APPENDIX - A

Introduction to Neural Networks

A1.1 Introduction

The neural network contains a large number of neuron like processing elements and a large number of weighted connections between the elements. The weights on the connections encode the knowledge of a network. Though biologically inspired, many of the neural network models developed do not duplicate the operation of the human brain. In many tasks such as recognizing human faces and understanding speech, current AI systems cannot do better than humans. It is conjectured that the structure of the brain somehow suited to these tasks and not suited to tasks such as high-speed arithmetic calculation. Further, one may link the connectionist structure to the non algorithmic nature of perceptual or cognitive reasoning. Rather than by being programmed, an artificial neural network can solve problems simply by example mapping.

A1.2 Basic concepts of Neural Networks

A neural network has a parallel-distributed architecture with a large number of nodes and connections. Each connection points from one node to another and is associated with a weight. Construction of a neural network involves the following tasks:

* Determine the network properties: the network topology (connectivity), the types of connections, the order of connections, and the weight range.

* Determine the node properties: the activation range and the activation (transfer) function.

* Determine the system dynamics: the weight initialization scheme, the activation-calculating formula, and the learning rule.

A1.2.1 *Network properties*

The *topology* of a neural network refers to its framework as well as its interconnection scheme. The framework is often specified by the number of *layers* (or slabs) and the number of nodes per layer. The types of layers include:

* The input layer: The nodes in it are called *input units,* which encode the instance presented to the network for processing. For example, each input unit may be designated by an attribute value possessed by the instance.

* The hidden layer: The nodes in it are called hidden units, which are not directly observable and hence hidden. They provide nonlinearities for the network.

* The output layer: The nodes in it are called *output units*, which encode possible concepts (or values) to be assigned to the instance under consideration. For example, each output unit represents a class of objects.

Input units do not process information; they simple distribute information to other units.

According to the interconnection scheme, a network can be either *feedforward* or *recurrent* and its connections either symmetrical or asymmetrical. Their definitions are given below:

* *Feedforward networks*: All connections point in one direction (from the input toward the output layer).

* *Recurrent networks*: There are feedback connections or loops.

* *Symmetrical connections*: If there is a connection pointing from node *i* to node *j*, then there is also a connection from node *j* to node *i,* and the weights associated with the two connections are equal, or notationally , $Wij = Wji$.

* *Asymmetrical connections:* If connections are not symmetrical as defined above, then they are asymmetrical.

A connection between nodes in different layers is called an *interlayer connection.* A connection between nodes within the same layer is called an *intralayer* connection. A connection pointing from a node to itself is called a *self -connection.* And a connection between nodes in distant (nonadjacent) layers is called a *supralayer connection* by some researchers. For example, full connectivity refers to how nodes are connected. For example, full connectivity often means that every node in one layer is connected to every node in its adjacent layer.

A *high-order connection* is a connection that combines inputs from more than one node, often by multiplication. The number of the inputs determines the order of the connection. The order of a neural network is the order of the highest -order connection. Neural networks are assumed to be first order unless mentioned otherwise.

Connection weights can be real numbers or integers. They can be confined to a range. They are adjustable during network training, but some can be fixed deliberately. When training is completed, all of them should be fixed.

A1.2.2 *Node properties*

The activation levels of nodes can be discrete (e.g., 0 and 1) or continuous across a range (e.g., [0,1]) or unrestricted. This depends on the activation (transfer) function chosen. If it is a hard-limiting function, then the activation levels are 0 (or -1) and 1. For a sigmoid function, the activation levels are limited to a continuous range of reals [0,1]. Figure A1.1 shows the sigmoid function F:

