The Technical Aspects of Artificial Neural Networks 3.1

Artificial Neural Network Models ccording to Nelson and Illingworth [1990], there are infinitely many ways to organize a neural network although perhaps only two dozen models are in common usage. A neural network organization can be described in terms of its neurodynamics and architecture. Neurodynamics refer to the properties of an individual artificial neuron that consist of the following: · combination of input(s); · production of output(s); · type of transfer (activation) functions; and · weighting schemes, i.e. weight initialization and weight learning algorithms. These properties can also be applied to the whole network on a system basis. Network architecture (also sometimes referred to as network topology) defines the network structure and includes the following basic characteristics: · types of interconnections among artificial neurons (henceforth referred to as just neurons21); · number of neurons and · number of layers 21 As mentioned earlier, they are also called processing elements, neurodes, nodes, units, etc. A Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 51 3.2 Neurodynamics 3.2.1 Inputs The input layer of an ANN typically functions as a buffer for the inputs, transferring the data to the next layer. Preprocessing the inputs may be required as ANNs deal only with numeric data. This may involve scaling the input data and converting or encoding the input data to a numerical form that can be used by the ANN. For example, in an ANN real estate price simulator application described in a paper by Haynes and Tan [1993], some qualitative data pertaining to the availability of certain features of a residential property used a binary representation. For example, features like the availability of a swimming pool, a granny flat and a waterfront location, were represented with a binary value of ‘1’, indicating the availability of the feature, or ‘0’ if it was not. Similarly, a character or an image to be presented to an ANN can be converted into binary values of zeroes and ones. For example, the character ‘T’ can be represented as shown in Figure 3-1. Figure 3-1 The binary representation for the letter ‘T’ 1111111 0001000 0001000 0001000 3.2.2 Outputs The output layer of an ANN functions in a similar fashion to the input layer except that it transfers the information from the network to the outside world. Post-processing of the output data is often required to convert the information to a comprehensible and usable form outside the network. The post-processing may be as simple as just a scaling of the outputs ranging to more elaborate processing as in hybrid systems. Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 52 For example, in chapter 4 of this thesis, on the prediction of financial distress in credit unions, the post-processing is relatively simple. It only requires the continuous output values from the ANN to be converted into a binary form of ‘1’ (indicating a credit union in distress) or ‘0’ (indicating a credit union is not in distress). However, in the foreign exchange trading system application in chapter 5, the post-processing of the network output is more complex. The ANN output is the predicted exchange rate but the trading system output requires a trading signal to be generated from the ANN output. Thus, the ANN output has to go through a set of rules to produce the trading signal of either a ‘Buy’ or ‘Sell’ or ‘Do Nothing’. 3.2.3 Transfer (Activation) Functions The transfer or activation function is a function that determines the output from a summation of the weighted inputs of a neuron. The transfer functions for neurons in the hidden layer are often nonlinear and they provide the nonlinearities for the network. For the example in Figure 3-2, the output of neuron j, after the summation of its weighted inputs from neuron 1 to i has been mapped by the transfer function f can be shown as: O f w x j j ij i i = æ è ç ö ø å ÷ Equation 3-1 Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 53 Figure 3-2 Diagram of the Neurodynamics of Neuron j S w w w w x ij i x x x Neuron j 1 2 i w1j w2j wij h j Oj = f(h j ) Transfer function f A transfer function maps any real numbers into a domain normally bounded by 0 to 1 or -1 to 1. Bounded activation functions are often called squashing functions [Sarle 1994]. Early ANN models, like the perceptron used, a simple threshold function (also known as a step-function, hard-limiting activation or Heaviside function): · threshold: f (x) = 0 if x < 0, 1 otherwise. The most common transfer functions used in current ANN models are the sigmoid (Sshaped) functions. Masters [1993] loosely defined a sigmoid function as a ‘continuous, real-valued function whose domain is the reals, whose derivative is always positive, and whose range is bounded’. Examples of sigmoid functions are: · logistic: f (x) = 1 1+ - e x · hyperbolic