Journal of International Information Management

Volume 8 | Issue 2 Article 4

1999

An abductive model of population change

Milan Aiken
The University of Mississippi

An abductive model of population change

Milan Aiken
The University of Mississippi

ABSTRACT

Logical abduction is a relatively unknown technique that may be used to classify or forecast. This paper describes abduction and applies it to a classification problem involving population change in the United States to test its accuracy. Results show that for this problem, it is as accurate as logistic regression, used in a prior study. Guidelines for using abduction are also presented.

INTRODUCTION

Although relatively unknown, logical abduction has been used to forecast precipitation (Saburo, 1984), seasonal climate (Leobow, Mehra, & Toldalagi, 1984), fish populations (Brooks & Probert, 1984), shrimp catches (Prager & Saila, 1984), interest rates (Ohashi, 1984; Scott & Hutchinson, 1984), and many other variables (Barron, Mucciardi, Cook, Craig, & Barron, 1984). Few studies have compared abduction with other techniques, however. One study found abduction to be more accurate than a neural network for residential property assessment, bank loan approval, and the "exclusive or" problem and was faster to develop (Aiken, Paolillo, & Vanjani, 1999). A second study found abduction to be superior to a neural network forecasting housing prices in the Boston area (Aiken & Vanjani, 1999), and a third study found it to be superior to logistic regression for the "exclusive or" problem and bank loan approval (Aiken & Alonzo, in press).

The purpose of this paper is to describe abduction and to illustrate its use with a population change classification problem. This problem has been used at least three times before to study different classification techniques including discriminant analysis, logistic regression, and artificial neural networks. Results show that abduction is just as accurate as logistic regression for this problem, but slightly less accurate than a neural network. The paper concludes with guidelines for using abduction.

ABDUCTION

Abduction was first defined as a form of logical inference by Charles Peirce in the 1860s (Fann, 1970), but a mathematical formulation of this theory known as the Group Method of Data Handling (GMDH) was developed about 100 years later (Ivakhnenko, Krotov, & Visotsky, 1979) and is perhaps the most commonly-used implementation of the method (Farlow, 1984).

Using numeric functions to describe complex relationships, abduction differs from deduction and induction in that abduction may be used for problems with a high degree of uncertainty. Like induction, however, abduction learns from examples. By iteratively evaluating a large number of potential models, abduction determines the functional element coefficients, number of network elements, types of network elements, and the connectivity among the elements. Abduction utilizes a network of functions so that only the relationships among small subsets of variables need to be discovered at a time. Using logical abduction, a mathematical model is constructed in a manner similar to that of evolution. Starting with a few basic equations, a new generation of more complex equations and a survival-of-the-fittest principle is used to determine which equations live and which equations die.

Like neural networks, abduction automatically learns the relationships among variables in a model and does not require the user to make assumptions about the underlying distribution or that the input variables are independent (Wasserman, 1989). Multi-linear and logistic regression and discriminant analysis have certain assumptions that must be met; otherwise, their results may be unreliable. Again, like neural network learning, abduction automatically evaluates a large number of potential models and develops a network with nodes containing mathematical functions, more complex than the nodes in a neural network. Unlike a neural network using back propagation, abduction automatically selects the network architecture. Thus, it is similar to that used in neural network applications using the genetic algorithm (Dorsey, Johnson, & Mayer, 1994). Thus, the primary differences between abduction and neural networks are that the former: (1) typically results in a network with fewer, more powerful nodes; (2) often results in faster network development; and (3) automatically determines the network architecture.

