EUROPEAN JOURNAL OF PURE AND APPLIED MATHEMATICS
Vol. 3, No. 2, 2010, 163-173
ISSN 1307-5543 – www.ejpam.com

A Self-Organizing Model for Logic Regression

Jerry Farlow

Department of Mathematics, University of Maine, Orono, Maine

Abstract: Logic regression, as developed by Ruczinski, Kooperberg, and LeBlanc [1], is a multivariable regression methodology that constructs *logical relationships* among Boolean predictor variables that best predicts a Boolean dependent variable. More specifically, it finds a regression model of the form $g[E\,|\,Y] = b_0 + b_1 L_1 + b_2 L_2 + \cdots b_m L_m$ where both the coefficients $b_0, b_1, \ldots, b_m$ and the logical expressions $L_j,\ j = 1, \ldots, m$ are determined, whereby a logical expression one means logical relationships among the predictor variables, such as $"X_1, X_2$ *are true but not* $X_5"$, or $"X_3, X_5, X_7$ *are true but not* X_1 *or* $X_2"$ In paper [1] the authors investigate the use a simulated annealing algorithm. In this paper. the Group Method of Data Handling (GMDH) is used.

2000 Mathematics Subject Classifications: 62

Key Words and Phrases: Logic regression, GMDH, algorithm, self-organizing methods.

1. Introduction

In this paper we study the possibility of using Ivakhenko's Group Method of Data Handling (GMDH) algorithm for finding the best logical expressions L_j among Boolean predictor variables $X_1, X_2, \ldots, X_m$ which best predicts a dependent Boolean variable Y. By logical expressions L_j, we mean Boolean expressions such as $"X_1, X_2$ *are true but not* $X_5"$, or $"X_3, X_5, X_7$ *are true but not* X_1 *or* $X_2"$ or $"$*if* X_1 *and* X_2 *are true then* X_5 *is true.* Before showing how this algorithm is performed, since the GMDH algorithm is not well-known, we summarize Ivakhnenko's basic algorithm for ordinary regression.

Email address: `norman.j.morin@frb.gov`

2. Basic Group Method of Data Handling (GMDH) Algorithm

2.1 Basic GMDH Method

If one were to anthropomorphize, one might say the GMDH algorithm builds a mathematical model similar to the way biological organisms are created through evolution. That is, starting with a few basic primeval forms (i.e. equations); one grows a new generation of more complex off-springs (equations) and then allows for a survival-of-the-fittest principle to determine which new off-springs survive and which do not. The idea is that new generations of off-springs (equations) are better suited to model the real world than earlier ones. Continuing this process for more generations, one finds a collection of models that hopefully describes the problem at hand. The process is stopped once the model begins to "over-fit" the real world, thus stopping when the model reaches some level of *optimal complexity*.

In 1966, Ukrainian cyberneticist, A.G. Ivakhnenko, discouraged by the fact that many mathematical models require the modeler to know things about the real world that are difficult or impossible to know, produced a heuristic self-organizing model, called the Group Method of Data Handling algorithm. For more information see Farlow, [1] or view one of the many websites which discuss applications of the technique. Ivakhnenko's website at http://www.gmdh.net/ will give the reader a complete history of the method.

The basic GMDH algorithm can be broken into a few distinct steps.

Step 1 (constructing new variables $z_1, z_2, \ldots, z_{C(m,2)}$)

The basic GMDH algorithm begins with regression-type data points of the form $y_i, x_{i1}, x_{i2}, \ldots, x_{im}$, $i = 1, 2, \ldots, n$, where the n observations are subdivided into two groups, the first nt observations are called the *training observations* and the remaining $nc = n - nt$ observations are called the *checking observations*. See the data in Figure 1. Normally, about half the observations are chosen to be in each group.

