POLYTECHNIC UNIVERSITY
Department of Computer and Information Science

Introduction to Artificial Neural Networks

K. Ming Leung

Abstract: A computing paradigm known as artificial neural network is introduced. The differences with the conventional von Neumann machines are discussed.

Directory

- **Table of Contents**
- **Begin Article**

Table of Contents

1. von Neumann Machine and the Symbolic Paradigm

The conventional electronic computer is built according to the von Neumann machine and the symbolic paradigm.

The basic operations may be modelled by having the computer repeatedly perform the following cycle of events:

1. The CPU fetches from memory an instruction
2. as well as any data required by the instruction.
3. The CPU then executes the instruction (process the data),
4. and stores the results in memory.
5. Back to step 1.

In this paradigm, it is possible to formalize many problems in terms of an algorithm, that is as a well defined procedure or recipe which, when executed in sequence, will produce the answer. Examples of such type of problems are the solution to a set of equations or the way to search for an item in a database. The required algorithm may be broken down into a set of simpler statements which can, in turn, be reduced eventually, to instructions that the CPU executes.

In particular, it is possible to process strings of symbols which represent ‘ideas’ or ‘concepts’ and obey the rules of some formal system. It is the hope in Artificial Intelligence (AI) that all knowledge could be formalized in such a way: that is, it could be reduced to the manipulation of symbols according to rules and this manipulation implemented on a conventional von Neumann machine.

We may draw up a list of the essential characteristics of such machines for comparison with those of neural networks to be discussed later.

1. The machine must be told in advance, and in great detail, the exact series of steps required to perform the algorithm. This series of steps is specified by the computer program.
2. The type of data it deals with has to be in a precise format - noisy data confuses the machine.
3. The machine can easily fail - destroy a few key memory locations for the data or instruction and the machine will stop functioning or ‘crash’.
4. There is a clear correspondence between the semantic objects

being dealt with (numbers, words, database entries etc) and the machine hardware. Each object can be 'pointed to' in a block of computer memory.

The success of the symbolic approach in AI depends directly on the consequences of the first point above which assumes we can find an algorithm to describe the solution to the problem. It turns out that many everyday tasks we take for granted are difficult to formalize in this way. For example, our visual (or auditory) recognition of things in the world; how do we recognize handwritten characters, the particular instances of which, we may never have seen before, or someone's face from an angle we have never encountered? How do we recall whole visual scenes on given some obscure verbal cue? The techniques used in conventional databases are too impoverished to account for the wide diversity of associations we can make.

The way out of these difficulties that will be explored in this course is to use artificial neural network (ANN) to mimic in some way the physical architecture of the brain and to emulate brain functions.

2. The Brain

Our brain has a large number ($\approx 10^{11}$) of highly connected elements ($\approx 10^4$ connections per element) called neurons. A neuron has three major components: a cell body called soma, one axon, and numerous dendrites. Dendrites are tree-like receptive networks of nerve fibers that carry electrical signals into the cell body. The cell body has a nucleus, and sums and threshold all incoming signals. The axon is a long fiber that carries signal from the cell body out to other neurons.

Signals in the brain are transmitted between neurons by electrical pulses (action-potentials or 'spike' trains) traveling along the axon. The point of contact between an axon of one cell and a dendrite of another cell is called a synapse. Each pulse arriving at a synapse initiates the release of a small amount of chemical substance or neurotransmitter which travels across the synaptic cleft and which is then received at post-synaptic receptor sites on the dendritic side of the synapse. The neurotransmitter becomes bound to molecular sites here which, in turn, initiates a change in the dendritic membrane potential. This post-synaptic-potential (PSP) change may serve to increase or de-