The *Abductive Information Modeler (AIM)* from Abtech Corporation is an example of how abduction may be implemented. Using this software, the synthesized abductive network may consist of seven types of elements described below and shown in Figure 1 (AIM User Manual, 1994):

1. **Singles** $w_0+(w_1x_1)+(w_2x_1^2)+(w_3x_1^3)$
2. **Doubles** $w_0+(w_1x_1)+(w_2x_2)+(w_3x_1^2)+(w_4x_2^2)+(w_5x_1x_2)+(w_6x_1^3)+(w_7x_2^3)$
3. **Triples** $w_0+(w_1x_1)+(w_2x_2)+(w_3x_3)+(w_4x_1^2)+(w_5x_2^2)+(w_6x_3^2)+(w_7x_1x_2)$ $+(w_8x_1x_3)+(w_9x_2x_3)+(w_{10}x_1x_2x_3)+(w_{11}x_1^3)+(w_{12}x_2^3)+(w_{13}x_3^3)$
4. **White Elements** A linear weighted sum of all the outputs of the previous layer. $w_1x_1+w_2x_2+w_3x_3+\ldots+w_nx_n$
5. **Normalizers** Normalizers transform all of the original input variables into a relatively common region with a mean of 0 and a variance of 1 using mean-

sigma normalization.

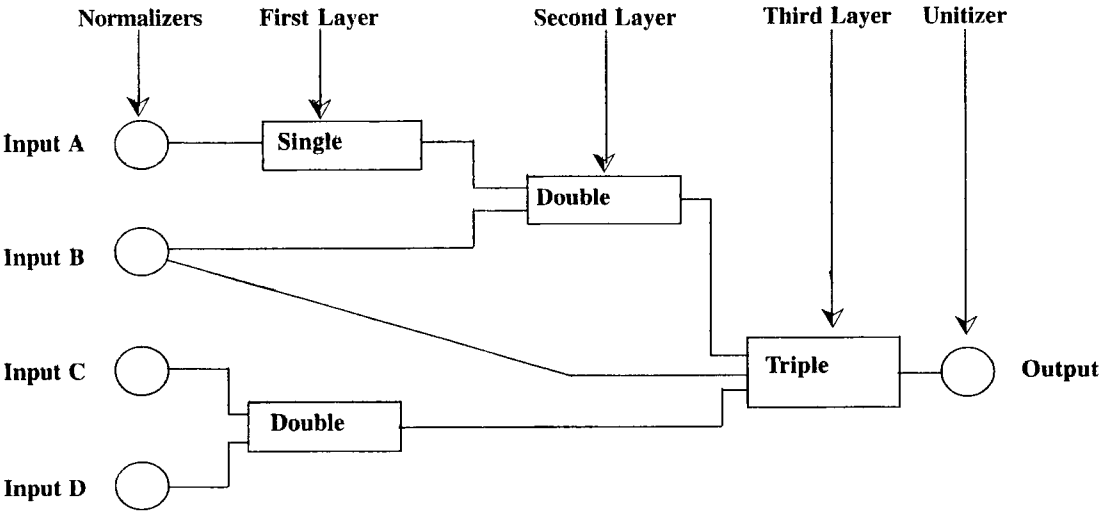
$w_0 + (w_1 x_1)$

6. **Unitizers** A unitizer converts the range of the network outputs to a range with the mean and variance of the output values used to train the network.

$w_0 + (w_1 x_1)$

7. **Wire Elements** Wire elements are used for a network that consists only of a normalizer and a unitizer.

Figure 1. Four Input, Three-Layer Abductive Network



AIM utilizes predicted squared error (PSE) to determine the optimal network configuration by reducing overfitting of the data. PSE is defined as (Barron, 1984):

$$PSE = FSE + KP$$

where FSE is the fitting squared error of the model on the training data and KP is a complexity penalty, defined as

$$KP = CPM * ((2*K) / N) * s_p^2$$

where K, N, and s_p^2 are determined by the database of examples used to synthesize the network and CPM, the Complexity Penalty Multiplier, is a user-determined variable. In general, simpler, rather than more complex models are preferred to avoid over-fitting. K is the total number of coefficients, N is the number of training data, and s_p^2 is an *a-priori* estimate of the true unknown

model error variance. As N goes up or sp2 goes down, *AIM* can fit the data with more confidence and will allow more complexity. Not all variables may be represented in the model because those that do not contribute significantly to the solution are eliminated.

