An Introduction to Adaptive Logic Networks With an Application to Audit Risk Assessment

Kenneth O. Cogger
Professor Emeritus, University of Kansas
Peak Consulting, Inc.
6650 E. Alameda Ave.
Denver, CO 80224
cogger@compuserve.com

Kurt Fanning
College of Business & Economics
Central Missouri State University
Warrensburg, MO 64093-5070
fanning@cmsuvmb.cmsu.edu

Overview

An *Adaptive Logic Network* (ALN) is a new analytical tool introduced in early 1996. Since then, ALN's have been applied to a wide variety of problem areas, including economic analysis, financial decision making, robotic control, audit risk assessment, sales forecasting, pattern classification, exploratory data analysis, machine maintenance, rehabilitation of spinal cord injury patients, machine learning, automated ECG interpretation, bankruptcy prediction, and subatomic particle recognition. See the reference section for an extensive list of publications.

It is useful to think of ALN's as being both (1) a powerful new type of neural network from the field of artificial intelligence, and (2) a significant generalization of the general linear model found in classical statistics. The problem areas mentioned above have, in fact, all been studied with traditional neural networks, the general linear statistical model, or both. With this interpretation, an ALN is an analytical tool combining and expanding the power of neural networks and classical statistics.

As a type of neural network, ALN's offer distinct advantages over past approaches. First, it is far more efficient from a computational viewpoint. Second, the results are far easier to interpret. Third, ALN's offer the potential for examining the properties of the network for compliance with any desired specification, such as in safety-critical applications; this is not possible with previous neural network models.

As a generalization of the classical linear statistical model, the understanding of an ALN is enhanced by the applicability of such standard tools as hypothesis tests and confidence intervals on critical parameters, which is problematical for standard neural networks. With ALN's, we may formally test whether parts of a network may be removed without losing significant explanatory power. We may also eliminate inputs which are statistically unimportant. A final important aspect of an ALN is its ability to incorporate apriori restrictions on the nature of the fitted model. For example, monotonicity or convexity constraints are easily imbedded.

In the remainder of this introduction, we shall describe the basic structure of an ALN, show how it relates to neural networks and the linear statistical model, illustrate the unique parameter estimation scheme for ALN's, and work through an application to a real data set.

The Basic Elements of an ALN

The *Adaptive* term in an ALN refers to the changeable, or adaptive, parameters in an arbitrary number of linear functions. These linear functions are of the form

$$L_j = \sum_{i=0}^{n} w_{ij} X_i - Y$$

An ALN adapts, or changes, the weights (w_{ij}) in each of these linear functions to produce desired results. Usually, to allow model flexibility, we require a constant term in each function therefore $X_0 \equiv 1$.

In the context of the linear statistical model, the X's are the *independent* or *explanatory* variables, while Y is the *dependent* variable. In the context of a typical neural network model, the X's are the input variables while Y is the network output. The restriction on X_0 is often referred to as the inclusion of a *bias* term in the neural network.

If $L_j = 0$, we have the definition of a straight line (if n=1), plane (n=2), or hyperplane (n>2). Thus, at such a zero threshold, we have the equation

$$Y = \sum_{i=0}^{n} w_{ij} X_i$$

An ALN incorporates these linear functions into the *leaf nodes* of a *decision tree*. Each leaf of the tree evaluates whether $L_j \geq 0$, resulting in a value of True(1) or False(0). Thus an evaluation of True(1) means that a linear threshold has not been reached and therefore

$$Y \leq \sum_{i=0}^{n} w_{ij} X_i$$

for that particular combination of values for the input and output variables. Thus a central feature of an ALN is the inclusion of *linear threshold units* (ltu's) at the leaf node level.

The *logic* term in ALN refers to the *parent* nodes of the ltu's, which are restricted to be either *And* or *Or* logic functions. All other parent nodes in the decision tree have the same restriction. Thus the ltu's are the children of these logic functions as are all other nodes in the