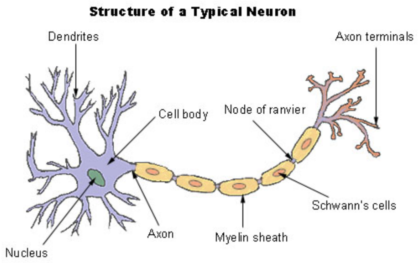

Figure 1: *A schematic diagram of a single neuron.*

crease the polarization of the post-synaptic membrane. In the former case, the PSP tends to inhibit generation of pulses in the afferent neuron, while in the latter, it tends to excite the generation of pulses. The size and type of PSP produced will depend on factors such as the geometry of the synapse and the type of neurotransmitter. All these factors serve to modify the effect of the incoming signal by imposing a certain weight factor. Each PSP will then travel along its dendrite and spread over the cell body of the neuron that is receiving the signal, eventually reaching the base of the axon (axon-hillock). The cell body sums or integrates the effects of thousands of such PSPs over its dendritic tree and over time. If the integrated potential at the axon-hillock exceeds a threshold, the cell 'fires' and generates an action potential or spike which starts to travel along its axon. This then initiates the whole sequence of events again in neurons contained in its pathway.

The way how the neurons are connected together and the nature of the synapses together determine the function of the particular part of the brain. The brain learns new tasks by establishing new path ways and by modifying the strengths of the synapses.

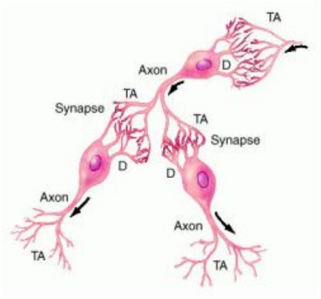

Figure 2: *A schematic diagram of a neuron connecting to two other neurons.*

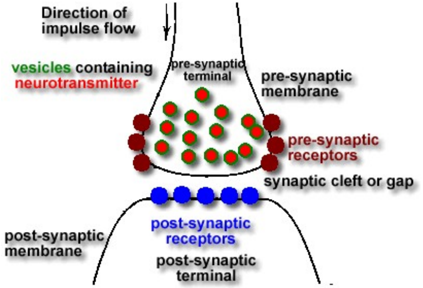

Figure 3: *A schematic diagram of a synapse.*

Reactions in biological neurons are rather slow ($\approx 10^{-3}s$) compared with electrical circuits inside the CPU ($\approx 10^{-9}s$), however the brain can perform certain tasks such as face recognition much faster than conventional computers mainly because of the massively parallel structure of the biological neural networks (in the sense that all neurons are operating at the same time).

3. Artificial Neural Networks

A Neural Network is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the inter-unit connection strengths, or weights, obtained by a process of adaptation to, or learning from, a set of training patterns. In some cases these weights have prescribed values at birth and need not be learned.

Artificial neural networks (ANNs) are not as complex as the brain, but imitate it in the following sense:

1. The building blocks of ANNs are simple computational devices

(capable of summing and thresholding incoming signals).

2. These devices are highly interconnected.
3. Information is processed locally at each neuron.
4. The strength of a synapse is modified by experience (learning).
5. The topological connections between the neurons as well as the connection strengths determine the function of the ANN.
6. Memory is distributed:
    - Long-term memory - resides in the synaptic strengths of neurons.
    - Short-term memory - corresponds to signal sent by neurons.
7. ANNs are inherently massively parallel.

The advantages of ANN computing are:

1. intrinsically massively parallel.
2. intrinsically fault-tolerant - many brain cells die each day and yet its function does not deteriorate much.

3. capable of learning and generalizing - one can often read some ones hand-writing even for the first time.
4. no need to know the underlying laws or governing equations.

Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks that are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency.

4. History

The history of neural networks that was described above can be divided into several periods:

1. First Attempts: There were some initial simulations using formal logic. McCulloch and Pitts (1943) developed models of neural networks based on their understanding of neurology. These

models made several assumptions about how neurons worked. Their networks were based on simple neurons which were considered to be binary devices with fixed thresholds. The results of their model were simple logic functions such as "a or b" and "a and b". Another attempt was by using computer simulations. Two groups (Farley and Clark, 1954; Rochester, Holland, Haibit and Duda, 1956). The first group (IBM researchers) maintained closed contact with neuroscientists at McGill University. So whenever their models did not work, they consulted the neuroscientists. This interaction established a multidisciplinary trend which continues to the present day.