	Y		X		
	Y	x_1	x_2	$\cdots$	x_m
Training set	y_1	x_{11}	x_{12}	$\cdots$	x_{1m}
	y_2	x_{21}	x_{22}	$\cdots$	x_{2m}
	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
	y_{nt}	$x_{nt,1}$	$x_{nt,2}$	$\cdots$	$x_{nt,m}$
Checking set	y_{nt+1}	$x_{nt+1,1}$	$x_{nt+1,2}$	$\cdots$	$x_{nt+1,m}$
	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
	y_n	$x_{n,1}$	$x_{n,2}$	$\cdots$	x_{nm}

Figure 1: Regression-Type Data for the GMDH Algorithm

Now, for each $C(m,2) = m(m-1)/2$ *pair* of distinct variables x_i, x_j one finds the least-squares regression polynomial for *y* of the form

$$y = A + Bx_i + Cx_j + Dx_i^2 + Ex_j^2 + Fx_i x_j \tag{1}$$

(i.e. find A, B, C, D, E, F) from the observations in the *training set* These $m(m-1)/2$ regression surfaces are illustrated in Figure 2.

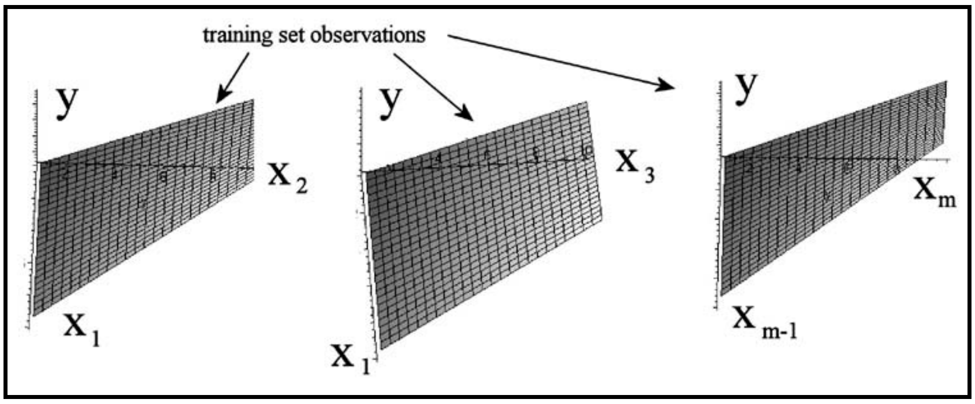

Figure 2: Computed Quadratic Regression Surfaces

Now evaluate each of the $m(m-1)/2$ regression polynomials at all *n* data points and store these values (new generation of variables) in the respective columns of a new array, say *Z*. The evaluation of the first regression polynomial and its values in the first column of *Z* is illustrated in Figure 3.

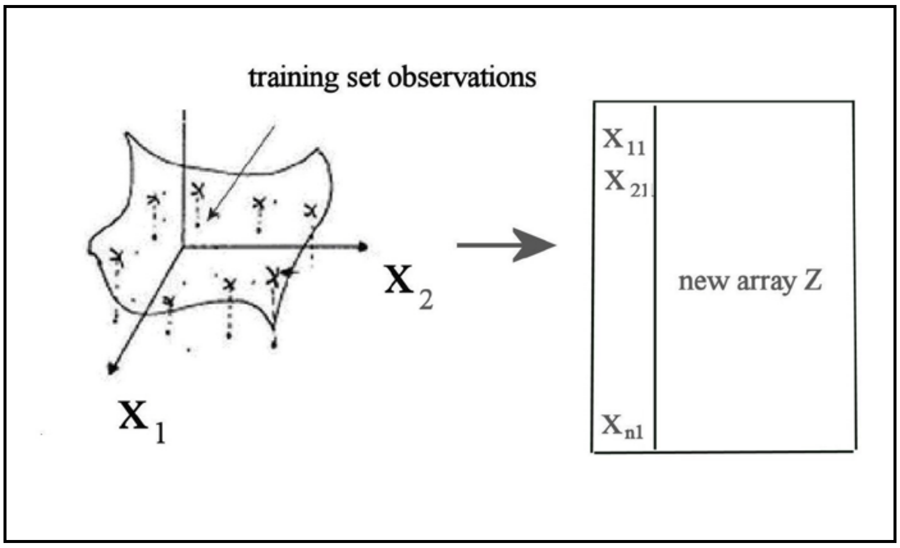

Figure 3: Evaluating the Quadratic Regression Polynomials

The object is to keep only the best of these new variables and this is where the observations in the checking set come into play.

