a crop field from season plagues. The system consisted on a flying vehicle that would inspect crop fields by flying over them.

Now, imagine how difficult this was. Anyone that could understand such a task would say that this project was designated to a multimillionaire enterprise capable of develop such technology. Nevertheless, it wasn't like that. The selected company was a small group of graduated engineers. Regardless the lack of experience, the team was qualified. The team was divided into 4 sections in which each section was designed to develop specific sub-systems. The leader was an electronics specialist. She developed the electronic system. Another member was a mechanics and hydraulics specialist. He developed the drive system. The third member was a system engineer who developed all software, and the communication system. The last member was designed to develop all related to avionics and artificial intelligence.

Everything was going fine. When time came to put the pieces together, all fitted perfectly until they find out the robot had no knowledge about its task. What happened? The one designated to develop all artificial intelligent forgot to "teach the system". The solution would be easy; however, training a neural network required additional tools. The engineer designated to develop the intelligent system passed over this inconvenient.

History of Neural Networks

The study of the human brain dates back thousands of years. But it has only been with the dawn of modern day electronics that man has begun to try and emulate the human brain and its thinking processes. The modern era of neural network research is credited with the work done by neuro-physiologist, Warren McCulloch and young mathematical prodigy Walter Pitts in 1943. McCulloch had spent 20 years of life thinking about the "event" in the nervous system that allowed to us to think, feel, etc. It was only until the two joined forces that they wrote a paper on how neurons might work, and they designed and built a primitive artificial neural network using simple electric circuits. They are credited with the McCulloch-Pitts Theory of Formal Neural Networks. (Haykin, 1994, pg: 36) (http://www.helsinki.fi)

The next major development in neural network technology arrived in 1949 with a book, "The Organization of Behavior" written by Donald Hebb. The book supported and further reinforced McCulloch-Pitts's theory about neurons and how they work. A major point brought forward in the book described how neural pathways are strengthened each time they were used. As we shall see, this is true of neural networks, specifically in training a network. (Haykin, 1994, pg: 37)(http://www.dacs.dtic.mil)

During the 1950's traditional computing began, and as it did, it left research into neural networks in the dark. However certain individuals continued research into neural networks. In 1954 Marvin Minsky wrote a doctorate thesis, "Theory of Neural-Analog Reinforcement Systems and its Application to the Brain-Model Problem", which was concerned with the research into neural networks. He also published a scientific paper entitled, "Steps Towards Artificial Intelligence" which was one of the first papers to discuss Artificial Intelligence in detail. The paper also contained a large section on what nowadays is known as neural networks. In 1956 the Dartmouth Summer Research Project on Artificial Intelligence began researching Artificial Intelligence, what was to be the primitive beginnings of neural network research. (http://www.dacs.dtic.mil)

Years later, John von Neumann thought of imitating simplistic neuron functions by using telegraph relays or vacuum tubes. This led to the invention of the von Neumann machine. About 15 years after the publication of McCulloch

and Pitt's pioneer paper, a new approach to the area of neural network research was introduced. In 1958 Frank Rosenblatt, a neuro-biologist at Cornell University began working on the Perceptron. The perceptron was the first "practical" artificial neural network. It was built using the somewhat primitive and "ancient" hardware of that time. The perceptron is based on research done on a fly's eye. The processing which tells a fly to flee when danger is near is done in the eye. One major downfall of the perceptron was that it had limited capabilities and this was proven by Marvin Minsky and Seymour Papert's book of 1969 entitled, "Perceptrons". (http://www.dacs.dtic.mil) (Masters, 1993, pg: 4-6)

Between 1959 and 1960, Bernard Wildrow and Marcian Hoff of Stanford University, in the USA developed the ADALINE (ADAptive LINear Elements) and MADALINE (Multiple ADAptive LINear Elements) models. These were the first neural networks that could be applied to real problems. The ADALINE model is used as a filter to remove echoes from telephone lines. The capabilities of these models were again proven limited by Minsky and Papert (1969). (http://www.dacs.dtic.mil).

The period between 1969 and 1981 resulted in much attention towards neural networks. The capabilities of artificial neural networks were completely blown out of proportion by writers and producers of books and movies. People believed that such neural networks could do anything, resulting in disappointment when people realized that this was not so. Asimov's television series on robots highlighted humanity's fears of robot domination as well as the moral and social implications if machines could do mankind's work. Writers of best-selling novels like "Space Oddesy 2001" created fictional sinister computers. These factors contributed to large-scale critique of Artificial Intelligence and neural networks, and thus funding for research projects came to a near halt. (Haykin, 1994, pg: 38) (http://www.dacs.dtic.mil)

An important aspect that did come forward in the 1970's was that of self-organizing maps (SOM's). Self-organizing maps will be discussed later in this project (Haykin, 1994, pg: 39). In 1982 John Hopfield of Caltech presented a paper to the scientific community in which he stated that the approach to Artificial Intelligence should not be to purely imitate the human brain but instead to use its concepts to build machines that could solve dynamic problems. He showed what such networks were capable of and how they would work. It was his articulate, likeable character and his vast knowledge of mathematical analysis that convinced scientists and researchers at the National Academy of Sciences to renew interest into the research of Artificial Intelligence and neural networks. His ideas gave birth to a new class of neural networks that over time became known as the Hopfield Model (http://www.dacs.dtic.mil) (Haykin, 1994, pg: 39).

At about the same time at a conference in Japan about neural networks, Japan announced that they had again begun exploring the possibilities of neural networks. The United States feared that they would be left behind in terms of research and technology and almost immediately began funding for AI and neural network projects (http://www.dacs.dtic.mil).

1986 saw the first annual Neural Networks for Computing conference that drew more than 1800 delegates. In 1986 Rumelhart, Hinton and Williams reported back on the developments of the back-propagation algorithm. The paper discussed how back-propagation learning had emerged as the most popular learning set for the training of multi-layer perceptrons. With the dawn of the 1990's and the technological era, many advances into the research and development of artificial neural networks are occurring all over the world. Nature itself is living proof that neural networks do in actual fact work. The challenge today lies in finding ways to electronically implement the principles of neural network technology. Electronics companies are working on three types of neuro-chips namely, digital, analog, and optical. With the prospect that these chips may be implemented in neural network design, the future of neural network technology looks very promising.

1.1 BIOLOGICAL VS. ELECTRICAL BRAINS

Biological and electrical brains are very similar in some aspects and very different in others. One of the main similarities is the modeling of neurons. A biological brain is a collection of individual neurons that send electric pulses to one another based on reactions and pulses perceived. An electrical brain is very similar in which “nodes” send electrical signals to one another through electric wires. These pulses then enact different responses in the various neurons influenced by them. A minor difference does lie in the pulses though, while a biological brain can vary the electric pulse in amplitude, most electric brains are stuck inside a specific voltage range.

The big difference between a biological brain and an electric brain is the ability of the biological brain to radically alter its structure while learning. A human brain on the other hand, learns by re-arranging the structure of the brain. A neuron in a human brain can alter its paths and electric charge to affect those around it. On the other hand, an electric brain must store information and give certain paths different weights. Paths never change or disappear, inside they grow stronger or weaker. Simply put, a biological brain’s hardware can change, while neural networks hardware cannot. Learning in general is a very hard concept for the electrically brain to grasp. This occurs because a set of guidelines and rules must be laid out for the electronic brains so it understands what must be remember and what must be committed to memory.

1.1.1 LAYERS

There are 3 main layers in a neural network. The first of these layers is the input layer. In the input layer, data is gathered from external sources. These external sources can be sensors, manually inputted data, or data generated by other neural networks or the same network. The input layer then passes the data to a hidden layer. Because the input layer accepts data, it often acts as a buffer for the hidden layer. The hidden layer serves two functions. The first function is processing the data. Here equations are solved, or answers are formulated. The second function of the hidden layer is to determine what is learned and what is forgotten. Here the learning rules and laws are applied, and the "structure" of the neural network is updated. Lastly, there is the output layer. The output layer is where the processing and data meet the external world. The output layer could be a set of lights, or a computer screen, even a voice synthesizer. This layer just like the input layer is one way. This one way creates a buffer that can protect the more sensitive hidden layers.

1.1.2 COMMUNICATIONS

An important aspect in a neural network is how the neurons communicate with one another. There are three different types of communication. The first is inter-layer connections. These are the communication lines used to communicate this from one layer to the next. When using this type of communication, the sending layer cannot vary by from the receiving layer by more than a difference of one. In other words, this means layer 1 can talk to only layer 0 or 2, using inter-layer communications.