Developing a Population Model

To investigate the accuracy of an abductive model, data were obtained from a study comparing discriminant analysis with logistic regression (Press & Wilson, 1978) and another study by Stevens (1992, pp. 301-302). the studies attempted to classify population changes for each state in the United States. For the target variable, a 1 was recorded for a given state if its population change between the 1960 census and the 1970 census was above the median change for all states and a 0 was recorded if it was below. Four demographic predictor variables were used to classify the population changes: average income, average deaths, average births, and presence of a coastline (0 or 1).

Table 1: Accuracies: Training on 10, Testing on 10

Training Set/ Testing Set	1	2	3	4	5	Mean
1	100%	50%	100%	60%	80%	72.5%
2	50%	100%	60%	50%	50%	52.5%
3	80%	60%	100%	70%	80%	72.5%
4	60%	70%	60%	100%	40%	57.5%
5	70%	70%	60%	50%	100%	62.5%
					Mean:	63.5%

As with neural networks and statistical techniques, the size of the training and testing sets is an important factor in the ability to accurately model the data. First the data were randomly split into five sets of 10 states each. Table 1 shows the training and testing accuracies for the data taking 10 states at a time. For example, training of the first set and testing on the first set resulted in a 100% accuracy rate -- no states were misclassified. Training on the second set of 10 (the second column), and testing on the first set, however, resulted in a 50% accuracy rate. The overall accuracy for training and testing 10 states at a time was low (63.5%), indicating that the model did not have enough observations in the training sets to adequately learn the associations among the variables.

Next, data were randomly split into five sets of 40 observations for training and 10 observations for testing in the same resampling manner as the Press & Wilson (1978) study. Adjusting the two parameters available for modification (complexity penalty multiplier and number of layers in the model), it was found that the highest testing accuracy occurred with a complexity penalty of 0.5 and four layers. This accuracy (72%) was the same achieved in the 1978 study using logistic regression. However, a more recent study using a neural network achieved a slightly higher testing accuracy of 74% (Fish, Barnes, & Aiken, 1995). The abductive model generated by the software for these parameters is shown in Figure 2 and the C code automatically provided by the software is shown in the appendix. Using the code, a predictive model may be incorporated into other software.

Figure 2. Model of Population Change (CP=0.5, Layers=4)