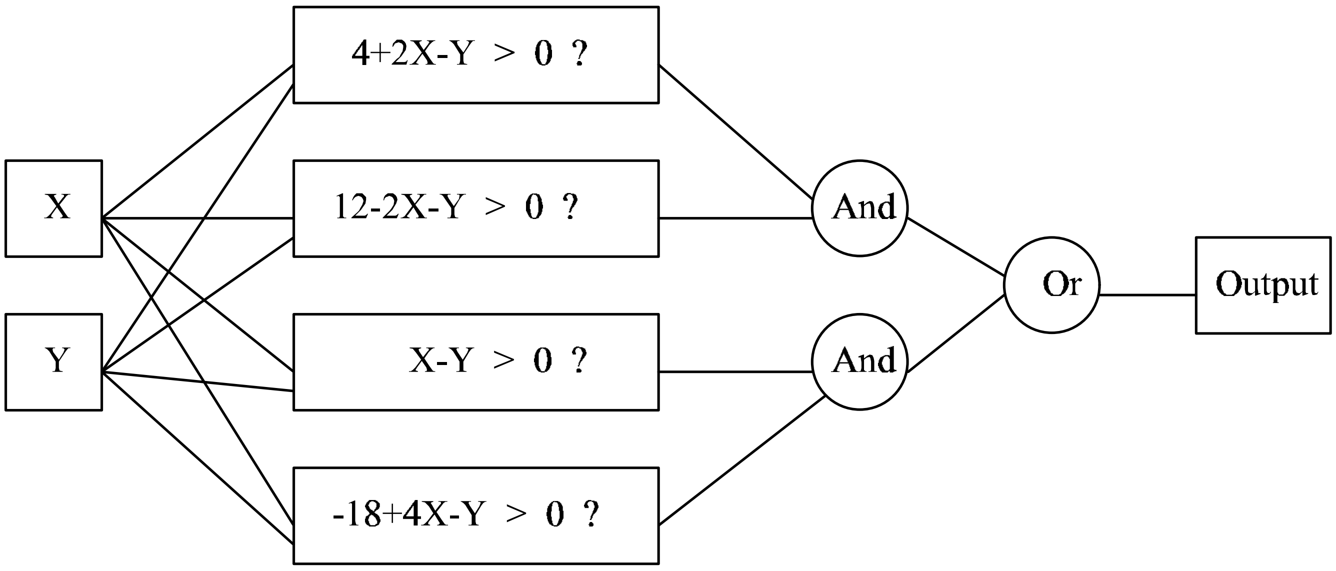

Figure 1
Three Layer ALN

decision tree. Figure 1 portrays a simple ALN with a particular set of ltu's.

This figure describes the structure of a three-layer ALN. Our convention in counting layers is to count only the layers incorporating linear threshold and logic functions. There is also a convenient shorthand notation to describe ALN decision trees. In this notation, the tree in Figure 1 may be described as Or(And(2),And(2)). The reader interested in an exercise may wish to construct the tree described by And(ltu,Or(And(2),Or(2)).

As mentioned previously, ALN's are closely related to the linear statistical model. To see this, note that an ALN consisting of just a single linear threshold unit would, *at the zero threshold boundary*, define a multiple regression equation. The general linear statistical model is therefore a special case of an ALN.

The number of possible ALN structures is boundless, but fortunately in practice only a reasonable number need be considered. This is true for at least two reasons. First, the size of the data set which is used to estimate parameters dictates a limit on the number of linear pieces employed in an ALN. If there are n independent (X) variables, each linear piece will have n+1 variable weights, and clearly we should not have more weights to be estimated than there are observations in our estimation sample. Second, many ALN structures which on the surface appear different are really equivalent. For example, And(ltu,And(2)) describes the same tree as And(3). These and other considerations effectively reduce the number of trees that might be explored in any given situation.

Appendix A describes all unique trees with four or fewer ltu's. There are only 17 such trees. In practice, we have found that the descriptive ability of many trees is nearly equivalent, so the large *potential* number of trees is not a significant problem.

Modeling Power of ALN's

We stated in the overview that ALN's represent a significant advance over traditional linear statistical models (the general linear model is the simplest possible ALN), while retaining their ease of interpretation. We also suggested that ALN's represent a new, more computationally efficient, form of a neural network. Let us see why this is so. Consider again the fairly simple ALN of Figure 1. Figure 2 portrays the two-dimensional output space for this network; the shaded area corresponds to all points (X,Y) where the output of the network is True(1).