The next type of communications is the intra-layer communications. This is where neurons within the same layer communicate with one another. Besides, neurons talking to other neurons, there are self-connections, which are when a neuron talks to itself. These connections are considered a special type of intra-layer communications. Lastly, there is the supra layer communications. These are the communications that occur when a neuron needs to talk to another neuron or layer that is more than a difference of 1 away. For example, layer 1 sending a message to layer 5, would be considered a supra layer communication. All three of these communications are required for a successful neural network.

Communications between neurons and layers are often weighted.

1.1.3 INTER-LAYER

Taking a more in-depth look at inter-layer communication, it can be seen that there are two different types of connections. The first type of connection is the full connection. A full connection is one that tries to maximize the number of

connections between neurons. A specific definition cannot be developed, due to the fact that learning methods influence, which connections are valid and which ones are not. Three variations are possible with full connections. The most common is the fully interlayer-connected network. This communication method is where all possible connections are present between layers, but no intra or supra layer connections exist (Figure 1.2). Another method, which many people think of when fully connected is mentioned is the plenary neural network. This network communication method has all possible connections, including those found in the intra and supra layers (Figure 1.2). Lastly, the third method is a plenary network that doesn't employ self-connections. This reduction in self-connections, increases the speed of the network, but does have an impact on the ability of neurons to learn.

1.2 INTRODUCTION TO BIOLOGICAL NEURON

Artificial Neural Networks (ANNs) are computer systems (software[1] or hardware) that are biologically inspired in that they attempt to simulate the processing capabilities of the networks of **neurons** in the human brain.

The number of applications of ANNs to real world problems is immense and they permeate industry and commerce. ANNs perform particularly well where there is a large amount of historical data, where the application involves recognizing patterns in the data or where the problem is one of classification. ANNs fit into the general area of **computational intelligence**[2] and rank alongside **fuzzy logic** as the most successful.

The biological inspiration for ANNs is motivated by the fact that the human brain is capable of so much when compared with a computer. Although computers can very quickly process numbers and carry out complex mathematical calculations they are unable to reason in a similar manner to human beings. So, the suggestion is that we should simulate some of the physical processes that provide the ability to reason and tackle difficult problems that computers allied to mathematical techniques are unable to solve well. The basic premise is that we should borrow from nature.

The eventual aim is to emulate the way human beings think. This is still only a pipe dream but currently ANNs are able to tackle complex real world problems that mathematical and statistical techniques are often unable to handle. In particular ANNs are able to deal with 'noisy' data and have strong generalisation capabilities. Noisy data is real world data that does not, for

[1] The vast majority of applications are software implementations of ANNs.

[2] Computational Intelligence is a term used to cover three techniques – artificial neural networks, fuzzy logic and genetic algorithms – that attempt to imbue computers with some 'intelligence'.

example, fit exactly to a mathematical function but contains random variations. Generalisation is the ability to handle examples that the network hasn't seen before. We are still unsure exactly how the brain operates but we understand enough to enable us to mimic the basic operations and interactions between neurons. The human brain is made up of approximately 1×10^{11} neurons with of the order of 1×10^{15} connections between them. Figure 1.1 provides a schematic diagram for a biological neuron and the connections between neurons.

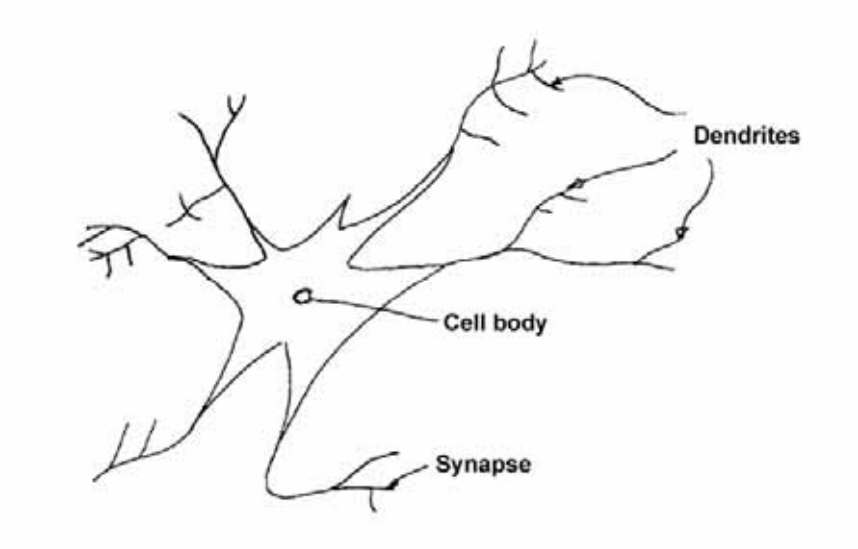

Figure 1.1 The Biological Neuron (Wasserman, 1989).

Each neuron in the brain possesses the capability to take electrochemical signals, as input, and process them before sending new signals via the connections between neurons, known as the dendrites. The cell body receives these signals at the synapse. When the cell body receives the signals they are summed – some signals excite the cells whilst others will inhibit the cell. On exceeding a threshold, a signal is sent via the dendrites to other cells. It is this receiving of signals and summation procedure that is emulated by the artificial neuron.

The biological neuron is only an inspiration for the artificial neuron. The artificial neuron is clearly a simplification of the way the biological neuron operates and indeed much is still not known about the way the brain operates for us to carry the analogy too far. ANNs, then, are an approach for modelling a simplification of the biological neuron – we are not modelling the brain but merely using it as an inspiration.

We can summarise this section in the following way.

- ANNs learn from data, are biologically inspired and are based on a network of artificial neurons;
- ANNs handle noisy data and are able to generalise in that they can cope with examples of problems that they have not experienced before.

1.3 ARTIFICIAL NEURON

The artificial neuron is the basic building block of ANNs (from now on we will use neuron to mean artificial neuron). There are a number of variations on this basic neuron but they all have the same simple design. Figure 2.2 shows the basic structure.

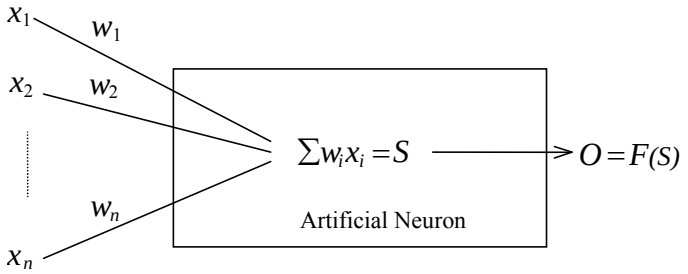

Figure 1.2 The Basic Artificial Neuron.

Each neuron receives inputs, x_1, x_2, …, x_n, which are connected to the input side of the neuron. Attached to every connection is a weight w_i which represents the connection strength for that input.

The cell node then calculates the weighted sum of the inputs given by

$$S = \sum_{i=1}^{n} w_i x_i$$

An **activation function,** F, takes the signal, S, as input to produce the output, O, of the neuron. In other words

$$O = F(S)$$

There are a number of functions that could be employed. For example it may simply be a threshold

$$O = \begin{cases} 1 \text{ where } S > T \\ 0 \text{ otherwise} \end{cases} \qquad \text{…..(1.1)}$$

where T is a constant threshold. This can be interpreted to mean that if the weighted sum is above T then the node ‘fires’.

Another commonly adopted activation function is the logistic or sigmoidal function given by

$$O = \frac{1}{1 + e^{-S}} \quad (-\infty < S < \infty) \qquad \text{.....(1.2)}$$

This has the effect of limiting the output of the neuron to a minimum of zero and a maximum of S. Figure 2.3 shows the effect of applying the sigmoidal function.

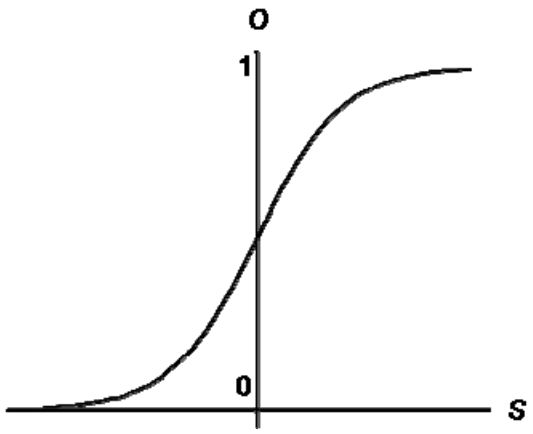

Figure 1.3 The Sigmoidal Activation Function.

As you can see, this has the effect of compressing the value of O to between zero and one. This function also has the effect of introducing nonlinearity into the network.

There are a number of other functions that are employed in real applications. Choosing an activation function is one of the many decisions faced by a neural network developer.