tangent: f (x) = e e e e x x x x - + - - The logistic function remains the most commonly applied in ANN models due to the ease of computing its derivative: Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 54 f x f x f x ' ( ) = ( )(1- ( )) The output, Oj, of the neuron x j of the earlier example in Figure 3-2 if the function f is a logistic function becomes: O e j w xij i j i = + -å - 1 1 q Equation 3-2 where qj is the threshold on unit j. If the function f is a threshold function instead, the output, Oj will be: O w x else j ij i j = i ì > í ï î ï 1 å 0 , , q Equation 3-3 However, Kalman and Kwasny [1992] argue that the hyperbolic tangent function is the ideal transfer function. According to Masters [1993], the shape of the function has little effect on a network although it can have a significant impact on the training speed. Other common transfer functions include: · linear or identity: f (x) = x Normally used in the input and/or output layer. · Gaussian: f (x) = e - x 2 /2 Sigmoid functions can never reach their theoretical limit values and it is futile to try and train an ANN to achieve these extreme values. Values that are close to the limits should be considered as having reaching those values. For example, in a logistic function where the limits are 0 to 1, a neuron should be considered to be fully activated at values around 0.9 and turned off at around 0.1. This is another reason why ANNs cannot do numerical Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 55 computation as well or as accurate as simple serial computers; i.e. a calculator. Thus ANNs is not a suitable tool for balancing check books! 3.2.4 Weighing Schemes and Learning Algorithms The initial weights of an ANN are often selected randomly or by an algorithm. The learning algorithm determines how the weights are changed, normally depending on the size of the error in the network output to the desired output. The objective of the learning algorithm is to minimize this error to an acceptable value. The back-propagation algorithm is by far the most popular learning algorithm for multilayer networks and will be discussed in more detail in section 3.4.1.2. 3.3 Neural Networks Architecture 3.3.1 Types of interconnections between neurons A network is said to be fully connected if the output from a neuron is connected to every other neuron in the next layer. A network with connections that pass outputs in a single direction only to neurons on the next layer is called a feedforward network. Nelson and Illingworth [1990] define a feedback network as one that allows its outputs to be inputs to preceding layers. They call networks that work with closed loops as recurrent networks. They also mention networks with feedlateral connections that would send some inputs to other nodes in the same layer. Feedforward networks are faster than feedback nets as they require only a single pass to obtain a solution. According to Nelson and Illingworth [1990] recurrent networks are used to perform functions like automatic gain control or energy normalization and selecting a maximum in complex systems. Most ANN books, however, classify networks into two categories only: feedforward networks and recurrent networks. This is done by classifying all networks with feedback Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 56 connections or loops as recurrent networks. Fully connected feedforward networks are often called multi-layer perceptrons (MLPs) and they are by far the most commonly used ANNs. All the ANNs used in this thesis are MLPs. They will be discussed in more detail in section 3.3.6. 3.3.2 The Number of Hidden Neurons Hidden neurons are required to compute difficult functions known as nonseparable functions which are discussed in section 3.3.5. The number of input and output neurons are determined by the application at hand. However, there are no standard rules or theories in determining the number of neurons in the hidden layers although there are some rules of thumb suggested by various ANN researchers: · Shih [1994] suggested that the network topology should have a pyramidal shape; that is to have the greatest number of neurons in the initial layers and have fewer neurons in the later layers. He suggested the number of neurons in each layer should be a number from mid-way between the previous and succeeding layers to twice the number of the preceding layer. The examples given suggest that a network with 12 neurons in its previous layer and 3 neurons in the succeeding layer should have 6 to 24 neurons in the intermediate layer. · According to Azoff [1994], a rough guideline based on theoretical conditions of what is know as the Vapnik-Chervonenkis dimension22, recommends that the number of training data should be at least ten times the number of weights. He also quoted a theorem due to Kolmogorov [Hecht-Nielsen 1990 and Lippman 1987] that suggests a network with one hidden layer and 2N+1 hidden neurons is sufficient for N inputs. 