Step 2 (screening out the least effective variables)

This step replaces the original variables (columns of *X*) by those columns of *Z* that best predict *y*, based on the checking set observations. That is, for each column *j* of *Z* we compute the *root mean square* (or some measure of association) r_j given by Eq. (2).

$$\boxed{r_j^2 = \frac{\sum_{i=nt+1}^{n} (y_i - z_{ij})^2}{\sum_{i=nt+1}^{n} y_i^2}, \qquad j = 1, 2, \ldots, C(m, 2)} \tag{2}$$

and then select those columns of Z that satisfy $r_j < R$, where R is some prescribed number. The number of columns of *Z* that replace columns of *X* may be larger or smaller than the number of columns of *X*, although often one keeps the number of columns of *X* constant at *m*. Note that the test of goodness of fit r_j was summed over the observation in the checking set.

Step 3 (test for optimality)

From Step 2 we find the smallest of the $r_j\text{'s}$ and call it RMIN. Then, each time one completes a generation or iteration, one plots the value of RMIN on a graph as shown in Figure 4.

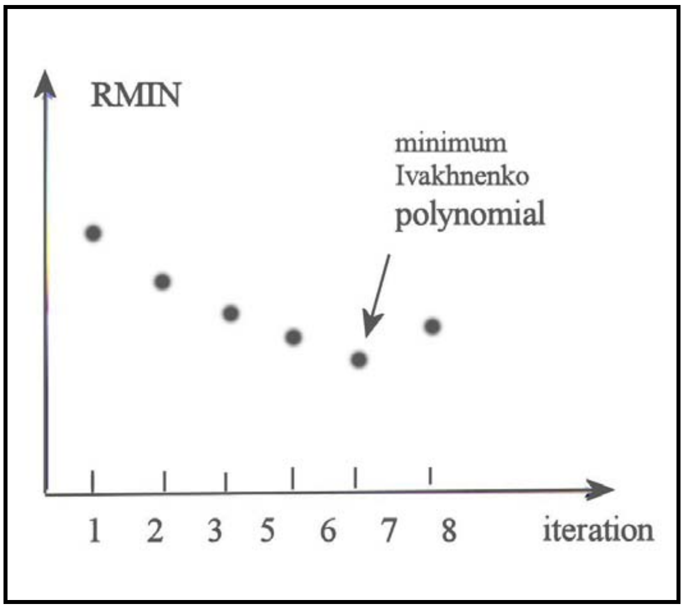

Figure 4: Determining the Optimal Polynomial

Experiments have shown that RMIN decreases for a few generations (the author's experience is maybe 3-5 iterations) and then begins to increase, the reason being the model gets better and better but eventually starts to over-fit the data. Hence, the rule is to stop the algorithm when the RMIN curve reaches its minimum, and then select the column with the minimum r_j value of the final array Z as the best predictor. When the GMDH algorithm stops, the columns of *Z* (in particular the column of Z that has the smallest r_j value) contains the computed values of a high-order polynomial of the form in Eq (3)

$$\boxed{\overline{y}_1 = a + \sum_{i=1}^{m} b_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij} x_j x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} d_{ijk} x_i x_j x_k + \cdots} \tag{3}$$

known as the *Ivakhnenko polynomial.* At each iteration the degree of the Ivakhnenko doubles, and for a *p-th* order regression polynomial the number of terms in the polynomial will be $(m+1)(m+2)\cdots(m+p)/m!$. For example, if one starts with $m=10$ input variables $x_1, x_2, ..., x_{10}$ and the algorithm is continued for 8 generations, the Ivakhnenko polynomial would be a polynomial in $x_1, x_2, \ldots, x_m$ of degree $2^8 = 256$. A sample term might involve the variables $x_1{}^2 x_3{}^4 x_4{}^9 x_6{}^{11} x_7{}^9 x_{10}{}^3$.

Step 4 (Applying the results of the GMDH Algorithm)

One doesn't actually compute the coefficients in the Ivakhenko polynomial, but saves the regression coefficients *A,B,C,D,E,F* at each generation. Hence, to evaluate the Ivakhnenko polynomial one simply carries out repeated compositions of these quadratic expressions. Figure 5 illustrates this process.