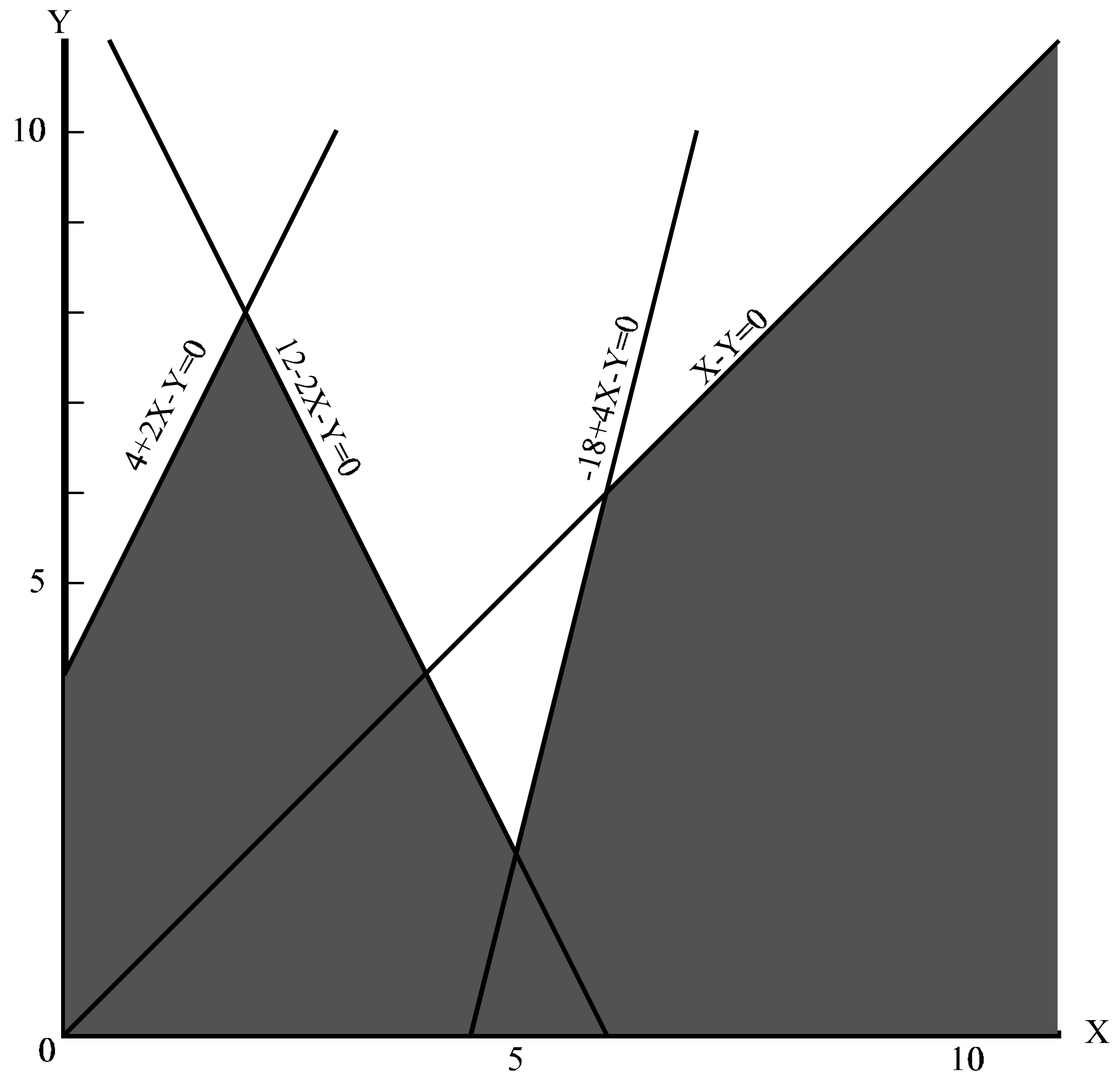

Figure 2
Output-Space Diagram
Or(And(2),And(2))

The first And evaluates as True only for points below two lines, giving a "pup-tent" shaped shaded region corresponding to the minimum of two linear functions. The second And gives half of a "cabin-tent" shaped region. The Or evaluates as True for points in either of these two regions, thus corresponding to the maximum of these two tent boundaries. This points out an important insight into the use of logic functions with linear threshold units: And equates to Minimum and Or equates to Maximum. The net result is that the boundary between 0-space and 1-space for the output of an ALN is a piecewise linear function. In higher dimensions, ALN's are joining hyperplanes. In our simple example, there are four pieces, the function is continuous(as are all ALN's), but is neither monotone, convex, or concave.

Such a piecewise linear function is easily interpreted, for within the domain of each piece, changes in the output are directly proportional to changes in each input, and the coefficients in each linear piece represent partial derivatives. In a standard neural network, the partial derivative of the output with respect to any input variable is impossible to compute unless all other variables are assigned fixed values, and even a slight change in any of those fixed values may dramatically alter the desired partial derivative. Thus ALN's offer an ease of interpretation not possible with standard neural networks. Piecewise linear functions are inherently easier to interpret.

Since any continuous function can be approximated arbitrarily closely by a piecewise linear function, an ALN with a sufficient number of linear threshold units can approximate arbitrary continuous functions. Those familiar with the Kolmogorov Representation Theorem for standard neural networks will appreciate the simplicity of this (informal) parallel proof for ALN's.

There is yet another advantage to the ALN approach. While the piecewise linear response of the ALN in Figure 1 turned out *not* to be monotone, convex, or concave, these properties could be imposed easily by modifying the type of tree and / or constraining coefficients in the ltu's. Using the same number of ltu's, with no restrictions on the weights, a convex surface is produced with And(4), and a concave surface with Or(4). With currently available software, it is also possible to restrict part of the output space to be convex and part to be concave. This ability is desired when modeling "S-shaped" curves such as sales growth over time. Monotonicity can also be imposed by restricting coefficients within ltu's to be positive or negative. The estimation algorithm for ALN's permits such restrictions to be easily imposed.

Estimation or Learning in ALN's

Now that we understand the basic elements of an ALN, we need to examine how appropriate parameter values can be determined for any particular application. In statistics, this process is called *parameter estimation*; in neural networks, it is called *supervised learning*. What is required is a sample set of observed X and Y variables which have representative values. These values might come from carefully planned experiments, unplanned but nevertheless regular observations (such as from economic data), or from fixed patterns such as alphabetic characters corresponding to (encoded) handwritten versions of those characters. We shall call this the estimation or training sample.

In statistics, the most widely used estimation scheme adopts the principle of least squares. There, parameter values are chosen which minimize, within the estimation sample, the total squared difference between the observed and estimated Y values. An ALN also uses least squares as its estimation criteria, but in a sequential fashion. We will describe it using a simple numerical example.

First, parameter values are randomly assigned to all linear functions in the ALN (Other assignments are possible if apriori knowledge exists) . Next, sequentially, training values (X_t, Y_t) in the estimation sample are input to the ALN. Truth values are then propagated through the tree, producing the tree output. An example is given in Figure 3, which portrays the initial tree from Figures 1 and 2 being presented with the training pair (X_t, Y_t) = (3,2).