$$F(x) = 1 / (1 + e^{-x}) \qquad \text{(A1.1)}$$

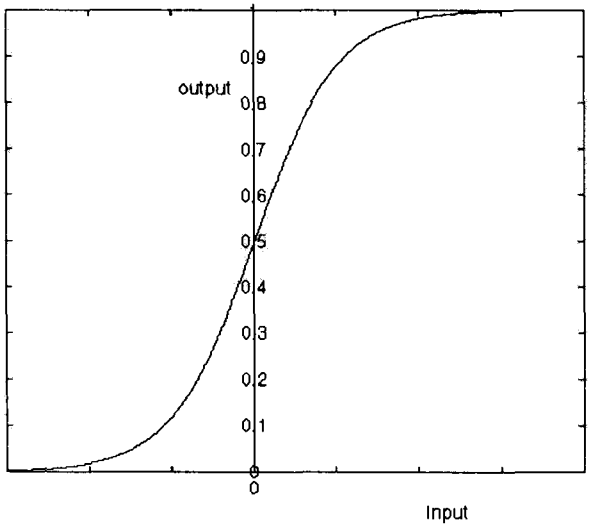

Figure A1.1: Sigmoid function

In case of a linear activation function, the activation levels are open. The activation function is mentioned again when we discuss the system dynamics.

A1.2.3 *System dynamics*

The weight initialization scheme is specific to the particular neural network model chosen. However, in many cases, initial weights are just randomized to small real numbers. The learning rule is one of the most important attributes to specify for a neural network. The learning rule determines how to adapt connection weights in order to

optimize the network performance. It indicates how to calculate the weight adjustment during each training cycle. However, the rule is suspended after training is completed.

When the neural network is used to solve a problem, the solution lies in the activation levels of the output units. The inference behaviour of a neural network involves how to compute the activation levels across the network. However one should note that training a neural network also involves the same calculation, since we need to know the actual activation levels and the desired activation levels so as to calculate the errors, which are then used as the basis for weight adjustment.

The activation levels of input units need not be calculated since they are given. Those of hidden and output units are calculated according to the activation function used. Provided that it is a sigmoid function, the activation level (Oi) of unit j is calculated by

$$O_i = 1/[1 + e^{-(\Sigma_j W_{ij} X_j - \varnothing_i)}] \qquad \text{(A1.2)}$$

Where Xj is the input from unit j, W_{ij} the weight on the connection from unit j to unit i and the threshold on unit. In the case of a hard-limiting activation function, the output of a neuron is given by

$$O_i = \begin{cases} 1, & \sum_j W_{ij} X_j > \theta_i \\ 0, & \text{else.} \end{cases} \qquad \text{(A1.3)}$$

A1.3 Inference and Learning

In the training (or learning) phase, a set of training instances is given. Each training instance is typically described by a feature vector (called an *input vector*). It may also be associated with a desired outcome (a concept, a class, etc.), which is encoded as another vector (called a desired *output vector*). Starting with some arbitrary or random weight setting, the neural network is trained to adapt itself to the characteristics of the training instances by changing weights inside the network. In each training cycle, we present an instance to the network. It generates an output vector, which is compared with the desired output vector (if available). In this way, the error for each output unit is calculated and then used to update relevant weights. In a multilayer network, the errors of hidden units are not observed directly but can be estimated with some heuristic. Each weight is hoped to reduce the error. When all instances are examined, the network will start over with the

first instance and repeat. Iterations continue until the system performance (in terms of the error magnitude) has reached a satisfactory level. In practice, we choose an error criterion which will be minimized during training. The criteria most commonly used for this purpose is sum of squared errors,

$$\sum_{p} \sum_{i} (T_{i,p} - O_{i,p})^2 \qquad \text{(A1.4)}$$

where $T_{i,p}$, and $O_{i,p}$ are, respectively, the desired and the actual activations of output unit i, in the instance p.

In the inference phase, the network propagates information from the input toward the output layer. When propagation stops, the output units carry the result of the inference.

Despite some specific variations with different models, general learning and inference procedures can be extracted as follows.

The Neural Network Learning Algorithm (A general view)

Given n training instances,

1. Initialize the network weights. Set i =1.
2. Present the i_{th} instance to the network on the input layer.
3. Obtain the activation levels of the output units using the inference algorithm (described next). If the network performance meets the predefined standard (or the stopping criterion), then exit.
4. Update the weights by the learning rule of the network.
5. If $i = n$ then reset i = 1. Otherwise, increment i by 1. Go to step 2.