In this basic description of a neuron, we have already come across the notion of a weight. It's not too simplistic to say that the problem of developing an ANN is primarily to find a method for learning or estimating these weights. You will see this more clearly later.

1.3.1 THE PERCEPTRON

Neural networks research started in the 1940s when McCulloch and Pitts (1943) introduced the idea of the perceptron. The original perceptron consisted of a single **layer** of neurons that had a threshold activation function as described above. An example is shown in Figure 1.4.

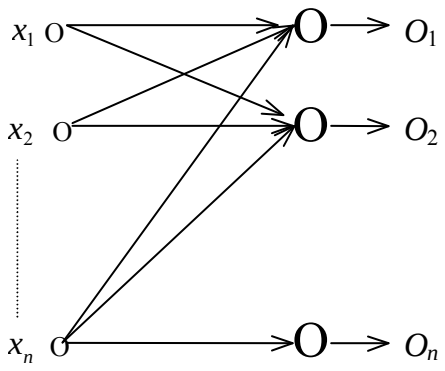

Figure 1.4 The Perceptron.

The input layer feeds the data through to the output layer where the neurons process the inputs and weights using the threshold activation function as described by eq. 1.1. The two layers are fully connected in that each input node is connected to each output node. The perceptron carries out pattern classification by learning the weights between the layers by **supervised learning** where the network is supplied with known input and output data. The differences between supervised and unsupervised learning will be discussed in Section 4.

A typical problem that was tackled in the early days of the perceptron is where the input and output data is binary (i.e. 0 or 1) and we have a set of inputs and outputs. This is an example of a training data set and each pair of inputs and outputs is known as a pair of training patterns. The perceptron learns the weights in the following way.

1. The weights are randomly initialised to small values.
2. For the first input pattern the network output is calculated using eq. 1.1. Denote this output by O_{jp} for input pattern p and output unit j.
3. The error is calculated as $T_{jp} - O_{jp}$ where T_{jp} represents the known output.
4. A simple learning rule for the weights might be

$$w_{ji}^{new} = w_{ji} + c(T_{jp} - O_{jp})a_i$$

where

c is a constant ($0 \leq c \leq 1$) representing the learning rate

a_i is the value of input unit i

w_{ji} is the weight from node j to node i

w_{ji}^{new} is the new weight from node j to node i.

5. The process is repeated for the next input pattern.

This process is repeated until there is an acceptable error in the classification, where the error is usually a root-mean-square measure such as

$$\frac{\sqrt{\sum_{p=1}^{n_p}\sum_{j=1}^{n_o}(T_{jp}-O_{jp})^2}}{n_p n_o}$$

where n_p is the number of patterns in the training set and n_o is the number of output units.

The exciting fact about this approach is that it was proved that this algorithm could learn anything it could represent. In other words the weights could be adjusted to simulate any function that could be represented by inputs and outputs. However there was a problem. In 1969 Professor Minsky found a whole class of problems that a perceptron could not simulate – problems that are not linearly separable. The best example of this is the exclusive-OR function. This is where we have two inputs, x and y, and an output, out; the relationships are given in Table 1.1 and the graphical representation is in Figure 2.5.

Table 1.1 The Exclusive-OR Problem

x	y	Out
0	0	0
1	0	1
0	1	1
1	1	0

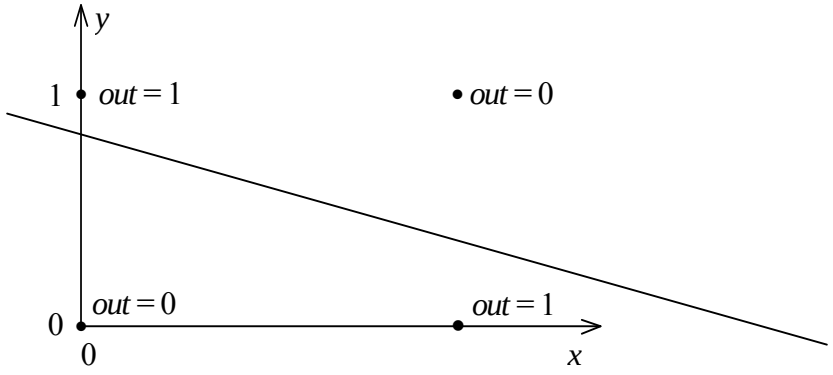

Figure 1.5 Graphical Representation of the Exclusive-OR Problem.

For a single neuron with threshold activation function with two inputs and one output it can be shown that there are no two weights that will produce a solution. It is not linearly separable in that there is no straight line which will 'split' the points with output values of 1 from those with output values of 0. Unfortunately, it seems that most real world problems are not linearly separable. This caused the development of neural networks to be put on hold until Rumelhart and McClelland (1986) put forward the notion of the **multilayered perceptron** with three layers of neurons. These networks form the basis for much of the success of neural networks today. In particular, multilayered neural networks that use the **backpropagation** algorithm (see Section 2.3) are employed in most successful ANN applications.

1.4 THE HODGKIN-HUXLEY MODEL

It is evident that Hodgkin and Huxley's theory and mathematical model for the generation of the action potential, published in complete form in 1952, was widely admired and received careful consideration, as shown in several references. Some of the historical and elementary theoretical background will be reviewed here. The intent is to present a simple, clear picture of the action potential and its associated refractory periods, for use later in studying the dynamics of small reverberatory networks. Perhaps this point of view will also be helpful to others, such as retired engineers, who would like to contribute to neuroscience as amateurs.

Hodgkin and Huxley's work was an experimental and theoretical triumph, but it developed out of a gradually clarifying view of neural processes, to which many people contributed. Some of these are mentioned by Hille (1984, pp. 24-28).

1.4.1 RESTING POTENTIAL

Bernstein used Walter Nernst's equation, which is based on thermodynamic principles, to calculate the resting potential inside the membrane from the concentration gradient for potassium. The qualitative explanation of this effect is that the membrane is semipermeable to potassium (but supposedly not to sodium) so that potassium tends to leak out, and in doing so the potassium ions leave a negative charge in the cytoplasm inside the neuron. This sets up a potential difference between inside and outside, which tends to drive potassium ions back in. An equilibrium is established between this and the concentration gradient driving them out. The Nernst equation gives the potassium equilibrium potential as

$$E_K = (RT/zF) \ln ([K^+]_o/[K^+]_i)$$

where R and F are physical constants, z is valence, and T is the absolute temperature (degrees K). $[K^+]_o$ is the outer concentration of K^+, and $[K^+]_i$ is the inner concentration.

1.4.2 THE EQUIVALENT CIRCUIT

To facilitate complete description of the factors determining resting potential in neurons, an electrical equivalent circuit can be used. This circuit represents a short cylindrical section of an axon, over which the membrane potential is approximately uniform. The equivalent circuit is shown in Fig. 1.6. The potential at one node, labeled V_m, corresponds to the uniform potential. The membrane has the distributed properties of conductance (the inverse of resistance) and capacitance, both of which are represented by discrete elements in the circuit. Biologically, there is a mechanism called the Na^+-K^+ pump, which metabolically establishes concentration gradients for sodium and potassium ions across the membrane. This is not represented by a circuit element; it simply maintains approximately fixed concentrations for sodium and potassium ions.

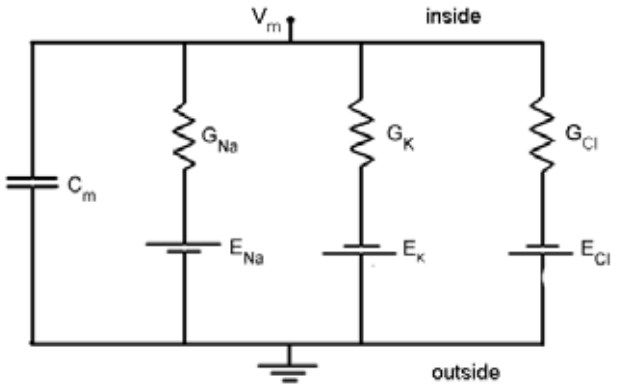

Figure 1.6 The conductances G_{Na}, G_K, and G_{Cl} (Cl for chloride) are small, so that the axon membrane is approximately an insulator separating two conductors: the cytoplasm and the external fluid. The cytoplasm is represented by the node with the membrane potential V_m. The external fluid corresponds to the lower node connected to the ground symbol. The ground indicates that the potential of the external fluid is being taken as the reference potential, and is therefore zero.