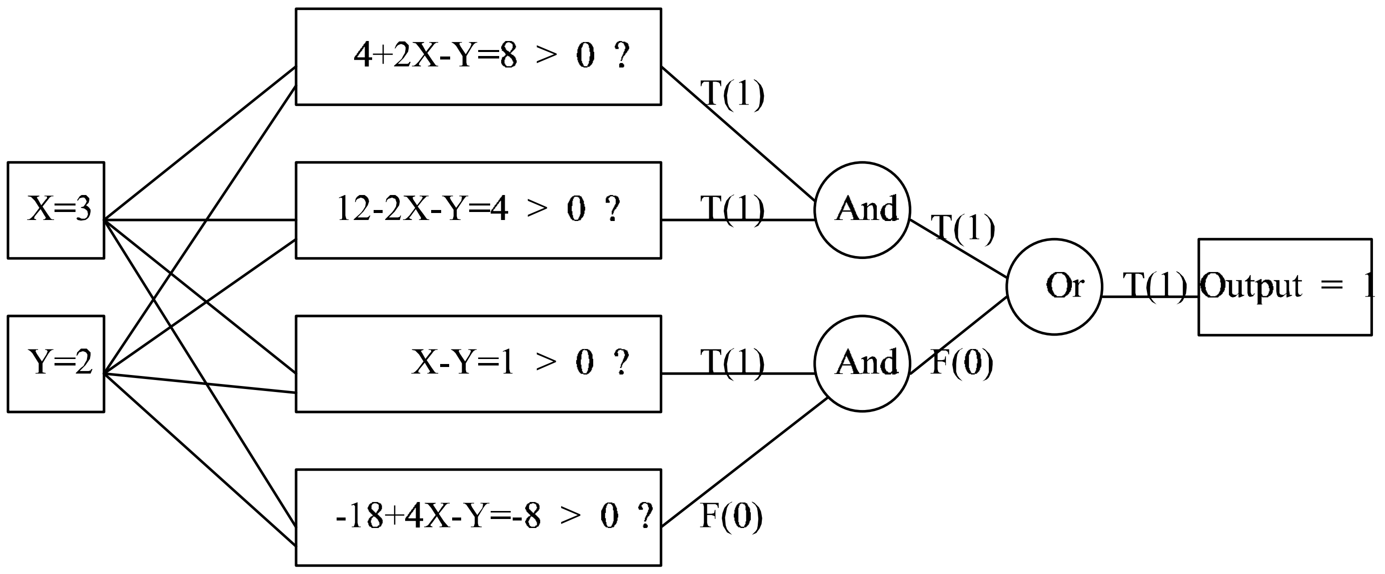

Figure 3
Or(And(2),And(2))
Evaluated at (X,Y)=(3,2)

For $X_t = 3$, it is seen *visually* in Figure 2 that the active surface is defined by $Y = 12 - 2X_t = 6$. Since $Y_t = 2 \neq Y$, iterative least squares is used to modify the active surface Y, moving it closer to the target, or training,value.

Assuming a learning rate (this is identical to the “smoothing constant” used in exponential smoothing; as the smoothing constant approaches zero, “batch” and “iterative” least squares produce the same solution) of α = 0.2, the necessary formulas are:

$$E = 0.5(Y - Y_t)^2$$

$$\frac{\P E}{\P w_0} = (Y - Y_t)\frac{\P Y}{\P w_0} = (Y - Y_t) = 4$$

$$\frac{\P E}{\P w_1} = (Y - Y_t)\frac{\P Y}{\P w_1} = (Y - Y_t)X_t = 12$$

$$\Delta w_0 = -\alpha \frac{\partial E}{\partial w_0} = -0.8 \quad \therefore w_0(new) = w_0(old) - 0.8 = 12 - 0.8 = 11.2$$

$$\Delta w_1 = -\alpha \frac{\partial E}{\partial w_1} = -2.4 \quad \therefore w_1(new) = w_1(old) - 2.4 = -2.0 - 2.4 = -4.4$$

and the active surface changes to $Y = 11.2 - 44X$, moving it incrementally toward the current target output. The next (X_t, Y_t) training pair is then presented to the network and the above steps are repeated. After passing through the entire training sample, a training *epoch* is said to be completed. Learning typically consists of passing throught the training sample for many epochs.

It is often recommended to utilize a series of such training runs, at decreasing learning rates a . For example, a training run with a coarse learning rate (e.g. a = 0.5) for 500 epochs might be followed by 300 epochs with a learning rate ofa = 0.1. The rationale is that a coarse learning rate allows widely varying linear pieces to be tested; once a reasonable set of pieces has been coarsely estimated, smaller learning rates fine tune the fit. It may be shown that the limiting behavior of this approach is equivalent to batch least squares.

Note that the ALN training process modifies only the weights on a single linear piece at each iteration. This is the key to the speed of ALN learning. With standard backpropagation networks, usually every parameter in a network is responsible for any output error and learning computations are comparatively slow because every weight in the network must be modified at each iteration. Readers unfamiliar with backpropagation networks may refer to Appendix B for a brief summary of this type of neural network and a simple example of how many parameters must be changed at each iteration.

Of course, our preceding example used a visual inspection of Figure 2 to determine which linear piece was active and *responsible* for the output error. In practice, such a determination must be made with a computer algorithm. Fortunately, this is trivially easy to program given the structure of an ALN. First, every logic node in an ALN can be assigned formal responsibility using the rules of logic. The assignment of responsibility proceeds from the output node backwards. First, by definition, the output node is always responsible. Second, the upper And node (see Figure 3) is responsible since a change in its truth value (from 1 to 0) would change the output of the network. The lower And node is not responsible, since a change in its truth value (from 0 to 1) would not alter the output of the network.

This eliminates the entire bottom half of the network from potential responsibility. No change, however large, in the weights of the bottom two linear threshold functions would change the output of the network. Thus responsibility is narrowed to the two linear threshold functions that are inputs to the upper And node. Since both are True(1), a change in either one would change the output of this And node. Thus both linear pieces are formally responsible. To determine which is the active surface, since this is an And(minimum) node, the minimum threshold exceedance (4) indicates that 12-2X-Y=0 is the active surface.