2. Promising and Emerging Technology: Not only was neuroscience influential in the development of neural networks, but psychologists and engineers also contributed to the progress of neural network simulations. Rosenblatt (1958) stirred considerable interest and activity in the field when he designed and developed the Perceptron. The Perceptron had three layers with the middle layer known as the association layer. This system could

learn to connect or associate a given input to a random output unit. Another system was the ADALINE (ADAptive LINear Element) which was developed in 1960 by Widrow and Hoff (of Stanford University). The ADALINE was an analogue electronic device made from simple components. The method used for learning was different to that of the Perceptron, it employed the Least-Mean-Squares (LMS) learning rule.

3. Period of Frustration and Low Esteem: In 1969 Minsky and Papert wrote a book in which they generalized the limitations of single layer Perceptrons to multilayered systems. In the book they said: "...our intuitive judgment that the extension (to multilayer systems) is sterile". The significant result of their book was to eliminate funding for research with neural network simulations. The conclusions supported the disenchantment of researchers in the field. As a result, considerable prejudice against this field was activated.

4. Innovation: Although public interest and available funding were minimal, several researchers continued working to develop neu-

romorphically based computational methods for problems such as pattern recognition. During this period several paradigms were generated which modern work continues to enhance. Grossberg's (Steve Grossberg and Gail Carpenter in 1988) influence founded a school of thought which explores resonating algorithms. They developed the ART (Adaptive Resonance Theory) networks based on biologically plausible models. Anderson and Kohonen developed associative techniques independent of each other. Klopf (A. Henry Klopf) in 1972, developed a basis for learning in artificial neurons based on a biological principle for neuronal learning called heterostasis. Werbos (Paul Werbos 1974) developed and used the back-propagation learning method, however several years passed before this approach was popularized. Back-propagation nets are probably the most well known and widely applied of the neural networks today. In essence, the back-propagation net is a Perceptron with multiple layers, a different threshold function in the artificial neuron, and a more robust and capable learning rule. Amari (A. Shun-Ichi 1967) was involved with theoretical developments: he published a paper which established a mathematical theory for a learning basis (error-correction method) dealing with adaptive pattern classification. While Fukushima (F. Kunihiko) developed a step wise trained multilayered neural network for interpretation of handwritten characters. The original network was published in 1975 and was called the Cognitron.

5. Re-Emergence: Progress during the late 1970s and early 1980s was important to the re-emergence on interest in the neural network field. Several factors influenced this movement. For example, comprehensive books and conferences provided a forum for people in diverse fields with specialized technical languages, and the response to conferences and publications was quite positive. The news media picked up on the increased activity and tutorials helped disseminate the technology. Academic programs appeared and courses were introduced at most major Universities (in US and Europe). Attention is now focused on funding levels throughout Europe, Japan and the US and as this funding becomes available, several new commercial applications in

industry and financial institutions are emerging.

6. Today: Significant progress has been made in the field of neural networks enough to attract a great deal of attention and fund further research. Advancement beyond current commercial applications appears to be possible, and research is advancing the field on many fronts. Neurally based chips are emerging and applications to complex problems developing. Clearly, today is a period of transition for neural network technology.

5. Neural Network Computing

An artificial neural network (ANN) consists of a large number of highly connected artificial neurons. We will consider the different choices of neurons used in an ANN, the different types of connectivity (architecture) among the neurons, and the different schemes for modifying the weight factors connecting the neurons. From here on, it is clear that we are not actually dealing with the real biological neuron network, we will often drop the word ”artificial” and simply refer to what we do as neural network (NN).

5.1. Common Activation Functions for Neurons

Let us denote the weighted sum of input into a neuron by y_{in}. The activation or output of the neuron, y, is then given by applying the activation (or transfer) function, f, to y_{in}:

$$y = f(y_{\text{in}}).$$

The activation function should be a rather simple function. Some typical choices are given below.

- **Identity Function**

If f is an identity function,

$$y = f(y_{\text{in}}) = y_{\text{in}},$$

the activation (output) of the neuron is exactly the same as the weighted sum of the input into the neuron. As we will see later, identity activation functions are used for neurons in the input layer of an ANN.