1.4.3 THE RESTING CONDUCTANCE

The conductances G_{Na}, G_K, and G_{Cl} are total values representing a great many individual ion channels. They are constants during the resting potential. Later, in modeling the action potential, G_{Na} and G_K will be taken as variables

dependent on V_m and time. The individual ion channels are diverse as well as numerous. Studies of the genes that encode ion channels indicate that a great many combinations of properties are possible, so that channels can serve a variety of special purposes. For the study of membrane potential in axons, ion channels can be divided into resting channels, which are always open, and voltage-gated channels which only open during the action potential. Thus, in the discussion of the resting potential, the symbols G_{Na}, G_K, and G_{Cl} only represent the totals of the resting channels of the three ionic types in Fig. 1.7. The three can be added together to give the total resting membrane conductance G_M.

$$G_M = 1/3333.3 = 3.0000 \times 10^{-4} \text{ mho/cm}^2 = 0.30000 \text{ mmho/cm}^2$$

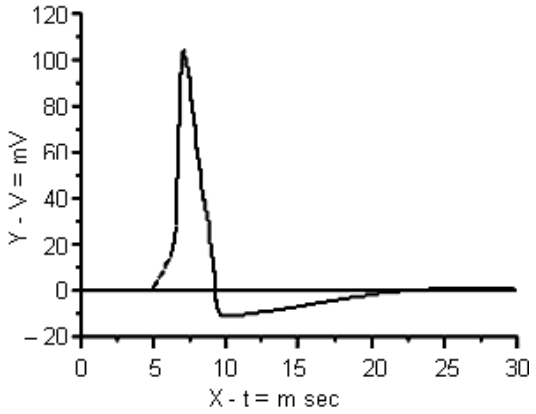

Figure 1.7 Resting Potential.

1.4.4 THE ACTION POTENTIAL

An action potential is shown in Fig. 1.8. This is taken from the Squid tutorial in Bower and Beeman (1998), which represents a section of axon with membrane properties similar to those of the squid giant axon. The figure is only part of a window in the Squid display. It can be seen that after the spike there is an after-potential which drops below 0 mV (This is the resting potential, taken as zero to conform with Hodgkin and Huxley; considered – 70 mV elsewhere) before settling down. This is the time interval in which there is a refractory period.

Hodgkin and Huxley's outstanding achievement was the quantitative explanation of the action potential. Their work was based on the sodium hypothesis presented by Hodgkin and Katz (1949). This proposed that, during activity after stimulation above the threshold, the membrane becomes selectively highly permeable to sodium ions, so that the inward flow of sodium through the membrane briefly eclipses the outward flow of potassium, and the internal membrane potential goes positive past zero (103 – 70 = +33 mV), and rises toward the sodium equilibrium potential. Then the sodium permeability

declines and the potassium permeability becomes large, until the spike is completed.

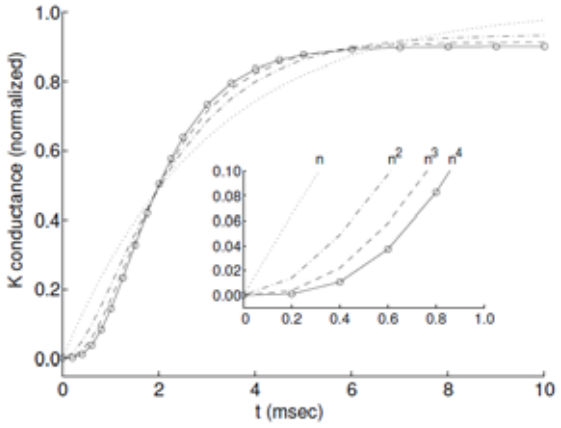

Fig. 1.8 The initial inflection in the curve cannot be well fit by a simple exponential (dotted line) that rises linearly from zero. Successively higher powers of p (p =2: dot-dashed; p = 3: dashed line) result in a better fit to the initial inflection. In this case, p = 4 (solid line) gives the best fit

For quantitative application of the sodium hypothesis to calculate the action potential, values for the sodium and potassium conductances as functions of voltage and time were needed. Hodgkin and Huxley obtained these by means of a voltage-clamp technique, which is described by Johnston and Wu (1995) on pages 143-145. Two silver-wire electrodes were inserted axially through one end of a length of axon. One electrode measured the membrane potential V_m, and a circuit compared it with a command voltage which had been set by the experimenter. If V_m deviated from the command voltage, a current was injected at the second electrode to eliminate the deviation. Thus V_m was clamped to the command value.

The experiments started from a holding voltage which was apparently intended to equal the resting potential. Hodgkin and Huxley (1952a), page 455 mention estimates of – 60 to – 65 mV for the actual resting potential, depending on factors such as junction potential, which modify the measured value (Their signs are reversed, being given as + 60 to + 65, conforming to the choice mentioned in Hodgkin, Huxley, and Katz (1952), page 425). A range of values for the command voltage was used. Following the account in Johnston and Wu, the starts were from a holding voltage of – 60 mV relative to the external fluid, followed by a voltage step to a command voltage, which was then maintained for 8 msec.

The injected current required to maintain the command voltage was recorded as a function of time. The injected currents used in the above procedure pass through the membrane to return to the external circuit. Thus they are equal to sums of the sodium and potassium currents flowing through the membrane. To separate the twos, the experiments can be repeated with one or the other type of channels blocked. The sodium channels will be blocked if the extracellular Na^+ concentration is reduced to its inside value, so that the Na^+ equilibrium potential is zero. They can also be blocked with tetrodotoxin. The potassium channels can be blocked by removing intracellular K^+, or with tetraethylammonium. The result is a family of curves for sodium current I_{Na} as a function of time with V_m as parameter, and a similar family for I_K.

Johnston and Wu (1995, pp. 145-147) describe how it was demonstrated that the instantaneous relation between current and voltage in the above experiments is linear; that is, that it follows Ohm's law, so that

$$I_{Na} = G_{Na} (V_m - E_{Na})$$

and

$$I_K = G_K (V_m - E_K).$$

Thus the conductances G_{Na} and G_K can be obtained as functions of V_m and time t by dividing the respective currents by their driving voltages.

The foregoing procedures give G_{Na} and G_K as functions of V_m and t over the range of discrete values of V_m and t which were tested. For use in simulation calculations of the action potential, the discrete data points must be fitted with continuous functions. Hodgkin and Huxley tried to relate the complex curve forms at particular values of V_m to exponentials, and in doing so were led to express them in terms of maximum values of G_{Na} and G_K which are multiplied by coefficients ranging between zero and 1. The coefficients turned out to be small integral powers of exponentials. The theoretical significance of these forms is discussed in Johnston and Wu (1995, pp. 149-154), and in Bower and Beeman (1998, pp. 37-38). The equations for approximating the voltage dependence are described in Bower and Beeman (1998, pp. 44-47).

Having constructed expressions for calculating the sodium and potassium conductances, Hodgkin and Huxley proceeded to use them to calculate the action potential. It was 1951, and the early Cambridge University computer was down for about 6 months for modification. Huxley managed to carry out numerical integrations of the differential equations with a hand-operated mechanical calculator. They were excited to see the action potential come out with the right shape. They saw that explanation of the action potential in the squid axon in terms of sodium and potassium conductances is fundamental, and that therefore it might be expected that similar explanations would apply to other excitable tissues, perhaps with different values for parameters.

1.4.5 COMPUTER SIMULATION OF NEURAL BEHAVIOR

The Hodgkin-Huxley theory provides a foundation for modeling neurons. It is especially suited to simulations on digital computers, since they handle the numerical integrations easily. The fundamental character of the theory gives assurance of a close correspondence to biological reality, and this gives the simulations value as natural experiments.

The GENESIS computer program for computational neuroscience (Bower and Beeman, 1998) is a flexible system for simulation, providing building blocks for simulations both below and above the level of the whole neuron. With GENESIS scripting, users can also design their own extended building blocks. The GENESIS tutorial named Squid, which is flexible in itself, embodies the classical Hodgkin-Huxley model of the giant axon. The Squid tutorial is of special interest here, because the chapter in Bower and Beeman (1998) which covers it includes Exercise 7 (p. 49) for investigating the refractory periods.

1.5 WHY SPIKING NEURONS?

Models of spiking neurons have been called the 3rd generation of artificial neural network (Maass 1997), as in

- Generation 1: Binary networks (activation of 0 or 1) such as implemented by McCulloch and Pitts' neurons and the Hopfield network.
- Generation 2: Real-valued networks, where activation is representative of the 'mean firing rate' of a neuron, such as Backpropagation networks and Kohonen self-organising maps.
- Generation 3: Spiking neural networks (SNNs).

Networks of the earlier generations have proven effective at modelling some cognitive processes and have been successful in many engineering applications. However the fidelity of these models with regards to neurophysiological data is minimal and this has several drawbacks.