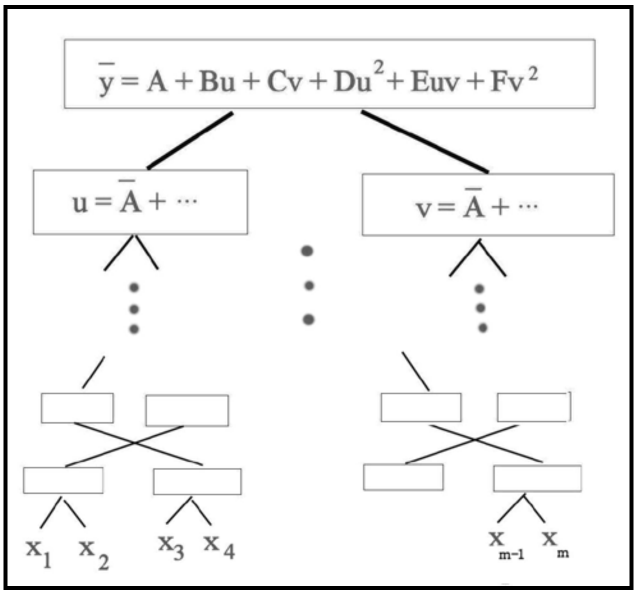
$\overline{y} = A + Bu + Cv + Du^2 + Euv + Fv^2$
$u = \overline{A} + \cdots$
$v = \overline{A} + \cdots$
x_1 x_2 x_3 x_4 x_{m-1} x_m

Figure 5: Evaluation of the Ivankhenko Polynomial

3. GMDH Algorithm Applied to Logic Regression

Step 1: (Divide Observations into Training and Checking Sets)

Starting with n observations of m Boolean predictor variables $X_1, X_2, ..., X_m$ and a dependent Boolean variable *Y*, we subdivide the observations into *nt* training set observations and $nc = n - nt$ checking set observations. For each of the

$$\binom{m}{2} = \frac{m(m-1)}{2}$$

distinct pairs

$$\left\{X_i, X_j : i = 1,..m-1,\ j = i+1, m\right\}$$

of predictor variables, we find the logical function that best predicts the dependent variable Y from among the 16 binary functions in Table 1.

Table 1: Sixteen Boolean Functions

0(0000)	FFFF	never true	-----
1(0001)	FFFT	not (X_1 or X_2)	$\bar{X}_1 \wedge \bar{X}_2$
2(0010)	FFTF	X_2 but not X_1	$\bar{X}_1 \wedge X_2$
3(0011)	FFTT	not X_1	$\bar{X}_1$
4(0100)	FTFF	X_1 but not X_2	$X_1 \wedge \bar{X}_2$
5(0101)	FTFT	not X_2	$\bar{X}_2$
6(0110)	FTTF	X_1 or X_2 but not both	$X_1 \veebar X_2$
7(0111)	FTTT	not (X_1 and X_2)	$\bar{X}_1 \vee \bar{X}_2$
8(1000)	TFFF	X_1 and X_2	$X_1 \wedge X_2$
9(1001)	TFFT	X_1 is X_2	$X_1 \equiv X_2$
10(1010)	TFTF	X_2	X_2
11(1011)	TFTT	If X_1 then X_2	$X_1 \Rightarrow X_2$
12(1100)	TTFF	X_1	X_1
13(1101)	TTFT	if X_2 then X_1	$X_1 \Leftarrow X_2$
14(1110)	TTTF	X_1 or X_2	$X_1 \vee X_2$
16(1111)	TTTT	always true	-----

We illustrate the process with the data set in Table 2, which has $n = 7$ total observations, $nt = 4$ training observations, $nc = n - nt = 7 - 4 = 3$, and $m = 4$ variables

Table 2: Sample Data

	Y	X_1	X_2	X_3	X_4
Training Set	1	0	0	1	0
	0	1	0	0	1
	0	0	1	1	1
	1	1	1	0	1

Checking Set	1	0	1	0	1
	0	1	0	0	1
	0	0	0	1	1

Note that the dependent variable X_1 did not predict Y in the first- and second observations (1s and 0s do not match), but do predict Y in the 3rd and 4th observations (1s and 0s match). Note, too that the logical relation $\overline{X}_1 \wedge \overline{X}_2$ correctly predicts Y in the 1st, 2nd, and 3rd observations, but not the 4th, hence the recorded values 1, 1, 1, and 0 in the respective column, and a 3 in the bottom row illustrating the number of correct predictions. If one carries out this analysis for all 16 logical expressions of X_1, X_2, one arrives at the results in Table 3.

Table 3: Correct and Incorrect Predictions of Y from X_1, X_2.

