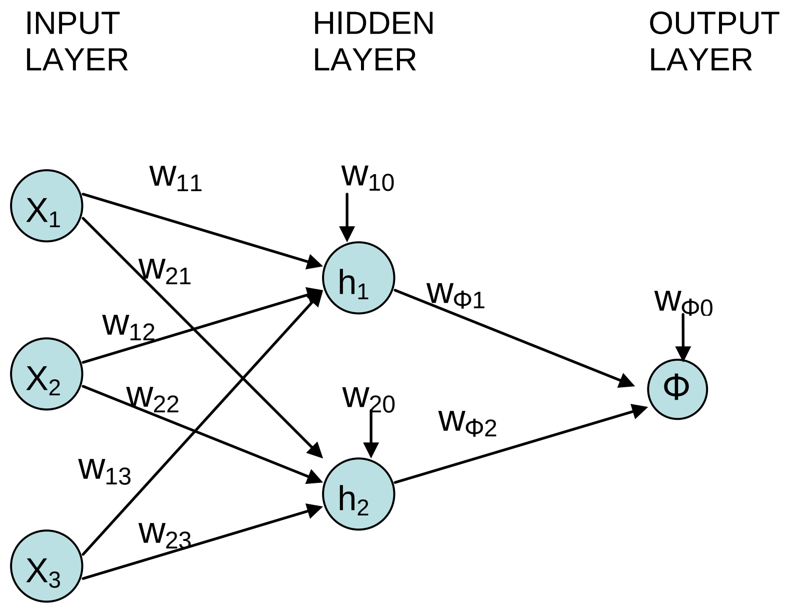
ARTIFICIAL NEURAL NETWORKS
(ANNs)

Professor Tom Fomby
Department of Economics
Southern Methodist University
March 2008

Artificial Neural Networks (hereafter ANNs) can be used for either prediction or classification problems. ANNs are based on representations of neural activity in the brain. The most popular design for ANNs is the so-called **multilayer feed-forward network**. Such networks have an **input layer**, an **output layer**, and one or more **hidden layers**. The following "architectural" diagram represents a 3-2-1 prediction ANN. The first number, 3, represents the number of inputs in the input layer; the second number, 2, represents the number of "neurons" or "nodes", 2, in the hidden layer, and the last number, 1, represents the number of nodes in the output layer.

3-2-1 ANN Architecture



X_1, X_2, X_3 = three inputs. h_1, h_2 = hidden node in hidden layer
Φ = output. w_{10}, w_{20}, $w_{\Phi 0}$ = "bias" weights, otherwise
w's first subscript = hidden node number
w's second subscript = input number
A total of 11 weights to be determined.

In the above diagram $x_1, x_2,$ and x_3 represent the three inputs, the two hidden nodes are represented by h_1 and h_2, and the output node is represented by ϕ. Moreover the "weights" connecting the three inputs to the first hidden layer are represented by w_{11}, w_{12}, and w_{13} where the first subscript, 1, represents linkage to the first node of the hidden layer, whereas the second subscript represents the input that the weight is associated with. Similarly, the weights connecting the three inputs to the second node of the hidden layer are represented by w_{21}, w_{22}, and w_{23}. In going from the hidden layer to the output layer the weights for the two hidden nodes are represented by $w_{\phi 1}$ and $w_{\phi 2}$. In addition to these "connecting" weights, this ANN also has bias weights w_{10}, w_{20}, and $w_{\Phi 0}$.

In ANNs the weights are applied vis-à-vis so-called "**squashing**" or **transfer** functions, say $f(z)$. Popular squashing functions include the logistic function, the arc tangent function, and the linear function. In XLMINER **prediction problems** the hidden layer squashing function is the logistic function, whereas the squashing function for the output layer(s) is the linear squashing function. This is because in prediction problems the output variable is an interval variable and can potentially range in value from $-\infty$ to $+\infty$. In contrast, **for classification problems**, the output squashing function(s) is the logistic function because, in classification problems, the output variables are of the binary form.