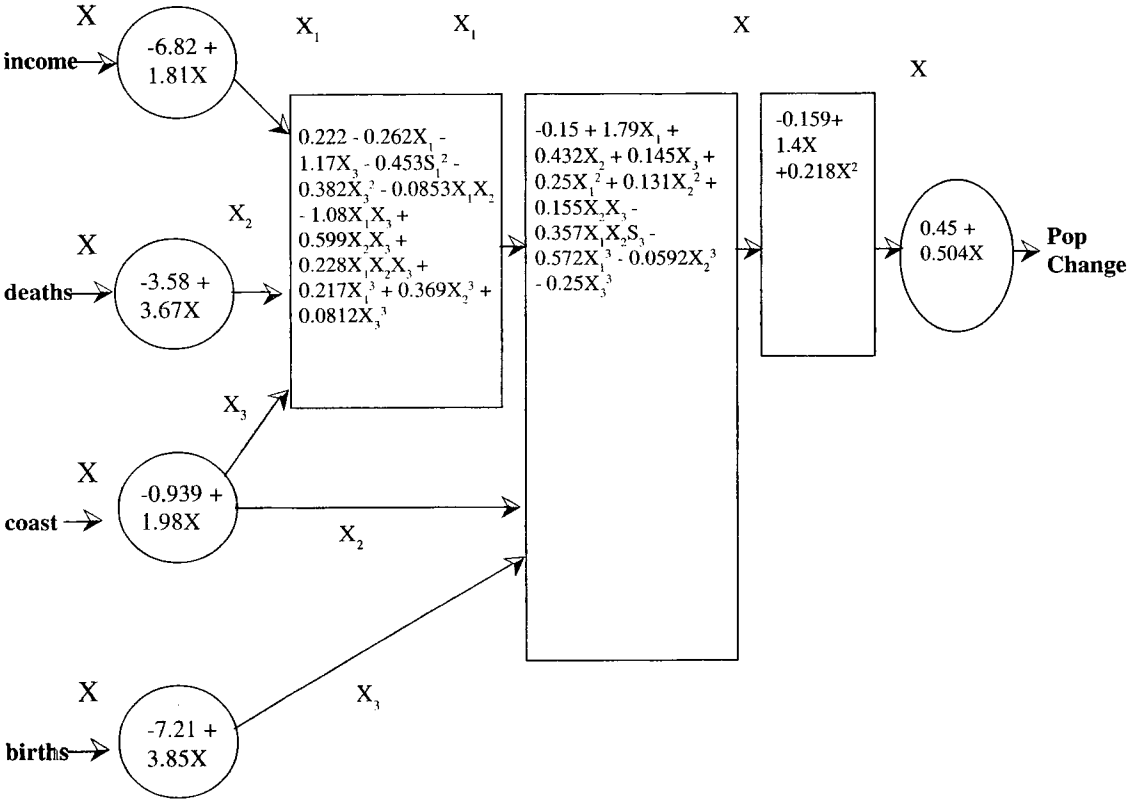
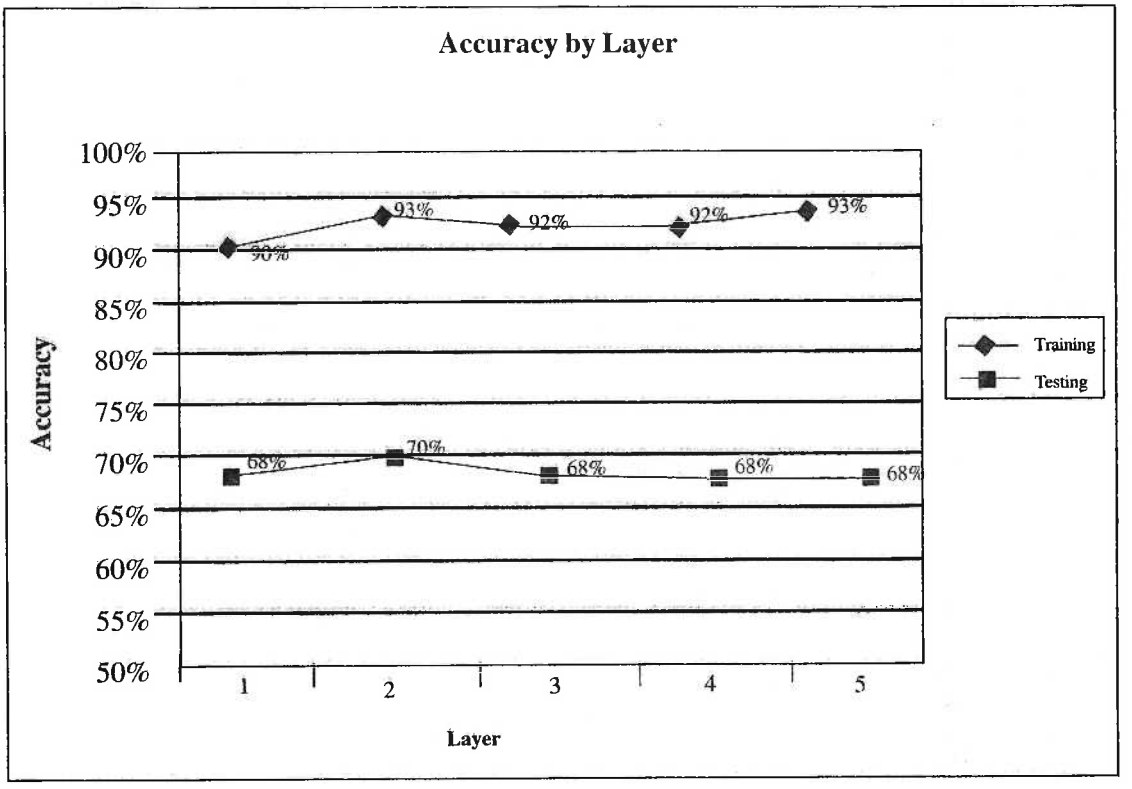
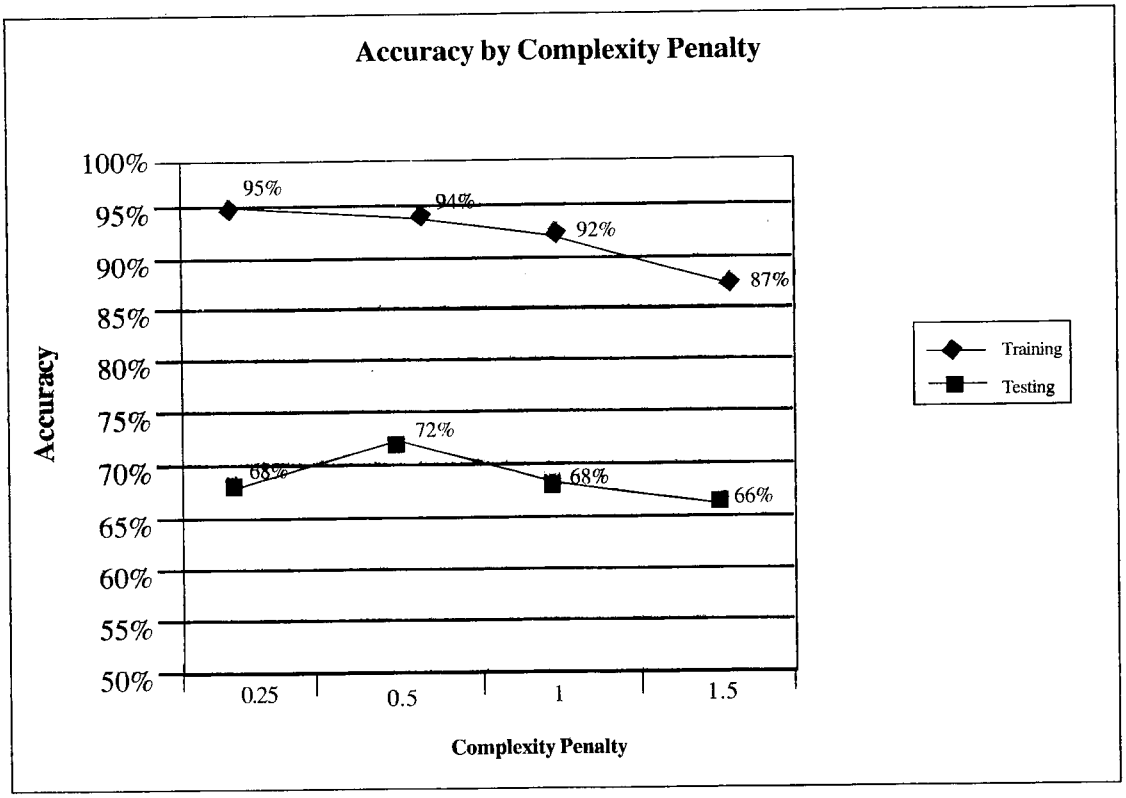