- **Binary Step Function with Threshold**

$$y = f(y_{\text{in}}) = \begin{cases} 1 & \text{if } y_{\text{in}} \geq \theta, \\ 0 & \text{if } y_{\text{in}} < \theta. \end{cases}$$

The threshold value is specified by θ. The output has binary values (0 or 1) only.

- **Bipolar Step Function with Threshold**

$$y = f(y_{\text{in}}) = \begin{cases} 1 & \text{if } y_{\text{in}} \geq \theta, \\ -1 & \text{if } y_{\text{in}} < \theta. \end{cases}$$

The threshold value is specified by θ. The output has bipolar values (1 or -1) only.

- **Binary Sigmoid Function**

$$y = f(y_{\text{in}}) = \frac{1}{1 + e^{-\sigma(y_{\text{in}} - \theta)}},$$

where σ is a positive parameter. This function switches from 0 to 1 in the vicinity of θ as the argument goes from $-\infty$ to $+\infty$. The larger the values of σ is the more abrupt the change occurs. Unlike the above step functions, the binary Sigmoid function is smooth. In fact it has a derivative which is also smooth. It is easy to see that its derivative is given by

$$f'(y_{\text{in}}) = \sigma f(y_{\text{in}})[1 - f(y_{\text{in}})].$$

Thus its derivative can be easily computed from the value of the original function itself.

- **Bipolar Sigmoid Function**

$$y = f(y_{\text{in}}) = \tanh\left(\frac{\sigma}{2}(y_{\text{in}} - \theta)\right),$$

where σ is a positive parameter. This function switches from -1 to 1 in the vicinity of θ as the argument goes from $-\infty$ to $+\infty$. The larger the values of σ is the more abrupt the change occurs. The bipolar Sigmoid function is also smooth with a smooth derivative given by

$$f'(y_{\text{in}}) = \frac{\sigma}{2}\left[1 + f(y_{\text{in}})\right]\left[1 - f(y_{\text{in}})\right].$$

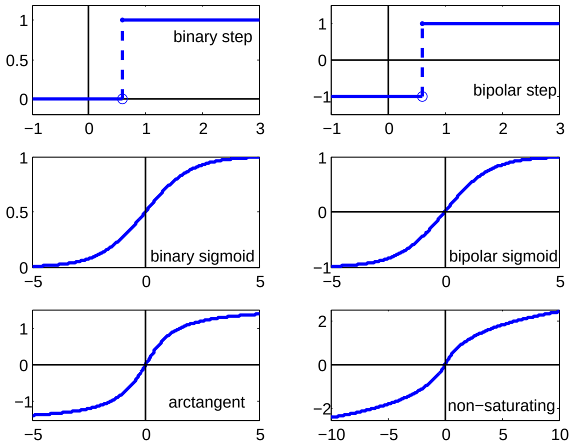

Figure 4: *Some common transfer functions.*

Thus its derivative can also be easily computed from the value of the original function itself.

- **An Alternate Bipolar Sigmoid Function**

The arctangent function can be used as an alternate for the bipolar Sigmoid function. It approaches its asymptotic values more slowly. It is given by

$$y = f(y_{\text{in}}) = \frac{2}{\pi} arctan(y_{\text{in}}),$$

and its derivative is

$$f'(y_{\text{in}}) = \frac{2}{\pi} \frac{1}{1 + y_{\text{in}}^2}.$$

- **Nonsaturating Activation Function**

For some applications, saturation of the activation is not especially beneficial. A nonsaturating activation function may be better. An example of such function is

$$y = f(y_{\text{in}}) = \begin{cases} \log(1 + y_{\text{in}}) & \text{if } y_{\text{in}} \geq 0, \\ -\log(1 + y_{\text{in}}) & \text{if } y_{\text{in}} < 0. \end{cases}$$

Note that the derivative is continuous at the origin:

$$f'(y_{\text{in}}) = \begin{cases} \frac{1}{1+y_{\text{in}}} & \text{if } y_{\text{in}} \geq 0, \\ \frac{1}{1-y_{\text{in}}} & \text{if } y_{\text{in}} < 0. \end{cases}$$

5.2. Network Architectures

The way how the neurons are connected to each other plays an important role in determining the function of a neural network.