- Neurophysiological knowledge cannot be integrated easily into the models and as such cannot be tested for applicability to or effect upon neural computation.
- Real neurons exhibit a very broad range of behaviours (tonic (continuous) and phasic (once-off) spiking, bursting, spike latency, spike frequency adaptation, resonance, threshold variability, input accommodation and bistability (Izhikevich 2004)). It's unlikely that these behaviours have no computational significance.

- There are specific interesting processes occurring at the spike level (such as Spike Timing Dependent Plasticity (Bi and Poo 1998)) that cannot be modelled without spikes.
- The dynamics of spiking networks are much richer, allowing for example
  - oscillations in network activity which could implement multiple concurrent processing streams (Izhikevich 1999), figure/ground segmentation and binding (Csibra, Davis et al. 2000; Engel, Fries et al. 2001), short term memory (Jensen, Idiart et al. 1996; Jensen, Gelfand et al. 2002) etc
  - much increased (perhaps by orders of magnitude) memory capacity (Izhikevich 2005). Transmission delays are very significant for computation particularly because they are random or Gaussian for real neurons - this causes the formation of polychronous (as against synchronous) spiking neuron groups which could possibly store many more population-encoded memories than there are synaptic weights. This idea is still to be fully researched and analysed.

One of the most exciting characteristics of spiking neural networks, with the potential to create a step-change in our knowledge of neural computation, is that they are innately embedded in time (Maass 2001). Spike latencies, axonal conduction delays, refractory periods, neuron resonance and network oscillations all give rise to an intrinsic ability to process time-varying data in a more natural and computationally powerful way than is available to 2nd generation models. Real brains are embedded in a time-varying environment; almost all real-world data and human or animal mental processing has a temporal dimension. Evidence is growing that rhythmic brain oscillations are strongly connected to cognitive processing (Klimesch 1999; Basar, Basar-Eroglu et al. 2001; Engel, Fries et al. 2001; Kahana, Seelig et al. 2001; Ward 2003). So utilising Spiking Neural Networks may be one of the first steps needed to bridge the current divide between existing ANN models and more flexible, realistic and, dare I say, intelligent, behaviour from artificial systems.

1.5.1 POTENTIAL DISADVANTAGES

- **Processing time**: Until recently, biologically realistic spiking models have required intensive computations for even small amounts of simulated time, making simulations of large networks or long time periods impractical in most situations. However this is no longer of such great concern with the formulation of a model which is both biologically realistic and computationally efficient (see the next section: What spiking neuron models are available?)
- **Complex dynamics**: One of the advantages may also be a disadvantage in that the complex behaviour needs to be understood and effectively managed. Although there are analogs to basins of attraction for networks

of spiking neurons on a global scale as long as noise is present in the system (Gerstner 1995), there is no such thing as a 'stable state' on a local neuron scale unless the entire network is completely inactive, so the network is always dynamically changing at some level.

- **Limited knowledge**: Much less is known about networks of spiking neurons than the more established ANN paradigms, and many well-accepted methodologies need to be adapted or possibly replaced. For instance with the right choice of parameters, STDP appears to be able to intrinsically support Hebbian learning with no need for arbitrary weight bounds or explicit weight or firing rate normalisation (van Rossum, Bi et al. 2000).

1.5.2 WHAT SPIKING NEURON MODELS ARE AVAILABLE?

Models fall into 3 broad categories ranging from complex to simple.

1. Systems of coupled equations of 2 or more variables using parameters with real biophysical correlates.
2. Systems of coupled equations of 2 or more variables using parameters with **no** biophysical correlates.
3. Integrate and Fire models.

These categories are not entirely mutually exclusive; some overlap is evident. Discussion of each follows.

1.5.3 COUPLED EQUATIONS USING PARAMETERS WITH REAL BIOPHYSICAL CORRELATES

Although it was published more than 50 years ago, the definitive model here remains the Hodgkin-Huxley model (Hodgkin and Huxley 1952). Its advantage is that it completely describes neuron behaviour in terms of all known biophysical parameters and as such is theoretically able to model any possible neuron behaviour. In practice, the parameter values can be difficult to determine, and in fact the values required to successfully emulate several known neuron types using this model are yet to be found (Izhikevich 2004). The model also has excessive computational requirements and hence cannot be used to simulate large networks over long time scales.

There are several simplified HH models available (Morris and Lecar 1981; Kistler, Gerstner et al. 1997; Wilson 1999); however any simplification always comes with consequent reduction in biological fidelity and/or simulated neuron behavioural repertoire.

1.5.4 COUPLED EQUATIONS USING PARAMETERS WITH NO

BIOPHYSICAL CORRELATES

If we assume that there is only one property of a neuron that needs to be modelled in order to capture everything about how one neuron influences others, and how neurons compute in general, then that property would be the membrane potential. In this case, the system of coupled equations can be greatly simplified while still exhibiting the rich set of spiking behaviors and sub-threshold characteristics of real neurons. The most successful model for this is the Izhikevich simple spiking neuron model (Izhikevich 2003), where success is measured in terms of both modelling efficiency and spiking behaviour. In fact, Izhikevich states that this is the simplest model possible which still exhibits all the required behaviours (Figure 1.9). It consists of just 2 equations and only 1 nonlinear term, so is computationally efficient – almost 100 times faster than the HH model to simulate. Studies have shown that the correspondence between a similar model of two variables (fitted to the data) and the HH models' spike timing predictions is very close – about 96% within 2 ms of each other (Brette and Gerstner 2005). The subthreshold dynamics of Izhikevich model neurons matches very closely those of HH (Boardman 2005).

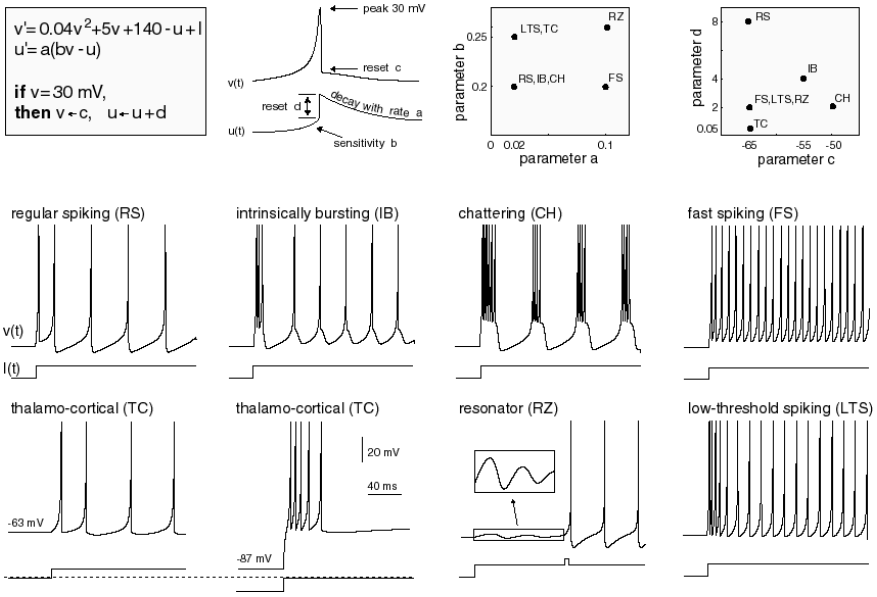

Figure 1.9 Top row – Izhikevich spiking neuron model. Bottom two rows – 8 of the 20 or so known neuron behaviours that can be emulated by the model.

The assumption that membrane potential is the only quantity needed for simulation of neural computation is arguable. A likely counter example is incorporation of synaptic plasticity into the model, which depends on concentrations of certain neurotransmitters and ions around the synapse (Urakubo and Watanabe 2002). However this doesn't rule

out the possibility that these values could be reliably approximated by some simple function of the membrane potential or spike timings, keeping the model equally simple while maintaining functional fidelity. This is the approach taken in (Izhikevich 2005).

A range of other models exist that lie somewhere between Izhikevich and Hodgkin-Huxley, e.g., (FitzHugh 1961; Rose and Hindmarsh 1989), often by trying to simplify the HH computations while modelling more known physical neuron properties than just membrane potential. In general, it can be said that the more properties modelled, the more computation is required; it is therefore an implementation-dependent trade-off as to which model should be chosen. However, with its ability to exhibit all known biological neuron behaviours whilst remaining computationally economical, the Izhikevich model is the current standout choice.

1.6 INTEGRATE AND FIRE MODELS

The simplest integrate and fire (I&F) model – the leaky I&F neuron – was originally developed when dominant thinking stated that neuron function can be well-enough approximated by simply integrating input and then firing at a given threshold. Other properties like spike frequency adaptation, bursting, resonance, latency and variable thresholds were incorporated into models as needs arose and dominant thinking changed (see (Izhikevich 2004) for a review); however unfortunately no single I&F model displays all these characteristics.