0 (0000) false	1 (0001) $\overline{X}_1 \wedge \overline{X}_2$	2 (0010) $X_1 \wedge \overline{X}_2$	3 (0011) $\overline{X}_1$	4 (0100) $X_1 \wedge \overline{X}_2$	5 (0101) $\overline{X}_2$	6 (0110) $X_1 \dot{\vee} X_2$	7 (0111) $\overline{X}_1 \vee \overline{X}_2$
0	1	0	1	0	1	0	1
1	1	1	1	0	0	0	0
1	1	0	0	1	1	0	0
0	0	0	0	0	0	0	0
2	**3**	**1**	**2**	**1**	**2**	**0**	**1**

8(1000) $X_1 \wedge X_2$	9(1001) $X_1 \equiv X_2$	10(1010) X_2	11(1011) $X_1 \Rightarrow X_2$	12(1100) X_1	13(1101) $X_1 \Leftarrow X_2$	14(1110) $X_1 \vee X_2$	15(1111) tautology
0	1	0	1	0	1	0	1
1	1	1	1	0	0	0	0
1	1	0	0	1	1	0	0
1	1	1	1	1	1	1	1
3	**4**	**2**	**3**	**2**	**3**	**1**	**2**

Note the best logical predictor is 9(1001), which represents the logical function $X_1 \equiv X_2$.

Table 3 is then computed for each of the $m(m-1)/2 = 6$ pairs of dependent variables, recording only the last row of totals which represent the number of correct predictions of a given logical expression. Doing this we arrive at Table 4, where the second row in the table lists the best logical predictor of Y from among the 16 logical functions of the given pair of variables, and the bottom row 3 gives the fraction (and percentage) of times the given logical function accurately predicted Y.

Table 4: Results for the first iteration.

Variable pairs	X_1, X_2	X_1, X_3	X_1, X_4	X_2, X_3	X_2, X_4	X_3, X_4
Function	$X_1 \equiv \bar{X}_2$	$X_1 \wedge X_3$	$X_1 \vee X_4$	$X_2 \Leftarrow X_3$	$X_2 \Rightarrow X_4$	$\bar{X}_3 \vee \bar{X}_4$
# correct	4/4(100%)	3/4(75%)	4/4(100%)	2/4(50%)	3/4(75%)	2/4 (50%)

The next step is to select the best m functions from Table 4 (4 in this example) that predicts Y.

Step 2: (Replace the Original Data with the Best Logical Estimates)

We now evaluate the best m logical functions in Table 4, which represent the best logical predictors of Y based on the observations in the training set. In this example they are $X_1 \equiv \bar{X}_2$, $X_1 \wedge X_3$, $X_1 \vee X_4$, $X_2 \Rightarrow X_4$ which yields 4, 3, 4, and 3 correct predictions. Evaluating these functions at the n observations, we arrive at the $n \times m = 4 \times 4$ array in Table 5, which we call XNEW and are the evaluated best predictors of Y. Note that in the column under $X_2 \Rightarrow X_4$ for the observations in the checking set, the values are 1, 1, and 1, which are the logical evaluations of the corresponding observations in the original data matrix X in Table 2. We now replace the data matrix X of independent variables with the newly computed matrix XNEW.

Table 5: New Computed Data XNEW Replaces X.

	$X_1 \equiv \bar{X}_2$	$X_1 \wedge X_3$	$X_1 \vee X_4$	$X_2 \Rightarrow X_4$
Training Set	1	0	1	0
	0	0	0	1
	0	1	1	1
	1	1	0	1

Checking Set	1	0	0	1
	0	0	1	1
	0	0	0	1

Step 3: (When to Stop: Goodness of Fit)

Steps 1 and 2 define the algorithm. To determine when the process is stopped, we compute the number of correct predictions RMAX of Y at the end of Step 2 for those observations in the *checking set.* This is an easy thing to do since best predictions reside in the first column of XNEW (and also the first column of X). Experiments have shown that RMAX increases for a few generations and then begins to decrease as the model begins to over-fit the data. Hence, the rule is to stop the algorithm when the RMAX curve reaches its maximum and select the first column of XNEW as the best predictor. When the algorithm stops, the columns of XNEW contain the computed values of the best m logical predictors of Y. This defines the algorithm.