Figure 3. Training and Testing Accuracies for Population Change Model (by layer)



Figures 3 and 4 show how the training and testing accuracies varied as the complexity penalty multiplier and the number of layers were changed. Although the default setting for the software is a penalty of 1 and four layers, these settings might not be optimal. Although the highest training accuracy occurred with a penalty of 0.25, the testing accuracy was lower due to over-fitting of the model to the data. Figure 3 shows that the testing accuracy for three layers and higher is relatively constant at 68%. Even though a greater number of layers is allowed, the software will use a smaller number of layers if the resulting model is more accurate. That is, when the complexity penalty is held constant at 1, a model with two layers is optimal. Specifying four layers, Figure 4 shows that being too permissive (a low penalty) may over-fit the data while being too restrictive (a high penalty) will not allow the model to include sufficient variables and links (West, Brockett, & Golden, 1997).

Figure 4. Training and Testing Accuracies for Population Change Model (by CP)



Modifying the number of layers showed that the highest testing accuracy occurred with two layers, while modifying the complexity penalty showed that the highest accuracy occurred with a penalty of 0.5. When a model is developed specifying 0.5 and two layers, the training accuracy is high (94%), but the testing accuracy is only 68% indicating that a certain amount of heuristics or subjectivity is still necessary to develop an abductive model.