The underlying assumption of all I&F models is that of neurons being integrators, which we now know is not always the case (e.g., resonance, bistability, inhibition-induced spiking). However they are computationally efficient, even more so than the Izhikevich model, and should be considered if the full range of spiking behaviour is not required.

1.6.1 SPIKING NEURAL NETWORKS – WHAT DO WE KNOW?

1.6.1.1 Delay coding and universal approximation

All Generation 2 neural networks (ie with continuous real-valued activation functions) can be emulated by SNNs, using delay coding (Maass 1997), also called latency or, somewhat confusingly, temporal or firing order coding. The process of delay coding is

- Determine the minimum and maximum activation levels of the Gen 2 neurons.
- Determine a suitable ‘time-slice’ length for the Gen 3 network that will equate to an iteration (one step) of the Gen 2 network. The Gen 3 network runs in continuous time made up of consecutive time-slices.

- Within each time-slice, fire each neuron at a time proportional to its Gen 2 activation level – e.g., neurons with maximum activation fire at the start of the time-slice and neurons with minimum fire at the end; neurons with ¼ activation level fire ¾ of the way through the time-slice, and so on.
- Postsynaptic neurons decode the presynaptic activation level based on the firing delay within the time-slice.

Two obvious consequences of the fact that all ANNs can be emulated by SNNs are that

- Generation 2 ANNs are a subset of SNNs.
- SNNs are universal approximators (because Gen 2 ANNs are).

Another consequence is that

- Since any continuous function can be approximated with arbitrarily high reliability by an SNN with a single hidden layer (as can an MLP trained with back propagation), then with biologically realistic choices of spiking neuron parameters, any continuous function can be computed by an SNN within 20 ms (Maass 2001).

1.6.2 SPIKE TIMING DEPENDENT PLASTICITY (STDP)

STDP is a generalisation and refinement of Hebbian learning, stating that an increase (potentiation) in synaptic efficacy occurs if a presynaptic neuron fires immediately prior to the postsynaptic, and a decrease (depression) occurs if postsynaptic firing immediately precedes presynaptic. The effect was observed by Bi and Poo in hippocampal neurons in 1998 (Figure 1.10). The time course and magnitude of the effect vary between experimental studies, but it seems to last 10-50 ms either side of the postsynaptic spike and is maximal at or near the time of the spike, decaying exponentially to 0 by the end of the time course (Bi and Poo 1998; Kepecs, van Rossum et al. 2002; Urakubo and Watanabe 2002). It operates on excitatory synapses only and is less effective at potentiating already-strong synapses, although when synaptic depression occurs, the existing synaptic strength has less bearing on the result.

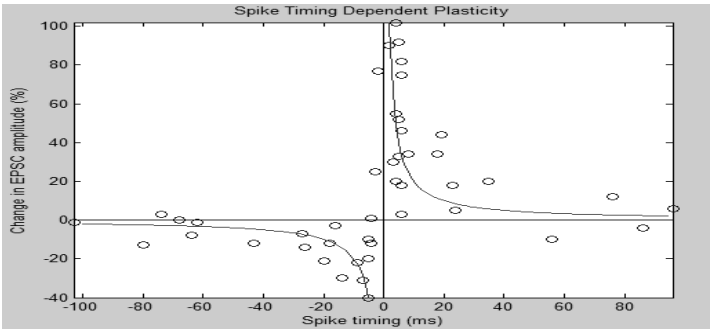

Figure 1.10 Change in Excitatory Post-Synaptic Current subsequent to different pre- and postsynaptic spike timings ($t_{post} - t_{pre}$). Exponential curves are shown for reference in red. Adapted from (Bi and Poo 1998).

Experimental data on the plasticity of inhibitory synapses on the other hand is scant (Swiercz, Cios et al. 2006), but often they are modelled more like Hebb's original rule whereby potentiation occurs if the two neurons fire closely together independent of the order of firing (Paugam-Moisy 2006). Others believe that inhibitory plasticity is computationally ill-advised and is not widespread in nervous systems (McBain, Freund et al. 1999).

STDP can be of itself lead to the stable development of network representations with no need for any form of weight or firing rate normalisation (Kempter, Gerstner et al. 1999; Song, Miller et al. 2000; van Rossum, Bi et al. 2000; Abbott and Gerstner 2004). However neurons are known to perform synaptic scaling in order to actively maintain a fixed long term average firing rate (Baddeley 1997); combined with standard Hebbian learning, this results in an implementation of Oja's rule, equivalent to principal components analysis or PCA (Oja 1982). STDP can be mathematically derived from several different starting constraints (Bohte and Mozer 2005). A multidisciplinary (molecular , biological and computational) and multiscale (of both temporal and spatial dimensions) review of STDP can be found in (Worgotter and Porr 2005).

Importantly, recent work has come up with the spiking neuron convergence conjecture (Legenstein, Naeger et al. 2005) which predicts that STDP can implement any input/output mapping that an SNN could ever potentially perform. They prove that the perceptron convergence theorem holds in the average case for STDP with Poisson input spike trains, then show through simulations that it holds in the test cases for more-realistic neurons and more-general input distributions. This endows STDP with universal learning capabilities.

1.6.3 NETWORK ARCHITECTURES

Neurons within nervous systems tend to have low average firing rates and be sparsely connected (Feldman and Ballard 1982; Brunel 2000; Reyes 2003), with synapses having a wide range of delays from one to several tens of milliseconds (Swadlow 1985). Rather than being a biological irrelevance, these appear to be crucial properties that affect how nervous systems perform computations. The concept of a temporal grouping of cells is central (Abeles 1991; Bienenstock 1995; Abeles 2002; Beggs and Plenz 2003; Ikegaya, Aaron et al. 2004), also see the very early paper (Rochester, Holland et al. 1956)! Temporal groups extend earlier ideas of grandmother cells, population coding, cell assemblies (Hebb 1949; Freeman 1991; Reilly 2001) and synfire chains (Abeles 1991) into the time domain since a group may not be a synchronously-firing collection of cells, but rather cells that fire in given order with known timings. Izhikevich calls this property 'polychrony' as against synchrony (Izhikevich 2005). Temporal groups come about due to sparse connectivity, spiking dynamics and particularly, random delays, and play a vital role in information processing by nervous systems, or in fact are the physical manifestation of the processing of information. So spiking neurons and their typical connectivity together support computation using SNNs, and it becomes arguably impossible, or at least unwise, to separate them.

The significance of the random delays for network function cannot be overstated. It's now well known that a sparsely connected network with random synaptic delays can easily have more states than the number of nodes (n) in the network (Izhikevich 2005) – remember a state in this terminology is not an attractor or stable state; rather it is a grouping of neurons that tend to fire in a given order over time. Contrast this to the Hopfield net (Hopfield 1982) where memories begin breaking down when just 0.13n for n = 100 (in general n/(4 log n) (McEliece, Posner et al. 1987)) are stored. In an SNN with well-chosen synaptic delays, there can actually be many more potential states than synapses in the entire network (Izhikevich 2005), which is an unprecedented memory capacity. Unfortunately, for systems of high-dimensional non-linear temporal interactions such as these, it is very difficult to conduct any rigorous mathematical analysis (Maass 2001), so no upper bound on memory capacity is known and no formal predictions of network behaviour can be made. Indeed, from a mathematical point of view, systems with temporal delays are infinite-dimensional. Despite the inability of off-the-shelf mathematics to provide analysis, it's clear that SNNs have unparalleled memory capacity.

Even randomly connected networks (or randomly connected satisfying certain constraints) with fixed (unitary) transmission delays exhibit potential to perform powerful computations, as long as there is a means to extract useful information from them. This is the basis of both Echo State Networks (Jaeger 2001) and Liquid State Machines (Maass, Natschlager et al. 2002), two closely

related architectures that use random recurrent networks as their computational engines. Put very simply, their strategy is this:

- Pass the input into a highly re-entrant random network, which converts the input into a high dimensional dynamical representation. The weights and delays in the network are fixed, not trained, but in general there will be many more neurons than inputs.
- Simply train the output layer with least mean squares, linear regression or even a simple delta-rule variant.

These networks have been used to solve some quite difficult benchmark non-linear learning problems. The effectiveness of such a simple strategy is testament to the underlying computational power of recurrent networks, even random untrained ones. Put even more simply, the rationale behind their functioning is that, in any random network of useful yet still tractable size, the dynamics are rich enough that at least one neuron will have acquired close to any desired representation.