To demonstrate the estimated form of a simple 3-2-1 ANN we consider the Boston Housing data. The output variable is MEDV, the median value of homes in the given Boston housing district, while the input variables are RM, AGE, and DIS. The ANN output for a 60% training data set with the inputs being normalized (more about this later) is reproduced below:

Inter-layer connections weights

	Input Layer			
Hidden Layer # 1	RM	AGE	DIS	Bias Node
Node # 1	-2.27656087	-0.02952365	0.539700908	0.372506819
Node # 2	-3.30116537	1.235438919	0.712189365	1.160557507

	Hidden Layer # 1		
Output Layer	Node # 1	Node # 2	Bias Node
Output Node	-1.94365018	-2.87373442	1.832804968

The input weights going into the hidden nodes are as follows: $w_{11} = -2.27656087$, $w_{12} = -0.02952365$, $w_{13} = 0.539700908$, $w_{21} = -3.30116537$, $w_{22} = 1.235438919$, $w_{23} = 0.712189365$. The hidden node weights going into the output layer are $w_{\phi 1} = -1.94365018$ and $w_{\phi 2} = -2.87373442$. In addition to these weights you have the "**bias node**" weights $w_{10} = 0.372506819$ and $w_{20} = 1.160557507$ associated with, respectively, the first and second hidden layer nodes while the bias node weight for going from the hidden layer to the output layer is $w_{\phi 0} = 1.832804968$. Mathematically then, the hidden nodes are represented by

$$h_1 = \frac{1}{1+e^{-z_1}} \quad \text{and} \quad h_2 = \frac{1}{1+e^{-z_2}}$$

where the squashing function is the logistic function and

$$\begin{aligned} z_1 &= w_{10} + w_{11}x_1 + w_{12}x_2 + w_{13}x_3 \\ &= 0.372506819 - 2.27656087\text{RM} - 0.02952365\text{AGE} + 0.539700908\text{DIS} \end{aligned}$$

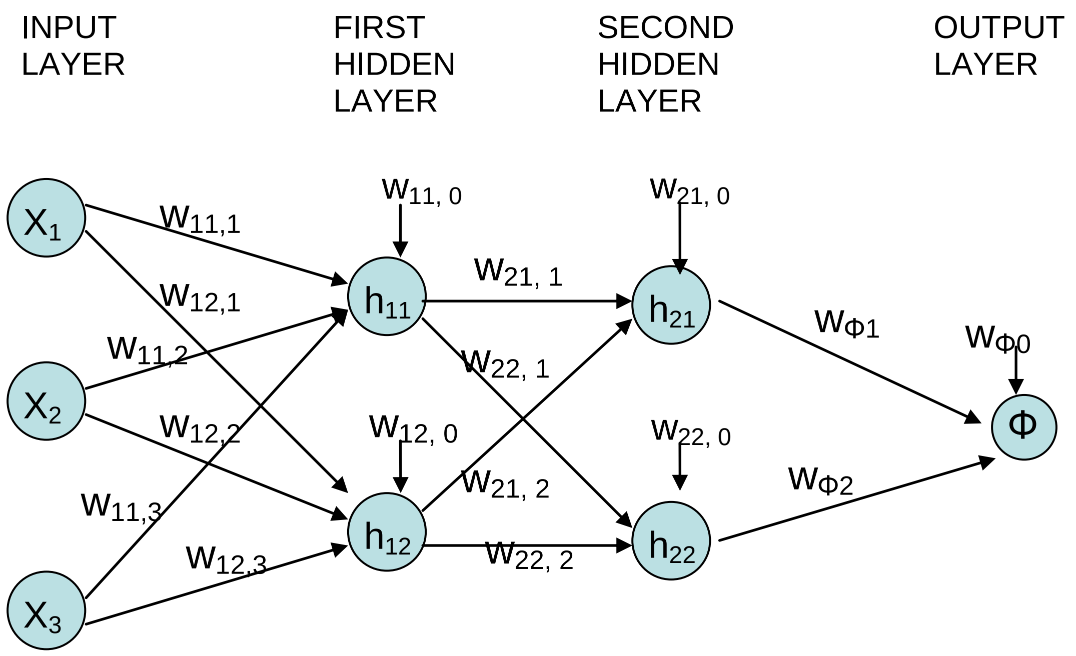
$$\begin{aligned} z_2 &= w_{20} + w_{21}x_1 + w_{22}x_2 + w_{23}x_3 \\ &= 1.160557507 - 3.30116537\text{RM} + 1.235438919\text{AGE} + 0.712189365\text{DIS}. \end{aligned}$$