This algorithm is quite general and efficient. Formal rules for responsibility of logic nodes with binary inputs are used to greatly reduce the number of nodes that must be evaluated, and the Minimum(Maximum) dualities of And(Or) nodes eliminate all but the one active linear piece.

Another less efficient algorithm for determining the active piece does not rely upon responsibility of logic nodes, but is easier to describe. For any training vector, compute the value of each linear piece. For our ALN, this produces the values 8,4,1,-8 (reading from top to bottom). Take the minimum(And) of (8,4) = 4, the minimum of (1,-8) = –8, then take the maximum(Or) of (4,-8) = 4. Keeping track of which piece (12-2X-Y) produced this value, the active surface is again identified.

In such an iterative estimation scheme, it is easy to incorporate apriori constraints on the parameters in the weight vectors. Such constraints allow enforcement of monotonicity conditions desired by the user.

Testing ALN’s With Standard Regression Tools

Statistical testing for most neural networks is problematical given the complicated nature of the input-output relationship. Such is not the case with ALN’s given their piecewise linear nature. Standard regression software packages may easily be used to refine the weight estimates of an ALN and even test for the appropriate number of linear pieces. Below, we demonstrate this for a simple ALN. The general approach is in Cogger and Armstrong(1997).

Suppose an ALN of the form And(ltu,Or(2)) has been determined based on a training set of data. Then the output of this ALN will be given by

$$Y = Min(L_1, Max(L_2, L_3)),$$

where each of the linear functions utilize weights on the independent variables determined by the estimation process described in the previous section. For any particular value of the independent variable(s), the responsible linear piece may be determined. To identify the responsible linear piece, define two indicator variables

$$z_1 = \begin{cases} 1: L_1 = Min(L_1, Max(L_2, L_3)) \\ 0: otherwise \end{cases} ;\ z_2 = \begin{cases} 1: L_2 = Min(L_1, Max(L_2, L_3)) \\ 0: otherwise \end{cases}$$

Then it follows that in the training sample, we have the identified model

$$Y = L_3 + (L_1 - L_3)z_1 + (L_2 - L_3)z_2$$

The contrasts, $(L_1 - L_3)$ and $(L_2 - L_3)$, between linear pieces in this expression have an interesting interpretation. In the case of a single independent variable X, a contrast will be of the form a(X-b), where a is some constant and b is the abscissa or *joint point* X=b where the linear pieces intersect. Thus the above expression is a generalization of the usual indicator variable approach to defining a piecewise linear model. In our model, there are two join points, meaning we have three separate linear pieces.

This model may now be generalized for estimation in a holdout sample to

$$Y = \sum_{i=0}^{n} w_i X_i + \mathrm{b}_1(L_1 - L_3)z_1 + \mathrm{b}_2(L_2 - L_3)z_2$$

Since this model is linear in all parameters, standard regression programs may be used to estimate the w and b parameters. The usual T and F statistics may be used to test hypotheses of interest. For example,

$H_0 : \mathrm{b}_i = 0$

may be tested with a standard T statistic. If this hypothesis is rejected, the corresponding contrast between two linear pieces is significantly different from zero, meaning that at least two linear pieces are required to describe the statistical relationship. If accepted, the corresponding contrast may be removed from the model, reducing the number of linear pieces by one. This would lead to a simplified, pruned, tree.

Also, the hypothesis

$H_0 : \mathrm{b}_1 = \mathrm{b}_2 = 0$

may be tested using the usual conditional error sum of squares procedure (see Draper and Smith(1991)). This uses an F statistic. If the hypothesis is accepted, one would conclude that both contrasts can be removed from the model, meaning that a simple multiple regression (one linear piece) is an adequate description of the data, as opposed to the piecewise linear nature of an ALN.

Finally, the overall F statistic for the complete model may be used to test whether *all* variable coefficients $(\mathrm{b}_1, \mathrm{b}_2$, and all w_i) in the model are zero. If this hypothesis is accepted, then the response variable Y is simply random variation; no linear or even piecewise linear association exists between Y and the X variables.

Thus standard regression analyses may be performed on an ALN, trimming branches off the decision tree and simplifying the model much like stepwise regression. ALN's are the first neural network model to permit such easy application of standard well-understood statistical tools.

A Sample Application in Audit Risk Assessment

We have conducted a preliminary study of the usefulness of ALN's in the detection of fraudulent management behavior in audit situations. Details of this study are available on request, and space limitations prevent a more complete discussion. Here, we only wish to describe what variables we employed, what type of ALN seemed most appropriate, the ease or lack of ease in developing the ALN, predictive accuracy results for classifying firms with known classifications, and some software materials we have developed which may be of help to others wishing to explore ALN's in similar application areas.

Information was gathered on 21 variables for each of 204 firms. A listing of these variables is given in Appendix C. There were an equal number of firms classified as Fraud or Nonfraud based on available court and administrative records, and they were paired based on industry codes. 150 paired firms were assigned to a training sample and the remaining 54 paired firms were assigned to a validation or holdout sample.

We developed an Excel Visual Basic for Applications(VBA) macro to assist us in rapidly examining all possible ALN structures containing four or fewer linear pieces. More pieces would clearly lead to overfitting and poor generalization in our holdout sample since each piece has 21 parameters (the weights on the 20 predictor variables plus a constant term) and these parameters are being estimated with only 150 training cases. An ALN with four linear pieces would be estimating 84 parameters. The VBA macro uses Atree 3.0, a commercial program published by Dendronic Decisions, Ltd. for the estimation of ALN's to efficiently explore all of the possible tree structures. The macro permitted us to complete this exploration in a few hours, a great saving of time over manual approaches.