The Neural Network Inference Algorithm (A general view)

1. Present the instance to the network on the input layer.
2. Calculate the activation levels of nodes across the network.
3. For a feedforward network, if the activation levels of all ouput units are calculated, then exit. For a recurrent network, if the activation levels of all output units become (near) constant, then exit; else go to step 2. However if the network is found unstable, then exit and fail.

A1.3.1 *Data representation*

A discrete feature value (e.g., the feature *colour* with values like *red, yellow, blue*, and so on) can be encoded by a single input unit. In this case an activation of 1 corresponds to the notion of "yes," or "true," whereas an activation of 0 corresponds to "no," or "false". The activation may fall between 0 and 1, indicating the probability of the value.

Continuous feature values (e.g., the feature *age* with values ranging from 1 to 150) can be represented in several ways. In continuous representation, each feature is encoded by an input unit and the feature value is mapped into the unit activation. In practice, the range of continuous values is often normalized to the interval between 0 and 1. It is also possible to represent a continuous value as a binary (rather than a decimal) number. For example, 8 is represented by four bits 1000; 7 is represented by 0111.

A1.3.2 *Functional classification of Neural Network models*

Neural computational models can categorized in terms of their applications:

* Classification: assignment of the input data to one of a finite number of categories. E.g, Single layer perceptron, Multilayer Perceptron
* Association
  - -Autoassociation: retrieval of an object (memory) based on part of the object (memory) itself. E.g. Hopfield nets
  - -Heteroassociation: retrieval of an object (memory) in one set using another object (memory) in a different set. E.g., Bidirectional Associative Memories (BAMs)
* Optimization: finding the best solution - often by minimizing a certain cost function. E.g. Boltzmann Machines.
* Self- organization: organizing received information using adaptive learning capabilities. E.g. Khonen networks, Competitive learning, Hebbian learning.

In this appendix only basic classification models single layer perceptron and multi-layer perceptron are dealt since they are more relevant to this thesis.

A1.4 Classification models

A neural network classifies a given object presented to it according to the output activation. For binary outputs, 1 corresponds to one class and 0 corresponds to the other. For continuous outputs between 0 and 1, 0.5 can be used as the threshold to make the decision. In the case of multiple outputs, the object is assigned to the class corresponding to the output node with the maximum activation. In this section, we examine the classification models based on the neural network.

A1.4.1 *Single -layer Perceptrons*

A single-layer perceptron consists of an input and an output layer. The activation function employed is a hard-limiting function. An output unit will assume the value 1 if the sum of its weighted inputs is greater than its threshold. In terms of classification, an object will be classified by unit i into class A if

$$\sum W_{ij} X_j > \theta_i \qquad \text{(A1.5)}$$

where W_{ij} is the weight from unit j, X_j is the input from unit j and $\emptyset i$ is the threshold on unit i. Otherwise, the object will be classified as class B. Suppose there are n inputs. The equation

$$\sum_{i=1}^{n} Wij\, Xj = 0_i \qquad \text{(A1.6)}$$

forms a hyperplane in the n-dimensional space, dividing the space into two halves. When n =2, it becomes a line. *Linear separability* refers to the case when a linear hyperplane exists to place the instances of one class on one side and those of the other class on the other side of the plane. Unfortunately, many classification problems are not linearly separable. The exclusive-or problem is a good example. To cope with a problem which is not linearly separable, a multilayer perceptron is required.

A1.4.2 *Multilayer Peerceptrons*

A multilayer perceptron is a feedforward neural network with at least one hidden layer. It can deal with nonlinear classification problems because it can form more complex decision regions (rather than just hyperplanes). Each node in the first layer (above the input layer) can create a hyperplane. Each node in the second layer can combine

hyperplane to create convex decision regions. Each node in the third layer can combine convex regions to form concave regions. The idea is illustrated in figure A1.2 and A1.3. It is thus possible to form any arbitrary regions with sufficient layers and sufficient hidden units.