22 Azoff referred to an article by Hush and Horne [1993]. Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 57 · Lawrence [1994, p. 237] gives the following formula for determining the number of hidden neurons required in a network: number of hidden neurons = training facts ´ error tolerance. Note: training facts refers to in-sample data while the error tolerance refers to the level of accuracy desired or acceptable error range. · Baum and Haussler [1988] suggest that the number of neurons in the hidden layer should be calculated as follows: j me n z = + where j is the number of neurons in the hidden layer, m is the number of data points in the training set, e is the error tolerance, n is the number of inputs and z the number of outputs. The latter two rules of thumb are very similar and may not be meaningful in cases where the error tolerances are significantly smaller than the number of training facts. For example, if the number of training facts is 100 and the error tolerance is 0.001, the number of hidden neurons would be 0.1 (meaningless!) in Lawrence’s proposal; while Baum and Hassler’s proposal would result in an even lower value. Most statisticians are not convinced that rules of thumbs are of any use. They argue that there is no way to determine a good network topology from just the number of inputs and outputs [Neural Network FAQ 1996]. The Neural Network FAQ [1996] suggests a method called early stopping or stopped training whereby a larger number of hidden neurons are used with a very slow learning rate and with small random initial weight values. The out-of-sample error rate is computed periodically during training. The training of the network is halted when the error rate in the out-of-sample data starts to increase. A similar method to early stopping is used in the development of the ANNs applications for the financial distress and foreign exchange Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 58 trading problems of this thesis. However, those ANNs do not use ‘lots of hidden units’ as suggested by the article. Instead, they start with small numbers of hidden neurons with the numbers increased gradually only if the ANNs do not seem to ‘learn’. In this way, the problem of overfit or curve-fit which can occur when there are more weights (parameters) than sample data can be avoided. However, a recent report by Lawrence et al. [1996] suggest that using “oversize” networks can reduce both training and generalization error. 3.3.3 The Number of Hidden Layers According to the Neural Network FAQ [1996], hidden layers may not be required at all. It uses McCullagh and Nelder’s [1989] paper to support this view. They found linear and generalized linear models to be useful in a wide variety of applications. They suggest that even if the function to be learned is mildly non-linear, a simple linear model may still perform better than a complicated nonlinear model if there is insufficient data or too much noise to estimate the nonlinearities accurately. MLPs that uses the step/threshold/Heaviside transfer functions need two hidden layers for full generality [Sontag 1992], while an MLP that uses any of a wide variety of continuous nonlinear hidden-layer transfer functions requires just one hidden layer with ‘an arbitrarily large number of hidden neurons’ to achieve the ‘universal approximation’ property described by Hornik et al. [1989] and Hornik [1993]. 3.3.4 The Perceptron The perceptron model, as mentioned in earlier chapters, was proposed by Frank Rosenblatt in the mid 1960s. According to Carling [1992], the model was inspired by the discovery of Hubel and Wiesel [1962] of the existence of some mechanism in the eye of a cat that can determine line directions. Rosenblatt developed the perceptron learning theorem (that was Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 59 subsequently proved by Arbib [1989]) which states that if a set of patterns is learnable by a perceptron, then the perceptron is guaranteed to find the appropriate weight set. Essentially, Rosenblatt’s perceptron model was an ANN model consisting of only an input layer and an output layer with no hidden layer. The input and output layers can have one or more neurons. Rosenblatt’s model uses a threshold function as a transfer function although the perceptron model can use any of the transfer functions discussed in section 3.2.3. Therefore if the sum of the inputs is greater than its threshold value, the output neuron will assume the value of 1, or else a value of 0. Fu [1994] states that in terms of classification, an object will be classified by neuron j into Class A if w x å ij i > q Equation 3-4 where wij is the weight from neuron i to neuron j, xi is the input from neuron i, and q is the threshold on neuron j. If not, the object will be classified as Class B. The weights on a perceptron