Appendix D lists sources for software used in this study, some of which is available free of charge. Appendix E lists abbreviated input and output files from this study so the reader can appreciate the relative simplicity of tasks necessary in the study of ALN's.

The VBA macro (1) requests input from the user as to what tree structure to try (e.g. and(or(2),and(2))), (2) executes Atree3.0 on the training data, (3)creates a worksheet with predicted scores for Fraud for both training and holdout samples, (4)optimizes a cutoff score for the maximum number of correct classifications in the training sample, and then(5)applies this cutoff score to the holdout sample. This procedure assures that the results in the holdout sample are not biased in any way, since all optimization is restricted to the training sample.

We found that several ALN structures yielded nearly equivalent results in the training sample, although the best one appeared to be an ALN structure of the form Or(Or(2),And(2)). This yielded 132 of 150 (=88%) firms correctly classified in the training sample with an optimum cutoff score of 0.507. Breaking this down, 67 of 75 (=89%) Fraud firms were correctly classified, and 65 of 75 (=87%) NonFraud firms were correctly classified. Naturally, these levels of accuracy are unreliable given that parameter values and the cutoff score are optimized precisely for that sample. Results are in the table below.

ALN in Estimation Sample

	Predicted		
Actual	**Fraud**	**NonFraud**	**Total**
Fraud	67	8	75
NonFraud	10	65	75
Total	77	73	150

When the indicated cutoff score and the estimated ALN are applied to the holdout sample, we found 32 of 54 (=59%) firms correctly classified. This reduction in classification accuracy is expected given the bias mentioned above. Breaking this down, 18 of 27 (=67%) of the Fraud firms were correctly classified and 14 of 27 (=52%) of the NonFraud firms were correctly classified. Results are tabled below.

ALN in Holdout Sample

	Predicted		
Actual	**Fraud**	**NonFraud**	**Total**
Fraud	18	9	27
NonFraud	13	14	27
Total	31	23	54

These preliminary results are especially encouraging given our experience with this same data set using traditional backpropagation networks, linear and quadratic discriminant analysis, multiple regression, and other analytical tools; usually, these techniques have difficulty achieving better than 50% total classification accuracy, which is no more than a random assignment would achieve in our sample. Of special note is the fact that the ALN actually performed better on Fraud firms than NonFraud firms in both training and holdout samples. Given the disparate Type I and Type II classification error costs in an audit setting, this is especially encouraging.

We note also that all variables used in this study are externally available public information. Information available to auditors in a real audit situation was not employed. If such information were incorporated, we would fully expect much more accurate classification rates for both fraud and nonfraud firms. We are also exploring the use of other discriminant variables and expanding the firms to include more recently acquired data.

Appendices

A. ALN Tree Structures

Below are listed all unique ALN decision trees with four or fewer linear pieces. If anyone wishes to extend this list to more pieces, the author would appreciate receiving a copy of the extended list for posting on the web. The list below may be found at stat1.cc.ukans.edu/~cogger/structures.pdf

Possible Tree Structures with up to Four Pieces

Tree	Notation	Pieces	Atree 3.0 Description	Equivalence
A	A	1	Ltu	-
B	&(2A)	2	And(2)	-
C	V(2A)	2	Or(2)	-
D	&(3A)	3	And(3)	-
E	V(3A)	3	Or(3)	-
F	&(A+B)	3	And(Ltu,And(2))	D
G	V(A+B)	3	Or(Ltu,And(2))	-
H	&(A+C)	3	And(Ltu,Or(2))	-
I	V(A+C)	3	Or(Ltu,Or(2))	E
J	&(4A)	4	And(4)	-
K	V(4A)	4	Or(4)	-
L	&(2A+B)	4	And(Ltu,Ltu,And(2))	J
M	V(2A+B)	4	Or(Ltu,Ltu,And(2))	-
N	&(2A+C)	4	And(Ltu,Ltu,Or(2))	-
O	V(2A+C)	4	Or(Ltu,Ltu,Or(2))	K
P	&(A+D)	4	And(Ltu,And(3))	J
Q	V(A+D)	4	Or(Ltu,And(3))	-
R	&(A+E)	4	And(Ltu,Or(3))	-
S	V(A+E)	4	Or(Ltu,Or(3))	K
T	&(A+G)	4	And(Ltu,Or(Ltu,And(2)))	-
U	V(A+G)	4	Or(Ltu,Or(Ltu,And(2)))	M
V	&(A+H)	4	And(Ltu,And(Ltu,Or(2)))	N
W	V(A+H)	4	Or(Ltu,And(Ltu,Or(2)))	-
X	&(B+C)	4	And(And(2),Or(2))	N
Y	V(B+C)	4	Or(And(2),Or(2))	M
Z	&(2B)	4	And(And(2),And(2))	J
AA	V(2B)	4	Or(And(2),And(2))	-
AB	&(2C)	4	And(Or(2),Or(2))	-
AC	V(2C)	4	Or(Or(2),Or(2))	K

There are 17 unique decision trees. Some duplicates are purposely listed (e.g. D and F) for instruction. In the above table, once a particular tree has been found equivalent to another, that tree is never again considered. For example, &(A+F) is not listed since it is equivalent to &(A+D).