Developing an abductive network is still less subjective than developing a neural network, however. For example, there are at least 12 parameters that may be changed in *NeuroForecaster*, including learning algorithm, learning rate, error tolerance, momentum rate, maximum weight change, bias, and others (Aiken & Gai, 1994).

GUIDELINES FOR USING ABDUCTION

As with developing regression and neural net models, an adequate training sample is necessary to learn accurately (Rumelhart, Widrow, & Lehr, 1994). As Table 1 indicates, 100% training accuracies were obtained on all samples of 10, but only one test using a different set of 10 resulted in an accuracy of 100% -- most were very low. The more of a sample that can be used for training, the higher the accuracy of the model will be, but fewer observations will be available for testing with the holdout sample. For this problem, when only 20% were used for training, the overall accuracy was 63.5%. When 80% and 100% were used for training, the overall accuracies were approximately 69% and 92%, respectively.

Further, the number of observations in the training and testing sets should be balanced as much as possible. That is, the proportion of target variables with a 1 or a 0 should be approximately the same in each set for maximum predictive accuracy (Wilson & Sharda, 1994).

While the prior two guidelines apply to any forecasting technique, a few are peculiar only to abduction models. The default settings of complexity penalty modifier = 1 and number of layers = 4 in the *AIM* software may not be optimal. As discussed, for this problem, settings of (0.5, 4) and (1, 2) resulted in higher testing accuracies than the default settings. Although it has not been heuristically tested, a general rule-of-thumb for the optimal number of layers may be the number of variables divided by two. This heuristic has been observed also in deciding how many middle layers should be in a neural network architecture (Aiken, Krosp, Govindarajulu, Vanjani, & Sexton, 1995). A similar heuristic may be observed for the complexity penalty modifier, i.e., CPM = number of variables divided by two. More variables should result in a more complex models with more layers. Testing of this hypothesis is left for future research.

CONCLUSION

While the abduction classification accuracy in this population change model was no higher than that provided by logistic regression and was slightly lower than that provided by a neural network, other studies have shown that abduction may be superior to these two techniques for other problems. This paper has described this relatively unknown forecasting technique and has demonstrated the trade-offs when the *Abductive Information Modeler* software's complexity penalty modifier and maximum number of layers are changed. Future research will compare the accuracy of abduction with other techniques on other problems.

REFERENCES

AIM User's Manual. (1994). Abtec Corporation, Charlottesville, Virginia.

Aiken, M. & Alonzo, M. (in press). A comparative analysis of forecasting techniques. *International Journal of Information and Management Sciences.*

Aiken, M. & Gai, Y. (1994, Spring). Software review: NeuroForecaster 3.1. *Journal of End User Computing, 6*(2), 34.

Aiken, M., Krosp, J., Govindarajulu, C., Vanjani, M., & Sexton, R. (1995, Spring). A neural network for predicting total industrial production. *Journal of End User Computing, 7*(2), 19-23.

Aiken, M., Paolillo, J., & Vanjani, M. (1999, March 10-13). A comparison of artificial neural networks with logical abduction. *Proceedings of the 30th Southwest Decision Sciences Institute,* Houston, TX.

Aiken, M. & Vanjani, M. (1999, February 24,26). An abductive model of housing prices. *Proceedings of the 1999 Southeast Decision Sciences Institute,* Savannah, Georgia, 226-227.

Barron, A. (1984). Predicted squared error: A criterion for automatic model selection. In *Self-Organizing Methods in Modeling: GMDH Type Algorithms.* Farlow, S. (Ed.), New York: Marcel-Dekker.

Barron, R., Mucciardi, A., Cook, F., Craig, J., & Barron, A. (1984). Adaptive learning networks: Development and application in the United States of algorithms related to GMDH. In *Self-Organizing Methods in Modeling: GMDH Type Algorithms.* Farlow, S. (Ed.), New York: Marcel-Dekker.