1.6.4 SPIKE CODING

The classic notion that spiking neurons encode information through their average spike rate over some time window (called a rate code) is obviously correct in some circumstances and obviously incorrect in others (Rieke, Steveninck et al. 1997). Sensory cells such as in the cochlear and the retina use a rate code (Izhikevich 2005), however response time in the visual cortex is known to be too fast to continue processing with this coding regime (Thorpe, Fize et al. 1996; Thorpe, Delorme et al. 2001) – each neuron in the visual processing hierarchy only has time to fire one or occasionally two spikes prior to recognition, so clearly the visual cortex cannot be using a rate code, but is instead somehow utilising the presence and/or the timing of spikes (called a temporal code, not to be confused with delay coding as described above, which is one form of temporal coding). Spike timing has been long-established as the information encoding principle used in the auditory system of bats for echolocation (Kuwabara and Suga 1993) and in the visual system of flies (Bialek, Rieke et al. 1991), for example.

There are a number of different coding strategies possible using spike times, shown below in increasing order of information encoding capacity (Thorpe, Delorme et al. 2001).

- **Count coding:** counts the total number of spikes of a neuron population in a given time – similar to rate coding except it entails one spike from each of many neurons instead of many spikes from one neuron; however the information capacity of rate coding is the same (very small).

- **Binary coding:** encodes a binary number, each digit represented by the presence (1) or absence (0) of a spike.
- **Rank order coding:** the order of firing of the neurons encodes the information.
- **Delay coding:** as described earlier.

Each of these coding strategies has very different information capacities summarized in the Table 1.2. In the table, n is the number of neurons under consideration, T is the time window over which each neuron can fire either 0 or 1 spikes, and p is the precision, which is the minimum interspike interval which can be discerned by postsynaptic neurons. In the row of the table labeled Example 1 it is assumed that n = 10 neurons, T = 10 ms and p = 1 ms. In Example 2, n = 1000 neurons while T and p remain unchanged, although this would overstate the information capacity of rank order coding since there are not n! discernable outcomes using this strategy in this case, as with 1 ms precision many spikes will appear to be simultaneous. In this case the number of discernable outcomes for rank order coding will actually be $\log_2({}^{1000}C_{100} \times {}^{900}C_{100} \times {}^{800}C_{100} \times \ldots \times {}^{100}C_{100}) = 3280$ bits (notice for Example 1 this degenerates to $\log_2({}^{10}C_1 \times {}^{9}C_1 \times {}^{8}C_1 \times \ldots \times {}^{1}C_1) = \log_2(10!)$). The information capacity of rank order coding is approaching that of delay coding (= 3320 bits) here because, with 1ms precision and 1000 spikes to squeeze into 10 ms, delays don't add a lot of extra information. Note that rank order coding also becomes indistinguishable from delay coding as the precision p approaches the mean interspike interval.

Table 1.2 Various descriptions and performance parameters

	Count coding	**Binary coding**	**Rank order coding**	**Delay coding**
Description	Counts the total number of spikes	A binary number	The order the neurons fire	The order and delays both encode information
Information capacity (bits)	$\log_2(n+1)$	n	$\log_2(n!)$	$n.\log_2(T/p)$
Example 1	3.6	10.0	21.8	33.2
Example 2	10.0	1000	?[3] (should be 3280)	3320

[3] The stated information capacity for rank order coding is not accurate when n > T/p since there cannot be n! discernible states in T milliseconds. See text for a more correct statement of information capacity in this case.

So which of the above temporal code strategies does the brain use when not using rate coding? For fast visual processing, rank order coding appears to suffice. For implementing STDP, delay coding would seem to be required in order to reliably control synaptic efficacy based on spike timings; however in nervous systems the distinction between order and delay coding may be blurred, since a delay of, for example, several hundred milliseconds between spikes from two neurons, does not change the order, yet is not likely to be interpreted as part of a single rank order code instance (i.e., even in order coding there are practical bounds on the delays).

Transmission delays in nervous systems display high variability even within connections that run in parallel, i.e., two connections that originate in one brain area, terminate in another brain area and follow very similar paths can have very different transmission delays. Conversely, some connection types such as thalamo-cortical can have very similar transmission delays irrespective of their length (Salami, Itami et al. 2003). It seems that nervous systems are able to exert purposeful control over these delays in situations where they may have computational significance. Some experimental data is shown in the Table 1.3 (Izhikevich 2005).

Table 1.3 Experimental Data based on Delay

Connection	Delay (ms)	Reference
Cat Layer 6 – LGN	1.0 – 44	(Ferster and Lindstrom 1983)
Rabbit Layer 6 – LGN	1.7 – 32	(Swadlow 1994)
Rabbit Cortico-cortical	1.0 – 35	(Swadlow 1985)
Rabbit Cortico-cortical	1.2 – 19	(Swadlow 1994)
Rabbit Cortico-(ipsi)cortical	2.2 – 32.5	(Swadlow 1994)
Rabbit Layer 5 – LGN	0.6 – 2.3	(Swadlow 1994)
Cat Cortico-collicular	0 – 3	(Ferster and Lindstrom 1983)
Mouse VB – Layer 4	2	(Salami, Itami et al. 2003)

Despite the high variability between connections in many brain areas, the delay in any given connection is consistently reproducible with sub-millisecond precision. The overarching theme here is that variable transmission delays are a fundamental property and a computational requirement of nervous systems, and hence should not be overlooked in SNN models.

1.6.5 COMPLEXITY

Interesting results have been obtained in analyses of the VC-dimension of spiking neurons (Maass and Schmitt 1997):

- The VC dimension of a threshold gate with n variable weights is $\Omega(n)$.

- The VC dimension of a spiking neuron with n variable delays is $\Omega(n.\log(n))$, even with fixed weights.

So the discriminatory power of variable delays is greater than that of variable weights. This implies that

- Networks with synaptic delays are potentially able to perform more powerful computations (Maass 1997; Maass 1997).
- Powerful learning algorithms could be formulated by adjusting synaptic delays in addition to or rather than synaptic weights.

Non-learnability results have been derived for SNNs that put the learning complexity of spiking neurons into the NP class of problems (Maass and Schmitt 1997). However due to the limitations of the mathematical tools that are used to conduct these rigorous analyses, simplifying assumptions must be made, and while these results mean it will be difficult to formally prove that learning algorithms work, it doesn't necessarily mean that they will be difficult to formulate.

1.6.6 SPIKING NEURAL NETWORKS – WHAT DON'T WE KNOW?

Despite the quite large body of knowledge expounded above, we still know very little about the dynamics, learning algorithms and computational abilities of SNNs. Obvious gaping holes in our knowledge include:

- **Broad mathematical analyses**: SNNs are high dimensional nonlinear dynamical systems. Consequently we generally cannot prove convergence for learning algorithms, and have little knowledge of upper bounds on memory capacities or of the dynamical behaviour of SNNs in differing circumstances. Given the richness of potential neuron behaviours, Izhikevich states in (Izhikevich 2004) "What happens when only tens (let alone billions) of such neurons are coupled together is beyond our comprehension". The (inadequate) alternative is to demonstrate these properties through exhaustive numerical simulation; however this proves nothing and it is difficult or impossible to generalise from these results.
- **Training of synaptic delays**: Given the computational power afforded by transmission delays in SNNs, there has been surprisingly little research published on mechanisms of training them. This may be partly due to the fact that there is dissent on whether or not the brain actually modifies delays as a facilitator of computational function (see (Eurich, Pawelzik et al. 1999) vs (Senn, Schneider et al. 2002) for example). However it has been shown that simple Hebbian-like learning rules can progressively modify synaptic delays so that spikes that arrive at different times within a given time window can ultimately be synchronised at the postsynaptic

neuron, and produce stable representations of temporal input (Huning, Glunder et al. 1998; Eurich, Pawelzik et al. 1999). These rules function by adding to a delay if the presynaptic neuron fires some time before the postsynaptic, and subtracting from a delay if the presynaptic neuron fires some time later. Biological mechanisms for controlling the delay can include changing the thicknesses of axons and dendrites or the extent of myelination (Fields 2005). An alternative to actively adapting transmission delays is to simply select appropriate delays from an initial over-abundance of random delays; delays that by chance closely match the input train are strengthened by normal Hebbian learning, while the remaining ineffective delays are depressed by the same mechanism and may ultimately vanish. It is well known that the juvenile brain contains many more synapses than the adult, but whether selection of delays is one reason for this reduction over time, and whether delay training is also undertaken by the brain, are still open questions.

- **How does the brain compute?** Transmission delays, spike coding, temporal grouping, nested oscillations, phase precession – irrespective of what we do or don't know about spiking neurons, some of the most fundamental attributes of the brain are still almost complete mysteries to us. Although SNNs are the next step towards a full understanding of nervous system computation and can help us to model many of these till-now neglected properties, they are not likely to be the ultimate level of detail required to model all of brain function, and we may yet need Generation 4 and beyond neural networks for this task.