model like the one shown in Figure 3-3 are adjusted by wij t wij t wij ( +1) = ( ) + D Equation 3-5 where wij(t) is the weight from neuron i to neuron j at time t (to the tth iteration) and Dwij is the weight adjustment. The weight change is computed by using the delta rule: Dw x ij = hd j i Equation 3-6 where h is the learning rate (0<h 2, then Equation 3-4 becomes: w xij j j i n = = å q 1 Equation 3-8 forming a hyperplane of n-1 dimension in the n-dimensional space (also called hyperspace), dividing the space into two halves. According to Freeman and Skapura [1991, pp. 24-30], many real life problems require the separation of regions of points in hyperspace into individual categories, or classes, which must be distinguished from other classes. This type of problem is also known as a classification problem. Classification problems can be solved by finding suitable arrangements of hyperplanes that can partition n-dimensional space into various distinct regions. Although this task is very difficult for n>2 dimensions, certain ANNs (e.g. MLPs) can learn the proper partitioning by themselves. As mentioned in the last section, the perceptron can solve most binary Boolean functions. In fact, all but two of the sixteen possible binary Boolean functions, which are the XOR and its complement, are linearly separable and can be solved by the perceptron. The XOR is a function that outputs a 1 if and only if its two inputs are not the same, otherwise the output is 0. The truth table for the XOR function is shown in Table 3-1. Gallant [1993] showed that a perceptron model (which he called a single-cell linear discriminant model) can easily compute the AND, OR and NOT functions. Thus, he defined a Boolean function to be a separable function if it can be computed by a single-cell Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 62 linear discriminant model; otherwise it is classified as a nonseparable function. He further states that the XOR is the simplest nonseparable function in that there are no nonseparable function with fewer inputs. Application of the perceptron model of Figure 3-6 to the XOR problem yields: Output, Oj = f(hj) = f(w1jx1 + w2jx2,q) = 1 0 1 1 2 2 1 1 2 2 , , ì í î + ³ + < w x w x w x w x j j j j q q Equation 3-9 where wij is the weight on the connection from neuron i to j and xi is the input neuron i, hj is the neuron j’s activation value and q is the threshold value of the threshold function f. A set of values must be found so that the weights can achieve the proper output value. We will show that this cannot be done. From Equation 3-9, a line on the x1 and x2 plane is obtained: q = w1jx1+w2jx2 Equation 3-10 By plotting the XOR function and this line for some values of q, w1 and w2 on the x1 and x2 plane in Figure 3-4, we can see that it is impossible to draw a single line to separate the 1s (represented by the squares) and the 0s (represented by the circles). The next section will demonstrate how a multilayer perceptron (MLP) can be used to solve this problem. Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 63 Figure 3-3 A Simple Perceptron Model Figure 3-4 A plot of the Exclusive-Or function showing that the two groups of inputs (represented by squares and circles) cannot be separated with a single line. -1 -1 1 1 X1 X2 q = w1x1 + w2x2 x1 x2 w1j w2j Output, Oj=f(hj,qj) Inputs hj j Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 64 Table 3-1 Truth Table for the Exclusive-Or Function X1 X2 Output 0 0 0 0 1 1 1 0 1 1 1 0 3.3.6 The Multilayer Perceptron As mentioned in earlier sections, an MLP (also called a multilayer feedforward network) is an extension of the perceptron model with the addition of hidden layer(s) that have nonlinear transfer functions in the hidden neurons. We have also mentioned that an MLP having one hidden layer is a universal approximator, and is capable of learning any function that is continuous and defined on a compact domain23 as well as functions that consist of a finite collection of points. According to Masters [1993, pp. 85-90], the MLPs can also learn many functions that do not meet the above criteria; specifically discontinuities can be theoretically tolerated and functions that do not have compact support (such as normally distributed random variables) can be learned by a network with one hidden layer under some conditions24. Masters states that in practice, a second hidden layer is only required if 23 A compact domain means that the inputs have definite bounds, rather than having no limits on what they can be. 24 Kurkova [1995] has since, proven this theoretical assumption. Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 65 a function that is continuous has a few discontinuities. He further states the most common reason for an MLP to fail to learn is the violation of the compact domain assumption, i.e. the inputs are not bounded . He concludes that if there is a problem learning in an MLP, it is not due to the model itself but to either insufficient training, or insufficient number of neurons, insufficient number of training samples or an attempt to learn a supposed function that is not deterministic. 