Brooks, H. & Probert, T. (1984). Let's ask GMDH what effect the environment has on fisheries. In *Self-Organizing Methods in Modeling: GMDH Type Algorithms.* Farlow, S. (Ed.), New York: Marcel-Dekker.

Dorsey, R., Johnson, J., & Mayer, W. (1994). A genetic algorithm for the training of feedforward neural networks. In *Advances in Artificial Intelligence in Economics, Finance, and Management.* Whinston, A. and Johnson, J. (Eds.), New York: JAI Press.

Fann, K. (1970). *Peirce's Theory of Abduction,* The Hague: Martinus Nijhoff.

Farlow, S. (1984). The GMDH algorithm. In *Self-Organizing Methods in Modeling: GMDH Type Algorithms.* Farlow, S. (Ed.), New York: Marcel-Dekker.

Fish, K., Barnes, J., & Aiken, M. (1995). Artificial neural networks: A new methodology for industrial market segmentation. *Industrial Marketing Management, 24*(5) 431-438.

Hess, P. & Montgomery, G. (1988, October). Abduction: Theory and application. *Proceedings of the 4th Annual Aerospace Applications of Artificial Intelligence Conference (AAAIC).*

Ivakhnenko, A., Krotov, G., & Visotsky, V. (1979). Identification of the mathematical model of a complex system by the self-organization method. In *Theoretical Systems Ecology: Advances and Case Studies.* Halfon, E. (Ed.), New York: Academic Press.

Lebow, W., Mehra, R., & Toldalagi, P. (1984). Forecasting applications of GMDH in agricultural and meteorological time series. In *Self-Organizing Methods in Modeling: GMDH type Algorithms.* Farlow, S. (Ed.), New York: Marcel-Dekker.

Montgomery, G. (1989). Abductive diagnostics. *Proceedings of the AIAA Computers in Aerospace Conference,* Monterey, California.

Nomura, J. (1984). A method for predicting sales amount by the use of IWSM and GMDH. In *Self-Organizing Methods in Modeling: GMDH Type Algorithms.* Farlow, S. (Ed.), New York: Marcel-Dekker.

Ohashi, K. (1984). GMDH forecasting of U. S. interest rates. In *Self-Organizing Methods in Modeling: GMDH Type Algorithms.* Farlow, S. (Ed.), New York: Marcel-Dekker.

Prager, M. & Saila, S. (1984). Predictive GMDH models of shrimp catches: Some practical considerations. In *Self-Organizing Methods in Modeling: GMDH Type Algorithms.* Farlow, S. (Ed.), New York: Marcel-Dekker.

Press, S. & Wilson, S. (1978, December). Choosing between logistic regression and discriminant analysis. *Journal of the American Statistical Association, 73*(364), 699-705.

Rumelhart, D., Widrow, B., & Lehr, M. (1994, March). The basic ideas in neural networks. *Communications of the ACM, 37*(3), 87-92.

Saburo, I. (1984). Nonlinear prediction models for river flows and typhoon precipitation by self-organizing methods. In *Self-Organizing Methods in Modeling: GMDH Type Algorithms.* Farlow, S. (Ed.), New York: Marcel-Dekker.

Scott, D. & Hutchinson, C. (1984). An application of the GMDH algorithm to economic modeling. In *Self-Organizing Methods in Modeling: GMDH Type Algorithms.* Farlow, S. (Ed.), New York: Marcel-Dekker.

Stevens, J. (1992). *Applied Multivariate Statistics for the Social Sciences,* 2nd Ed., Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Wasserman, P. (1989). *Neural Computing: Theory and Practice,* New York: Van Nostrand Reinhold.

West, P., Brockett, P., & Golden, L. (1997). A comparative analysis of neural networks and statistical methods for predicting consumer choice. *Marketing Science, 16*(4).

Wilson, R. & Sharda, R. (1994). Bankruptcy prediction using neural networks. *Decision Support Systems, 11,* 545-557.