It is often convenient to visualize neurons as arranged in layers. Neurons in the same layer have the same basic characteristics, that is they have the same form of activation function, and the same pattern of connections to the other neurons. However, they are expected to have different connection strengths and threshold values.

A NN typically have an input layer where the neurons all have activation functions given by the identity function. A signal enters the ANN from the input layer. It also has an output layer, where the output for each neuron is given by y.

All the remaining neurons in the NN are organized into layers known as the hidden layers. Neurons within a layer are connected to the neurons in its adjacent layer. There may be 0, 1, or more such hidden layers. NNs are classified by the total number of layers, not counting the input layer.

An ANN is called a feedforward net if the input signal going into the input layer propagates through each of the hidden layers and finally emerge from the output layer.

The figure shows the simplest of such a feedforward NN. There are N neurons in the input layer and they are labeled by $X_1, X_2, \ldots, X_N$. The signal in those neurons are denoted by $x_1, x_2, \ldots, x_N$, respectively. There is just a single output neuron, labeled by Y. The signal there, denoted by y, is the output signal of the network. The strengths of the connections between the input neurons and the output neuron are labeled by $w_1, w_2, \ldots, w_N$, respectively.

The input signal, represented by a vector $(x_1, x_2, \ldots, x_N)$, is trans-

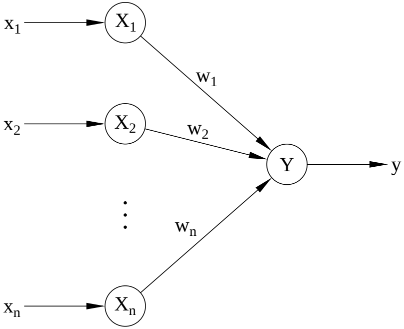

Figure 5: *The simplest feedforward NN having 1 output neuron.*

ported to the output neuron, Y, but each component of the input signal is weighted by the corresponding weight factor. The net signal received by Y is given by

$$n = x_1 w_1 + x_2 w_2 + \ldots + x_N w_N = \sum_{i=1}^{N} x_i w_i.$$

Y then applies its transfer function to the net signal it receives and produces the output signal

$$y = f(n).$$

Note that the output, y, is in general a nonlinear function of the input vector, $\mathbf{x}$. The only exception is the case where the transfer function happens to be a linear function.

The figure shows a single-layer feedforward NN having multiple output neurons ($M > 1$). They are labeled by $Y_1, Y_2, \ldots, Y_M$, and their outputs are $y_1, y_2, \ldots, y_M$, respectively. The weight connecting the i-th input neuron, X_i, and the j-th output neuron, Y_j, is denoted by w_{ij}. (Note some people uses a notation where the subscripts i and j are reversed.)

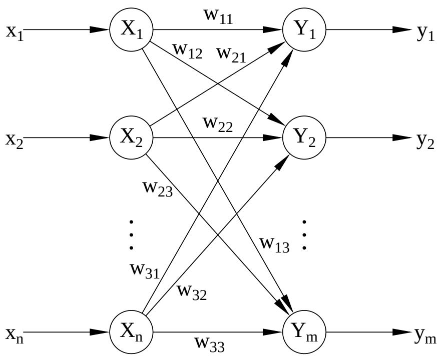

Figure 6: *A single-layer feedforward NN having multiple output neurons.*

A layer is called recurrent if it has closed-loop connection from a neuron back to itself.

5.3. Network Learning Algorithms

Algorithms for setting the weights of the connections between neurons are also called learning or training rules. There are three basic types of training rule.

- **Supervised Training**

In supervised training, the network is trained by presenting it a sequence of training inputs (patterns), each with an associated target output value. Weights in the net are adjusted according to a learning algorithm.

- **Unsupervised Training**

In unsupervised training, a sequence of training inputs is provided, but no target output values are specified. The weights are adjusted according to a learning algorithm.

- **Fixed-Weights Nets with no Training**

There are also nets whose weights are fixed at the outset. There is no training of the net at all.

5.4. Applications of NN

We will present examples illustrating the usefulness of NN to solve certain problems. For the moment we will accept the weights and thresholds as given and will be concerned with how they are obtained through appropriate training processes.