Given these hidden node values we can get the output node by using the linear squashing function resulting in

$$\begin{aligned} \phi &= w_{\phi 0} + w_{\phi 1}h_1 + w_{\phi 2}h_2 \\ &= 1.832804968 - 1.94365018\ h_1 - 2.87373442\ h_2 . \end{aligned}$$

To score the validation data set we feed the input values of each validation case into the hidden node formulas and then get the output score by using the output formula immediately above. This model is highly nonlinear in the weights that must be estimated from the training data set and, unfortunately, a conventional method like least squares is inappropriate. Instead a frequently used method for determining the weights of this model is the **Back Propagation method** which we will discuss subsequently. But before we do, let's consider the two-hidden-layer ANN 3-2-2-1 which is represented diagrammatically below.

3-2-2-1 ANN Architecture



X_1, X_2, X_3 = three inputs
h_{11}, h_{12} = hidden node of First hidden layer
h_{21}, h_{22} = hidden node of Second hidden layer
$w_{ij,\,k}$, = i, j hidden node weight associated with k-input (or node in layer)
$w_{11,\,0}$, $w_{12,\,0}$, $w_{21,\,0}$, $w_{22,\,0}$, $w_{\Phi 0}$ = “bias” weights

Then the two nodes in the first hidden layer are represented by

$$h_{11} = \frac{1}{1+e^{-z_{11}}} \text{ and } h_{12} = \frac{1}{1+e^{-z_{12}}}$$

where

$$z_{11} = w_{11,0} + w_{11,1}x_1 + w_{11,2}x_2 + w_{11,3}x_3$$

and

$$z_{12} = w_{12,0} + w_{12,1}x_1 + w_{12,2}x_2 + w_{12,3}x_3 \,.$$

The two nodes of the second hidden layer are represented by

$$h_{21} = \frac{1}{1+e^{-z_{21}}} \text{ and } h_{22} = \frac{1}{1+e^{-z_{22}}} \text{ where}$$

$$z_{21} = w_{21,0} + w_{21,1}h_{11} + w_{21,2}h_{12}$$

and

$$z_{22} = w_{22,0} + w_{22,1}h_{11} + w_{22,2}h_{12}\,.$$

Finally the output layer is given by

$$\phi = w_{\phi 0} + w_{\phi 1}h_{21} + w_{\phi 2}h_{22}\,.$$

For practice, using the above equations, you should try to write out the following 3-2-2-1 ANN model estimated from the Boston Housing data set:

Inter-layer connections weights

	Input Layer			
Hidden Layer # 1	**RM**	**AGE**	**DIS**	**Bias Node**
Node # 1	-1.39398404	-0.27329631	-0.18890335	-0.24997849
Node # 2	-1.67793629	0.952938014	-0.12204585	0.480725851

	Hidden Layer # 1		
Hidden Layer # 2	**Node # 1**	**Node # 2**	**Bias Node**
Node # 1	1.139652823	1.799861432	-1.04346317
Node # 2	0.289661956	-0.36803742	-0.56943396

	Hidden Layer # 2		
Output Layer	**Node # 1**	**Node # 2**	**Bias Node**
Output Node	-1.61382135	-0.0082267	0.147752257

ANN Classification Models

The output layer of classification ANN models has as many output nodes as there are classification levels. XLMINER only handles binary classification problems - one output node for the "success" (1) and one output node for the "failure" (0). Below we report a 3-2-1 Classification Model based on the Boston Housing data and using the binary classification variable CAT.MEDV. The squashing function for going from the input layer to the output layer is the logistic function while the squashing function for the output layers is also the logistic function. Notice below the output layer weights for the two classes are essentially equal in magnitude but opposite in sign which guarantees that Pr(output = 1) = 1 – Pr(output = 0) as one would desire of binary probability outcomes.