3.3.6.1 Solving the XOR Problem with A Multilayer Perceptron Model An MLP model that successfully solves the XOR problem is shown in Figure 3-5. The model incorporates two hidden neurons in the hidden layer. The appropriate weights and threshold values for each neuron are also shown in the diagram. A plot of the XOR function and the two resulting lines from the model is shown in Figure 3-6. The lines have separated the plane into three regions; the central region is associated with the network output of 1 and the remaining two regions containing the points (0,0) and (1,1) are associated with the output of 0. Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 66 Figure 3-5 A Multilayer Perceptron Model That Solves the XOR Problem (adapted from Freeman and Skapura 1991, p.29) Figure 3-6 A Possible Solution to the XOR Problem By Using Two Lines to Separate the Plane into Three Regions x1 x2 0.6 -0.2 Inputs hj q = 0.5 q = 0.5 q = 1.5 -1 -1 Output, Oj=f(hj,qj) 1 1 X1 X2 1.5 0.5 1.5 0.5 Output = 0 Output = 1 Output = 0 Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 67 3.4 Learning Learning is the weight modification process of an ANN in response to external input. There are three types of learning: 1. Supervised learning It is by far the most common type of learning in ANNs. It requires many samples to serve as exemplars. Each sample of this training set contains input values with corresponding desired output values (also called target values). The network will then attempt to compute the desired output from the set of given inputs of each sample by minimizing the error of the model output to the desired output. It attempts to do this by continuously adjusting the weights of its connection through an iterative learning process called training. As mentioned in earlier sections, the most common learning algorithm for training the network is the back-propagation algorithm. Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 68 1. Unsupervised learning It is sometimes called self-supervised learning and requires no explicit output values for training. Each of the sample inputs to the network is assumed to belong to a distinct class. Thus, the process of training consists of letting the network uncover these classes. It is not as popular as supervised learning and is not used in this thesis and hence will not be considered further. 1. Reinforcement learning It is a hybrid learning method in that no desired outputs are given to the network, but the network is told if the computed output is going in the correct direction or not. It is not used in this thesis and hence will not be considered further. 3.4.1 Learning Algorithms Although there are many learning algorithms (rules) in common used, this section will only discuss the two most popular ones: the delta rule and its generalization, the backpropagation algorithm. The learning procedures have to select the weights {wij} and the ‘biases’ {qj} which is usually taken to be one [Ripley 1993] by minimizing the total squared error, E: E t o p p p = å - 1 2 2 Equation 3-11 where o p is the output for input x p , tp is the target output and the p indexes the patterns in the training set. Both the delta rule and the backpropagation algorithms are a form of the gradient descent rule, which is a mathematical approach to minimizing the error between the actual and desired outputs. They do this by modifying the weights with an amount Chapter 3: The Technical and Statistical Aspects of Artificial Neural Networks CNW Tan Page 69 proportional to the first derivative of the error with respect to the weight. The gradient descent is akin to trying to move down to the lowest value of an error surface from the top of a hill without falling into any ravine. 3.4.1.1 The Delta Rule/ Least Mean Squares (LMS) (Widrow-Hoff) The Least Mean Square (LMS) algorithm was first proposed by Widrow and Hoff (hence, it is also called the Widrow-Hoff Rule) in 1960 when they introduced the ADALINE (Adaptive Linear), an ANN model that was similar to the perceptron model except that it only has a single output neuron and the output activation is a discrete bipolar function25 that produces a value of 1 or -1. The LMS algorithm was superior to Rosenblatt’s perceptron learning algorithm in terms of speed but it also could not be used on networks with hidden layers. Most literature claims the Delta Rule and the LMS Rule are one and the same [Freeman and Skapura 1991, p. 96, Nelson and Illingworth 1991, p. 137, Carling 1992, p.74, HechtNielsen 1990, p. 61]. They are, in terms of the weight change, Dwij, formula given in Equation 3-6: Dw x ij = hd j i Equation 3-6 where h is the learning rate (0<h<h