The second column in the table introduces a notation which may be helpful in extending this table to 5 or more pieces.

B. Traditional Backpropagation Neural Network

One common neural network is the *feedforward* neural network, in which the output of any node in the network is passed in only one direction, namely, to the output node. *Backpropagation* refers to the learning or estimation scheme the network employs to determine appropriate weights or parameter values in the functions which are computed by each node of the network. This estimation scheme uses iterative least squares, like an ALN, except that when calculating

$$\frac{\P E}{\P w_0} = (Y - Y_t)\frac{\partial Y}{\P w_0} = (Y - Y_t) = 4$$

$$\frac{\P E}{\P w_1} = (Y - Y_t)\frac{\P Y}{\P w_1} = (Y - Y_t)X_t = 12$$

the chain rule of calculus must be used to determine these derivatives for every parameter in the network, since a typical feedforward network is fully connected; every node function is usually responsible for the error E.

The node functions for a backpropagation network must be continuously differentiable. Common choices for the input and output functions of a node are

$$Input = \sum_i w_i X_i$$

$$Output = [1 + \exp(-Input)]^{-1}$$

Which is a *sigmoid* function with an S-shaped curve. This function is convenient for the calculation of derivatives because

$$\frac{dOutput}{dInput} = Output \cdot [1 - Output]$$

and this expression uses values which have been previously computed.

Figure 4 portrays a fully connected four-layer neural network. Here, each node is fully connected to all nodes in the subsequent layer. In addition to the input and output layers, there are two intermediate, or *hidden*, layers. As can be imagined, if each node is of the sigmoidal type described, the output of such a network is a highly nonlinear function of the inputs and is often difficult to interpret.

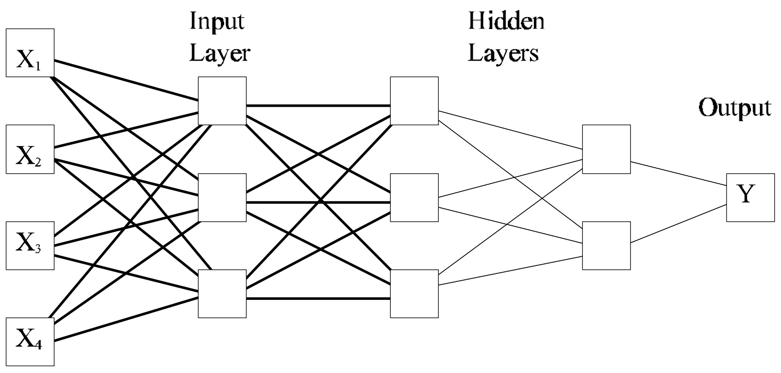

Figure 4
Typical Feedforward
Neural Network

One practical difficulty with such a network architecture is that there are 38 parameter values to be estimated. For each training case, the error derivative must be computed for each parameter and changes must be made in each parameter value. By contrast, only a single linear piece in an ALN would be modified at each stage, meaning that an ALN comparable to Figure 4 would require changes in only 5 parameters at each stage. Instead of many derivatives being calculated and backpropagated, only the responsible piece need be determined, and this is easily accomplished by a simple algorithm.

C. Variables in Audit Risk Study of ALN's

Variable	**Definition**
1	Fraud?(0-1)
2	Size of Board of Directors
3	% Outside Directors
4	CEO=Chair?(0-1)
5	Audit Committee?(0-1)
6	Compensation Committee?(0-1)
7	Non-Big6 Audit Firm?(0-1)
8	Profit Sharing Plan?(0-1)
9	Altman's Z-Score
10	3-year Annualized Sales Growth Rate
11	Change in CEO Last 3 Years?(0-1)
12	CEO=CFO?(0-1)
13	Litigation in Last 3 Years?(0-1)
14	LIFO Inventory Valuation?(0-1)
15	Accounts Receivable / Sales
16	Inventory / Sales
17	Net Plant,Property,Equipment / Total Assets
18	Debt / Equity
19	Sales / Total Assets
20	Accounts Receivable Growth > 10% in Last Year?(0-1)
21	Gross Margin Growth > 10% in Last Year?(0-1)

D. Software for Estimation of ALN's

The only available commercial software to implement Adaptive Logic Networks is Atree 3.0, published by

Dendronic Decisions, Ltd.
800 Tower One, Scotia Place
10060 Jasper Ave.
Edmonton, AB T5J 3R8
CANADA
(403) 421-0800
(403) 421-0850 (Fax)

An educational version of Atree3.0, as well as other information about ALN's, may be downloaded from
www.cs.ualberta.ca/~arms

The VBA macro for Excel97, which automates many of the tasks associated with using Atree3.0 in a discriminant analysis setting, is available at no charge from one of the authors. Those familiar with VBA can easily modify this macro for more general applications involving Atree 3.0. Please address requests to cogger@compuserve.com.

E. Representative Atree 3.0 Files

Atree 3.0 allows many analyses to be performed on Adaptive Logic Networks. For example, an estimated ALN may be applied to different data sets than the estimation sample, two and three-dimensional plots may be produced, etc.. Below are partial listings of the input files required by Atree 3.0 in our fraud risk study, as well as one output file. All are standard text files.