Inter-layer connections weights

	Input Layer			
Hidden Layer # 1	**RM**	**AGE**	**DIS**	**Bias Node**
Node # 1	-3.45681	0.662791	0.598812	2.78478
Node # 2	-3.21809	1.20055	0.138475	3.0873

	Hidden Layer # 1		
Output Layer	**Node # 1**	**Node # 2**	**Bias Node**
1	-3.31916	-3.41982	2.00815
0	3.3328	3.38932	-2.00153

This ANN Classification model is written mathematically as follows:

The hidden nodes in the single hidden layer are

$$h_1 = \frac{1}{1+e^{-z_1}} \quad \text{and} \quad h_2 = \frac{1}{1+e^{-z_2}}$$

where

$$\begin{aligned} z_1 &= w_{10} + w_{11}x_1 + w_{12}x_2 + w_{13}x_3 \\ &= 2.78478 - 3.45681\text{RM} + 0.662791\text{AGE} + 0.598812\text{DIS} \end{aligned}$$

$$\begin{aligned} z_2 &= w_{20} + w_{21}x_1 + w_{22}x_2 + w_{23}x_3 \\ &= 3.0873 - 3.21809\text{RM} + 1.20055\text{AGE} + 0.138475\text{DIS}. \end{aligned}$$

Given these hidden node values we can get the binary output nodes using the following logistic squashing function resulting in

$$\phi_1 = \frac{1}{1+e^{-u_1}}$$

and

$$\phi_0 = \frac{1}{1+e^{-u_0}}$$

where

$$\begin{aligned} u_1 &= w_{\phi=1,0} + w_{\phi=1,1}h_1 + w_{\phi=1,2}h_2 \\ &= 2.00815 - 3.31916\,h_1 - 3.41982\,h_2 \end{aligned}$$

and

$$\begin{aligned} u_0 &= w_{\phi=0,0} + w_{\phi=0,1}h_1 + w_{\phi=0,2}h_2 \\ &= -2.00153 + 3.3328\, h_1 + 3.38932\, h_2 \end{aligned}$$

Then to score this model in the sense of obtaining a confusion table one has to chose a cutoff probability for the "success" class (=1). Obviously the cutoff probability is a tuning parameter in the ANN Classification Model. That is, the confusion tables for the ANN Classification models are dependent on the choice of cutoff probability.

Normalization of Input Data and the Back Propagation Method

It is often recommended that the inputs to an ANN be normalized before training. By normalization we mean the following. Let X denote one of the inputs to the ANN. The normalized value of this input, X^*, is defined as follows:

$$X^* = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad ,$$

where the minimum and maximum values of X are represented by $X_{\min}$ and $X_{\max}$, respectively. This normalization converts the original X value to a normalized value X^* that resides in the [0,1} interval. It in turn helps the back propagation method better determine the weights of the ANN.

One of the most popular methods for determining the weights of ANN models is the so-called **back propagation method**. As the title implies the errors of the ANN are calculated from the output layer back through the hidden layers of the model. Given an ANN structure, the back propagation method starts out with random draws on the weights near zero. Then the initial observation of the training data set is run through the network and, given a prediction problem, the error is determined as err = $(y - \hat{y})$ where y is the first training value of the output variable and $\hat{y}$ is the ANN predicted value using the initially drawn weights. Using this error, the connection and bias weights are "updated" by a fraction of the output error. Given the updated weights, the second observation of the training data set is feed through the network and an error is again determined given the second realized value of the output variable. This error is then used to update the weights again with each likewise iteration through the training data set leading to, in general, a sequence of smaller and smaller errors and, thus, smaller and smaller revisions of the weights until one more training observation leads to a minimal revision in the weights. At this point the final weights of the ANN are determined and the back propagation process stops. Then the resulting ANN model can be used to score additional data sets for validation and testing purposes.

Avoiding Over-Training of ANN Models – Picking the Right Architecture

Of course, the training data set fit can be continually improved by making the structure of the ANN more and more complex to the point of fitting the training data set perfectly. But this would result in the fitting of not only the signal in the data but the noise as well. This is, of course, called over-training the ANN model. Unfortunately, such over-trained models are quite likely to perform poorly on an independent data set. One way to prevent the over-training of an ANN is to try several different architectures of increasing complexity and then choose the architecture that provides the best accuracy when scoring the validation and test data sets. That is, the training data set is used to determine the weights of the competing ANN architectures vis-à-vis back propagation while the validation data set is used to determine the "winning" architecture, the winning architecture producing the best validation data set scores.