4. Computations

We tested the algorithm's ability to find best logical relations of several independent variables from among $m = 50$ independent variables with $n = 250$ observations ($nt = 150$, $nc = 100$). All independent variables were binary 0 and 1 variables, each having probability 0.5. The dependent variable Y was generated as a logical expression of the dependent variables for some observations, and random 0s and 1s for other observations. In this way we could determine how well the GMDH algorithm works.

To carry out the generation of the dependent variable Y, we first select a number $0 \le q \le 1$, then generate a uniform random number r between 0 and 1, and then the observations y_i, $i = 1, \ldots, n$ computed by the rule

$$y_i = \begin{cases} \text{specified logical function when } r \ge q \\ \text{random 0 or 1 when } r < q \end{cases}$$

Note that when $q = 0$ the dependent observations y_i are the values of specified logical function of the dependent variables, and when $q = 1$ the computed values of Y are random 0s and 1s and have no relation to the independent variables. The goal was to determine how well the algorithm could pick out the chosen logical relation when $0 < q < 1$.

Experiment 1: The value $q = 1$ was chosen with logical function $X_{15} \Rightarrow X_{35}$. It was not surprising that at the first iteration the maximum logical function was $X_{15} \Rightarrow X_{35}$ and that it predicted the dependent variable 150 out of 150 times in the training set and 100 out of 100 times in the checking set. The second through fifth place logical relations were (2) $X_{15} \Rightarrow X_{42}$, (3) not both X_{15} and X_{49}, (4) not both X_{15} and X_{16}, and (5) $X_{16} \Rightarrow X_{35}$ which is not surprising since the chosen logical relation $X_{15} \Rightarrow X_{35} \equiv \overline{X}_{15} \cup X_{35}$ so the variables $\overline{X}_{15}$ and X_{35} would naturally come into play. The following Table 6 shows the results for the same logical function for different values of q after 1 iteration.

Table 6: Correct predictions of Y after 1 iteration for different values of q.

Value of q	1	0.9	0.8	0.7	0.6
Best logical functions	$X_{15} \Rightarrow X_{35}$	$X_{15} \Rightarrow X_{35}$	$X_{15} \Rightarrow X_{35}$	$X_{15} \Rightarrow X_{35}$	$X_{15} \Rightarrow X_{35}$
Fraction of correct predictions in the training set	150/150 (100%)	143/150 (95%)	134/150 (89%)	132/150 (88%)	124/150 (83%)

Fraction of correct predictions in the checking set	100/100 (100%)	97/100 (97%)	89/100 (89%)	84/100 (84%)	79/100 (79%)

Experiment 2: Using the same data and values of q as in Experiment 1, but with the new logical function $y = (X_5 \cup X_{10}) \cap (X_{15} \cup X_{20})$, we arrived at the results in Table 7 after two iterations. In these cases the algorithm reached its maximum predictions after only two iterations. Note that when $q = 0.5$ the maximum logical function was close to the entered logical function but not exact. This is not surprising since only 50% of the dependent variables were computed from $y_i = (X_5 \cup X_{10}) \cap (X_{15} \cup X_{20})$.

Table 7: Results after 2 Iterations

Value of q	1	0.8
Best logical function	$(X_5 \cup X_{10}) \cap (X_{15} \cup X_{20}$	$(X_5 \cup X_{10}) \cap (X_{15} \cup X_{20})$
Fraction of correct predictions in the training set	150/150 (100%)	140/150 (93%)
Fraction of correct predictions in the checking set	100/100 (100%)	87/100 (87%)
Value of q	0.6	0.5
Best logical function	$(X_5 \cup X_{10}) \cap (X_{15} \cup X_{20}$	$(X_{10} \cup X_{20}) \cap (X_{10} \cup X_{15})$
Fraction of correct predictions in the training set	120/150 (80%)	110/150 (73%)
Fraction of correct predictions in the checking set	74/100 (74%)	71/100 (71%)

5. Conclusions

From the experiments performed we conclude that the GMDH algorithm is an effective tool in finding logical relations among predictor variables.

References

[1] I. Ruczinski, C. Kooperberg, and M. LeBlanc. Logic Regression. *Journal of Computational and Graphical Statistics* 12(3), 475-511, 2003.

[2] S. J. Farlow. Self-organizing Methods in Modeling. *GMDH Type Algorithms, S. J. Farlow, editor. Marcel Dekker* , 1984.