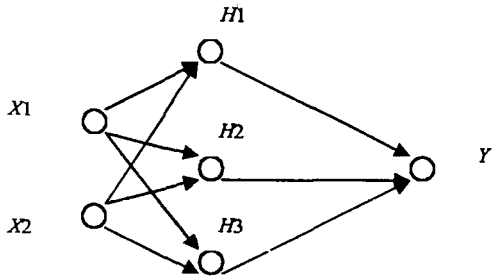

Figure A1.2: Multi-layer Perceptron.

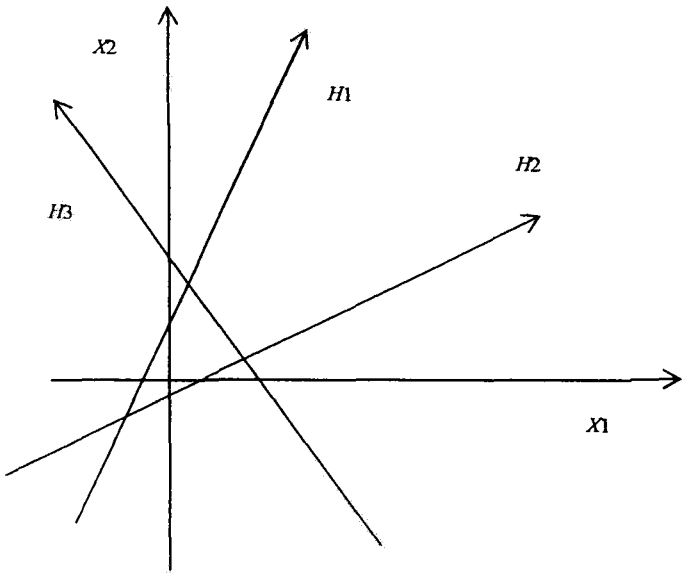

Figure A1.3 : Decision regions created by a multilayer Perceptron.

A1.5 Applications of Neural Networks

A comprehensive list of all applications of neural networks is impractical. However, based on journal reports, conference proceedings the following list of illustrative examples are provided. It include

Aerospace: High performance aircraft autopilot, flight path simulation, aircraft control systems, autopilot enhancements, aircraft component simulation, aircraft component fault detection.

Automotive: Automobile automatic guidance system, warranty activity analysis.

Banking: Cheque and other document reading, credit application evaluation.

Defence: Weapon steering, target tracking, object discrimination, facial recognition, new kinds of sensors, sonar, radar and image signal processing including data compression, feature extraction and noise suppression, signal/image identification.

Electronics: Code sequence prediction, integrated circuit chip layout, process control, chip failure analysis, machine vision, voice synthesis, nonlinear modeling.

Entertainment: Animation, special effects, market forecasting.

Financial: Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, credit line use analysis, portfolio trading program, corporate financial analysis, and currency price prediction.

Insurance: Policy application evaluation, product optimization.

Manufacturing: Manufacturing process control, product design and analysis, process and machine diagnosis, real-time particle identification, visual quality inspection systems, beer testing, welding quality analysis, paper quality prediction, computer chip quality analysis, analysis of grinding operations.

Medical: Breast cancer cell analysis, EEG and ECG analysis, prosthesis design, optimization of transplant times, hospital expense reduction, hospital quality improvement.

Oil and Gas: Exploration.

Robotics: Trajectory control, forklift robot, manipulator controllers, vision systems.

Speech: Speech recognition, speech compression, vowel classification, text to speech synthesis.

Securities: Market analysis, automatic bond rating, stock trading advisory systems.

Telecommunication: Image and data compression, automated information services, real-time translation of spoken language, customer payment processing systems.

Transportation: Truck brake diagnosis systems, vehicle scheduling, routing systems.