1.6.7 WHO IS USING SPIKING NEURONS?

Work with spiking neurons and SNNs has been going on at the periphery of neural network research for many years. This includes:

- A considerable amount of theoretical work by mathematicians and physicists, some of which has been discussed above; see (Rieke, Steveninck et al. 1997) for more information.
- Obviously all neuroscientists, who are most interested in the physical mechanisms of spiking, STDP etc, hence use spiking neuron models with high biological realism.
- Brain region modelling – deep but not broad models of targeted brain regions operating in specific modalities e.g., hippocampus in a specific navigation task (Hasselmo, Bodelon et. al., 2002).
- Cognitive modelling – understandably, there is a disconnect between higher level brain models and models of individual spikes (solving this completely is arguably solving AI!)

- Some applications – currently dominated by auditory (mostly speech) processing (Hopfield and Brody 2000; Hopfield and Brody 2001; Loiselle, Rouat et al. 2005; Verstraeten, Schrauwen et al. 2005) and visual processing (Perrinet and Samuelides 2002; Azhar, Iftekharuddin et al. 2005; Kornprobst, Vieille et al. 2005); SpikeNet Technology is a commercialised vision package (Thorpe and Gautrais 1997; Thorpe, Guyonneau et al. 2004), also see http://www.spikenet-technology.com. Robotics with SNNs is just beginning to heat up (Di Paolo 2002; Nielsen and Lund 2003; Roggen, Hofmann et al. 2003; Floreano, Epars et al. 2005; Floreano, Zufferey et al. 2005), although most current robotics implementations depend on evolutionary algorithms to create the SNNs.

1.6.8 CONCLUSION

With the knowledge we are currently obtaining of the fundamental importance of spike timings and oscillations to neural processing, 2nd generation ANNs can no longer provide a viable basis for neural modelling. Spiking Neural Networks present many new challenges but also afford many new opportunities for breaking entirely new ground in artificial intelligence research.

1.7 APPLICATIONS OF ANN

Let us suppose we wish to train an ANN to recognize handwriting. For our purposes we wish to train it to recognize the letters H and C. More exactly for this simple application we would be happy for it to be able to identify an H or a C. Handwriting recognition is a problem that neural networks have been able to tackle very well. The issues that arise here are common to all neural network classification problems.

The first activity is to represent the letters for input to the ANN. Typically we draw a grid over the letter and represent the inputs as zero or one depending on whether the hand-written letter goes through the cell in the grid. Figure 1.11 gives an example for an H and a C.

The inputs will be strings of zeros and ones. The string is achieved by going from the top left-hand corner across to the right and then left to right a row at a time, where a 1 indicates that the letter goes through that square of the grid. In the example shown, the inputs will be 110111011 for H and 110100111 for C. We would only need one output for this particular problem. We could perhaps give it a value of 1 for an H and 0 for a C. So there are nine inputs and one

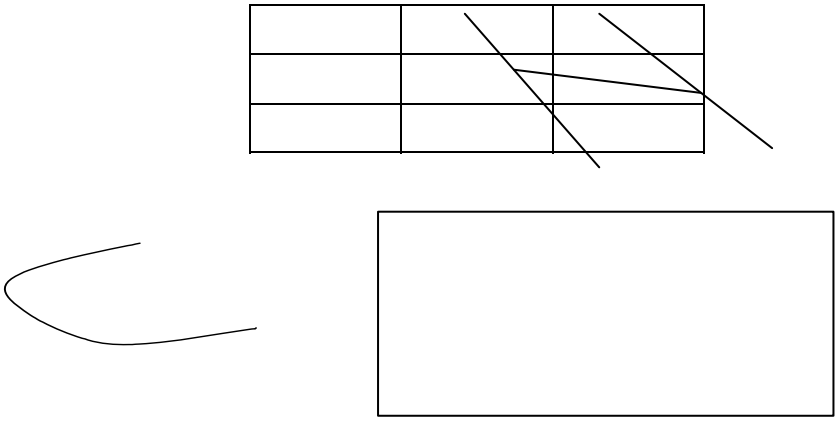

Figure 1.11 Letter Recognition – an H and a C

output. For most applications we only require one hidden layer. As to how many neurons there should be in the hidden layer this is still a case of 'trial and error'. Typically you experiment from, say, about 7 down to 2. Suppose we have 4 in the hidden layer. This is called a 9-4-1 network. This network has, therefore, 40 [i.e. $(9 \times 4) + (4 \times 1)$] weights that have to be learned. For a neural network to have good generalisation capabilities (to be able to classify inputs it has not seen before) the literature reports that you need approximately 5-10 times as many training pairs as weights. So in this example we would need about 200 letters with approximately 100 Hs and 100 Cs. As you can see, if you dropped to 2 neurons in the hidden layer you would need only 100 data sets. The questions that need answering then are:

1. How do we represent the data for input?
2. How many neurons should be in the output layer and, in the case of classification problems, what values will those neurons take?
3. How many neurons should be in the hidden layer?
4. What activation function should we use for the neuron?
5. What is an acceptable error rate?

This section has introduced you to the most common paradigm in neural network applications – the feed-forward, multilayered perceptron with the backpropagation algorithm. This approach is just one example of a supervised algorithm.

SUMMARY OF ANN

Artificial Neural Networks – What are They good for?

It would not be possible to list all the applications of ANNs here. An ANN approach will often be an option where the problem being tackled has the following features.

- The type of problem:
  - is one of recognising patterns in the data.
  - requires classification of data into, for example, classes.
  - is one of monitoring of equipment in real time.
- There is a large amount of data that may be 'noisy'.

Some example applications are:

- Monitoring of engine condition in a fleet of vehicles.
- Signal processing – recognising patterns in signals.
- Face recognition. Neural networks are particularly good at recognising shapes, e.g. fingerprints, signatures, tanks on an horizon.
- Process control - using ANNs to monitor equipment.
- Forecasting corporate bankruptcy based on financial indicators.
- Credit scoring to assess credit worthiness when considering giving a loan.
- We know how neural networks work and the types of applications for which they are suitable.

Artificial Neural Networks - What are the Drawbacks?

- They require large amounts of historical data that accurately reflects the make-up of the population under consideration.
- They are 'black box'. Unlike expert systems, they are incapable of explaining why they make a particular decision. This is a major problem when trying to 'sell' neural network technology to management. The only way to test the efficacy of an ANN solution is to test the trained network with many examples that it has not seen before.
- There are a large number of parameters that the ANN developer has to make decisions about. For example he/she has to decide on the learning rate, activation functions, the network structure or topology, how to represent the problem, etc.

In summary, ANNs are a powerful, practical solution to many problems faced by industry and commerce and should be considered as one of the tools in the armoury of the professional trying to find solutions to difficult problems.

REFERENCES

1. Aarts, E.H., F.M.J. de Bont, E. H. A. Haberrs, P.J. M. Laarhoven. (1986), "A Parallel Statistical Cooling Algorithm", *Lecture Note in Computer Science* 210, pp 87

2. Aarts E., anf J. Korst, (1989), *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing,* New York: Wiley

3. Abe S. (1989) "Theories of the Hopfield Neural Networks", *Proc. International Joint Conference on Neural Networks* (IJCNN'89), Washington D.C., Vol. I, pp 557- 564, June

4. Abe S. (1991) "Determining Weights of the Hopfield Neural Networks", *Proc.International Conference on Artificial Neural Networks* (ICANN'91) Helsinki, pp 1507-1510, June

5. Abe S. "Global Convergence and Supression of Spurious States of the Hopfield Neural Networks ", *Trans. IEEE Circuits & Systems*,

6. Abraham, R.H., and C.D. Shaw, (1992), *Dynamics of the Geometry of Behavior*, Reading, MA, Addison-Wesley.

7. Aiyer, S.V.B., M. Niranjan, F. Fallside, (1990), "On the Optimization Properties of the Hopfield Model", *Proc. International Conference on Neural Networks* (ICNN'90), pp 245-249

8. Aiyer S.V.B., N. Niranjan and F. Fallside, (1990), "A Theoretical Investigation into the Performance of the Hopfield Model.", *IEEE Transactions on Neural Networks* 15, 15, 204-215

9. Akiyema et al, (1991) "The Gaussian Machine: A stochastic Neural Network for Solving Assignment Problems", *Journal of Neural Network Computing,* Winter, pp 43-51

10. Allwright, J.R.A. and D.B. Carpenter, (1989), "A Distributed Implementation of Simulated Annealing for the Traveling Salesman Problem", *Parallel Computing* 10, pp 335, North Holland