- **NN as logic functions**

The AND logic function is defined by the following table:

x_1	x_2	y
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1

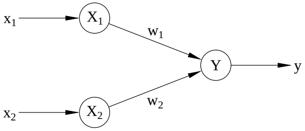

Figure 7: *A single-layer feedforward NN having 2 input and 1 output neurons.*

We are using bipolar values here rather than binary ones, and so the transfer function used for the output neuron is the bipolar step function. A NN with two input neurons and a single output neuron can operate as an `AND` logic function if we choose weights $w_1 = 1, w_2 = 1$ and threshold $\theta = 1$. We can verify that for

$$x_1 = 1, x_2 = 1 : \quad n = 1 \times 1 + 1 \times 1 = 2 \quad \text{is} > 1 \quad \text{so} \quad y = 1$$

$$x_1 = 1, x_2 = -1 : \quad n = 1 \times 1 + 1 \times (-1) = 0 \quad \text{is} < 1 \quad \text{so} \quad y = -1$$

$$x_1 = -1, x_2 = 1 : \quad n = 1 \times (-1) + 1 \times 1 = 0 \quad \text{is} < 1 \quad \text{so} \quad y = -1$$

$$x_1 = -1, x_2 = -1 : \quad n = 1 \times (-1) + 1 \times (-1) = -2 \quad \text{is} < 1 \quad \text{so} \quad y = -1$$

Note that many other choices for the weights and threshold will also work.

As another example, we consider the `OR` logic function:

x_1	x_2	y
1	1	1
1	-1	1
-1	1	1
-1	-1	-1

For a NN to operate as an `OR` function we choose weights $w_1 = 1, w_2 = 1$ and threshold $\theta = -1$. We can verify that for

$$x_1 = 1, x_2 = 1: \quad n = 1 \times 1 + 1 \times 1 = 2 \quad \text{is} > -1 \quad \text{so} \quad y = 1$$

$$x_1 = 1, x_2 = -1: \quad n = 1 \times 1 + 1 \times (-1) = 0 \quad \text{is} > -1 \quad \text{so} \quad y = 1$$

$$x_1 = -1, x_2 = 1: \quad n = 1 \times (-1) + 1 \times 1 = 0 \quad \text{is} > -1 \quad \text{so} \quad y = 1$$

$$x_1 = -1, x_2 = -1: \quad n = 1 \times (-1) + 1 \times (-1) = -2 \quad \text{is} < -1 \quad \text{so} \quad y = -1$$

Note that many other choices for the weights and threshold will also work.

As we will see in the next chapter, the NN as defined above cannot perform the `XOR` logic function:

x_1	x_2	y
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1

no matter what the values of the weights and threshold are. NN having a different architecture is needed.

- **NN as classifiers**

Other than operating as a logic function, NN can also be used to handle classification problems. For example, we want to classify customers into two classes. Class 1 are those who will likely buy a riding lawn-mower, and class 2 are those who will likely not do so. The decision is based on the customer's income and lawn size. For each customer, there is a vector

$$\mathbf{x} = (x_1, x_2),$$

where x_1 is the income in units of $100,000$ dollars, and x_2 is the lawn size in acres. For this example, we use the same NN with weights $w_1 = 0.7, w_2 = 1.1$ and threshold $\theta = 2.3$.

Using this NN, we find that a customer who makes $30,000$ a year and has a lawn size of 2 acres will likely buy a riding lawn-mower, since the net signal received by Y

$$0.3 \times 0.7 + 2 \times 1.1 = 0.21 + 2.2 = 2.41$$

is larger than the threshold $theta = 2.3$. On the other hand, a customer who makes $150,000$ a year and has a lawn size of 1 acre will likely not buy a riding lawn-mower, since the net signal received by Y

$$1.5 \times 0.7 + 1 \times 1.1 = 1.05 + 1.1 = 2.15$$

is smaller than the threshold.

In real applications, input vectors typically have a large number of components (i.e. $N \gg 1$).

References

[1] Laurene Fausett, "Fundamentals of Neural Networks - Architectures, Algorithms, and Applications", Prentice Hall, 1994.