Fraud.adl

```
ALN aFraud(21, 1)  //Defines number of variables; variable 1 is the output or dependent variable
{
var 1 // Indicator variable for fraud(1) vs nonfraud(0)
 {epsilon = 0.25;} //prevents overfitting with linear pieces that are unnecessary
var 2 // Size of board of directors
 {wmax = 0;} //apriori restriction on this variables coefficient
var 3  //Percentage outside directors
 {wmax = 0;}
 var 4  //1 if CEO is also board chair
 {wmin = 0;}

…..(other variables deleted to abbreviate listing)……

var 21   //1 if gross margin growth > 1.10 past growth
 {wmin = 0;}

tree=and(ltu,or(2)); //Tree to be estimated.
};
```

Fraud.shl

```
LoadADL "c:\atree30\samples\fraud\fraud.adl"

// Train twice, beginning at course (0.5) learning rate, then finetune (0.1)
// Stop after 500 cycles through the estimation set or when RMSE<0.4 (0.3)
// Fraud estimation data is in fraudest.dat
aFraud.Train(500, 0.4, 0.5, "c:\atree30\samples\fraud\fraudest.dat")
aFraud.Train(500, 0.3, 0.1, "c:\atree30\samples\fraud\fraudest.dat")

// create a decision tree on var 1, and save the tree and all parameters
// maximum layers for the DTREE is 3

DTREE dFraud(aFraud, 1, 3)
dFraud.Write("c:\atree30\samples\fraud\fraud.dtr")
//Calculate aln output for all cases in holdout sample
dFraud.Eval("c:\atree30\samples\fraud\fraudhld.dat","c:\atree30\samples\fraud\fraudhld.out")
//Calculate aln output for all cases in estimation sample
dFraud.Eval("c:\atree30\samples\fraud\fraudest.dat","c:\atree30\samples\fraud\fraudest.out")
```

fraudest.dat (abbreviated listing)

```
1   14      64      1       ……
0   12      75      0       ……
1   6       50      1       ……
0   7       71      1       ……
.   .       .       .       .
.   .       .       .       .
```

fraudhld.dat (abbreviated listing)

```
1   7       29      1       ……
0   9       33      1       ……
1   4       0       0       ……
0   5       40      1       ……
.   .       .       .       .
.   .       .       .       .
```

fraud.dtr (abbreviated Atree 3.0 output file describing the estimated aln)

```
// c:\atree30\samples\fraud\fraud.dtr exported on Tue Jun 24 15:26:27 1997
// ALN Decision Tree file format v1.0 (C)1994 Dendronic Decisions Limited
VERSION = 1.0;
VARIABLES = 21;
……(summary statistics on all 21 variables in study)……
OUTPUT = x0;
LINEARFORMS = 3;
…..(description of estimated coefficients for each linear piece)…..
BLOCKS = 1;
0 : MIN(0, MAX(1, 2));  (compare to the tree= definition in fraud.adl)
DTREE = 1;
0 : block 0;
```

References

Armstrong, W.W., and Thomas, M.M. (1996), "Adaptive Logic Networks," Section C1.8, *The Handbookof Neural Computation,* Fieseler, E., and Beale, R., eds., Institute of Physics Publishing and Oxford University Press USA.

Armstrong, W.W., Chungying Chu, and Thomas, M.M. (1996), "Feasibility of Using Adaptive LogicNetworks to Predict Compressor Unit Failure," Chapter 12, *Applications of Neural Networks in Environment, Energy, and Health,*Keller, P.E., Hashem, S., Kangas, L.J., and Kouzes, R.T., eds.,World Scientific Publishing Company Ltd., London.

Cogger, K.O., and Armstrong, W.W. (1997), "Identification, Estimation, and Testing of Piecewise Linear Multiple Regression Models Using Adaptive Logic Networks," Working Paper.

Cogger, K.O.,Koch, P.D., and Lander, D.M. (1997), "Adaptive Logic Network, ARIMA, and Regression Forecasts of International Equity Markets During Volatile Conditions," *Advances in Financial Economics***,** vol. 3, JAI Press, 117-157.

Draper, N.R., and Smith, H. (1981), *Applied Regression Analysis,* New York: John Wiley.

Fanning, K., and Cogger, K.O. (1994),"A Comparative Analysis of Artificial Neural Networks Using Financial Distress Prediction," International Journal of Intelligent Systems in Accounting, Finance, and Management, vol. 3, 241-252.

Fanning, K., Cogger, K.O., and Srivastava, R. (1995), "Detection of Management Fraud: A Neural Network Approach," International Journal of Intelligent Systems in Accounting, Finance, and Management, vol. 4, 113-126.

Fanning, K., and Cogger, K.O. (1997), "Neural Network Detection of Management Fraud Using Published Financial Data," International Journal of Intelligent Systems in Accounting, Finance, and Management, vol. 6.

Kostov, A., Andrews, B.J., Popovic, D.B., Stein, R.B., and Armstrong, W.W. (1995), "Machine Learning in Control of Functional Electrical Stimulation Systems for Locomotion," *IEEE Trans. Biomed. Eng.,*vol. 42, no. 6, 541-551.

Kremer, S.C., and Melax, S.(1994), "Using Adaptive Logic Networks for Quick Recognition of

Particles," *IEEE International Conference on Neural Networks,* Orlando, Florida, vol. 5, 3015-3019.

Polak, M.J., Zhou, S.H., Rautaharju, P.M., Armstrong, W.W., and Chaitman, B.R. (1995), "Adaptive Logic Network Compared with Backpropagation Network in Automated Detection of Ischemia from Resting ECG," *Proceedings, Computers in Cardiology Conference,* Vienna, Austria, 217-220.