Appendix - Population Change C Code

```
#include <stdio.h>
#define powl (x)  (x)
#define pow2 (x)  ( (x) * (x) )
#define pow3 (x)  ( (x) * (x) * (x) )
#define LIMIT (v, mn, mx)  ( (v>(mx)) ? (mx) : ( (v<(mn) )? (mn) : v) )
/*****************************************************************
ABDUCTIVE NETWORK SUBROUTINE:  AIMnet ( )
Generated by AbTech's Abductory Induction Mechanism
INPUTS:  the following 4 double(s):
        income
        coast
        deaths
        births
OUTPUTS:  Pointer(s) to the following 1 double(s):
        change
STATISTICS:
    income    :    Mean  =   3.75925,    Sigma  =   0.551135
                   Min   =   2.948,      Max    =   4.917
    coast     :    Mean  =   0.475,      Sigma  =   0.505736
                   Min   =   0,          Max    =   1
    deaths    :    Mean  =   0.975,      Sigma  =   0.272453
                   Min   =   0.5,        Max    =   2.4
    births    :    Mean  =   1.875,      Sigma  =   0.259931
                   Min   =   1.5,        Max    =   2.7
    change    :    Mean  =   0.45,       Sigma  =   0.503831
              :    Min   =   0,          Max    =   1
              :    KP    =   25.8%,      FSE    =   74.2%
              :    Predicted Error = 0.288394

*****************************************************************/
AIMnet (income, coast, deaths, births, change)
double   income  ;      /*  input variable */
double   coast   ;      /*  input variable */
double   deaths  ;      /*  input variable */
double   births  ;      /*  input variable */
double  *change  ;      /*  input variable */
{
double node2   ;    /*  working variable */
double node4   ;    /*  working variable */
double node6   ;    /*  working variable */
double node9   ;    /*  working variable */
double node3   ;    /*  working variable */
double node8   ;    /*  working variable */
double node7   ;    /*  working variable */
```

```
#ifdef DEBUG
    printf ("AIMnet:   received income = %g. \n" , (double) income);
    printf ("AIMnet:   received coast = %g. \n" , (double) coast);
    printf ("AIMnet:   received deaths = %g. \n" , (double) deaths);
    printf ("AIMnet:   received births = %g. \n" , (double) births);
#endif   /*   DEBUG   */

   /* node2 -- income */
   node2     = -6.82092 + 1.81444*LIMIT (income, 2.948, 4.917);

   /* node4 -- coast */
   node4     = -0.939225 + 1.97731*LIMIT (coast, 0, 1);

   /* node6 -- deaths */
   node6     = -3.5786 + 3.67035*LIMIT (deaths, 0.5, 2.4);

   /* node9 -- Triple */
   node9     = 0 + 0.222279 - 0.262496*node2 - 1.16807*node6
                - 0.45334*pow2 (node2) - 0.381815*pow2 (node6)
                - 0.0853384*node2*node4 - 1.08216*node2*node6
                + 0.599155*node4*node6 + 0.22813*node2*node4*node6
                + 0.216619*pow3 (node2) + 0.369197*pow3 (node4)
                + 0.081151*pow3 (node6) ;

   /*   node3--births */
   node3     =    -7.21345 + 3.84718*LIMIT (births, 1.5, 2.7);

   /*   node8--Triple */
   node8     = 0 - 0.14999 + 1.78591*node9 + 0.432448*node3
                + 0.15411*node4 + 0.250397*pow2 (node9)
                + 0.13076*pow2 (node3) + 0.15474*node3*node4
                - 0.35695*node9*node3*node4 - 0.571903*pow3 (node9)
                - 0.059167*pow3 (node3) - 0.25*pow3 (node4) ;

   /*   node7--Single */
   node7     = 0 - 0.159035 + 1.40406*node8 + 0.217865*pow2 (node8)
                - 0.400344*pow3 (node8) ;

   /*   node1--change */
   *change  = 0 + 0.45 + 0.503831*node7 ;

   /*   perform output limiting on change */
   *change = LIMIT (  *change, 0, 1 ) ;

#ifdef  DEGUG
  printf ("AIMnet:   returning change = %g.\n", *change) ;
#endif   /*  DEGUB  */

}
```