Proceedings of ACEDC'96
PEDC, University of Plymouth, UK

PRACTICAL IMPLEMENTATION AND USE OF GROUP METHOD OF DATA HANDLING (GMDH): PROSPECTS AND PROBLEMS

S A Dolenko, Yu V Orlov, and I G Persiantsev
Nuclear Physics Institute, Moscow State University, Moscow, 119899 Russia
E-mail at Internet: dolenko@micnel.npi.msu.su

Issues of practical implementation of Group Method of Data Handling (GMDH) are discussed. The method was tested on a wide range of artificial and real-world problems. The method allows to find an analytical formula, which expresses the dependence of modeled system output on the values of most significant inputs of the system. GMDH proved to be most effective to solve small and medium-sized problems with continuous output.

Basics of the Method

One of the most common problems in engineering design and control is the problem of mathematical modeling. Consider the object under investigation as a "black box" with several input variables (inputs) and one output variable (output). The purpose of modeling is to find some means of predicting the value of output for any values of inputs, based on a set of learning data.

One of the methods of mathematical modeling is the Group Method of Data Handling (GMDH) [Ivakhnenko, (1), Farlow, ed. (2), Ivakhnenko et al (3)]. The idea of GMDH is the following: we are trying to build an analytical function (called "model") which would behave itself in such a way that the predicted value of the output would be as close as possible to its actual value. For many applications such an analytical model is much more convenient than the "distributed knowledge" representation that is typical for neural network approach.

The most common way to deal with such a problem is to use linear regression approach. In this approach, first of all we must introduce a set of basis functions. The answer will then be sought as a linear combination of the basis functions. For example, powers of input variables along with their double and triple cross-products may be chosen as basis functions.

To obtain the best solution, we should try all possible combinations of terms, and choose those which give best predictions. The decision about quality of each model must be made using some numeric criterion. (Correct choice of the criterion is a separate problem.) However, it is clear that full testing for a problem with many inputs and a wide set of basis functions is practically impossible, as it would take too much time and it would require too much computer memory. To reduce computational expenses, one should reduce the number of basis functions (and the number of input variables), which are used to build the tested models. To do that, one must change from a one-stage procedure of model selection to a multi-stage procedure.

Let us take first two input variables and let us combine a simple set of basis functions. For example, if we denote input variables as x_1 and x_2, let the set of basis functions be $\{1, x_1, x_2, x_1 \cdot x_2\}$. (1 corresponds to constant bias and must be always included in the set.) Now we check $2^4-1=15$ possible models, and choose one that is the best. (Any one of the tested models is often called partial description, or PD.)

After that, we take another pair of input variables, and repeat operation, resulting in one more PD with its own value of criterion. Doing the same for each possible pair of n input variables, we obtain $n \cdot (n-1)/2$ PDs, each with its own value of the used criterion.

Then we compare these values and choose several PDs which give better approximation for the output variable. Usually we select a pre-defined number F of best PDs that must be preserved at the next step of algorithm.

The values predicted by the preserved PDs (called Survivors), serve at the next iteration as input variables along with initial input variables of the whole system. All the described actions are repeated again with the broadened set of input variables, then the next iteration goes, etc.

The work of this algorithm has a straightforward analogy with work of a gardener during selection of a new hybrid plant. The gardener sows some seeds, waits for the plants to grow and selects several plants that bear the property desired for the hybrid to stronger extent. Then he collects seeds from the selected plants and sows these seeds again, bringing up a second generation, etc., until he obtains a plant with the desired property.

GMDHTEST: the Opportunities of the Software Package

The described algorithm was implemented as a computer program running under Microsoft Windows 3.1 and called GMDHTEST. The main features of the program are:

- This version of the algorithm uses polynomial basis functions. The set of these functions (and thus possible Partial Descriptions) may be affected by special parameters.

- The formula obtained at each iteration, analytically expresses the output in terms of initial inputs, rather than in terms of previous iteration.
- We may choose from several types of selection criteria. Six types of these has been used before [Farlow, ed. (2), Ivakhnenko et al (3), Barron and Xiao (4)]. In this work, we have also used a new type of selection criterion.
- After the model is obtained, it should be checked if some insignificant terms may be removed from the model. Such procedure is called Optimization. There are several modes of optimization which trade thoroughness for speed.
- Besides non-linear models, the algorithm may consider extended linear models with up to 15 most significant input variables.
- Besides usual sequential testing of all the possible combinations of inputs, the program allows random selection of the next tested combination.
- Scaling of all inputs and the output is possible into [0..1], [-1..1] or [±variance] for each variable. Scaling may be also turned off.

Results

The program described above was used to get optimal models for several artificial (generated) and real-life problems.

Estimation Criteria. We used two main criteria for estimation of method performance: the linear correlation coefficient **C** and the coefficient of multiple determination **R**, which is expressed by the formula:

$$R = 1 - \frac{\sum (\text{actual} - \text{predicted})^2}{\sum (\text{actual} - \text{mean})^2}$$

The values of **R** and **C** for tested problems are summarized in Table 1.

Modeling of Complex Functions. To test the ability of GMDH to select really important input variables, we used two artificial problems from the work by Friedman (5). Each of these problems had 10 inputs and one output. The underlying formulae were in both cases complex non-linear functions of only five first input variables, using polynomials, exp() and sin() functions. All the input variables were generated in [0..1] range by random. The train and test sets had 200 patterns each. The algorithm demonstrated a stable ability of approximation of the target dependencies without using any phantom variables in the resulting formulae, unless the train set was too small (less than 100 patterns).

Sunspot Activity Forecasting. This well-known benchmark problem [de Groot and Wurtz, (6)] was used to test the forecasting ability of the algorithm. Fifteen years' data were used to predict sunspot activity one year ahead. From the whole set of data (1700-1979 years) we formed 265 patterns, with first 200 used for training and the rest for testing. Each variable (all inputs and the output) was scaled into ±variance. The obtained formula was:

$$S_t = 9.3\cdot10^{-2} + 1.3\cdot S_{t-1} - 0.61\cdot S_{t-2} + 0.23\cdot S_{t-9} + \\ + 0.32\cdot S^2_{t-2} - 7\cdot10^{-2}\cdot S^2_{t-9} - 0.43\cdot S_{t-1}\cdot S_{t-2} + \\ + 6.9\cdot10^{-2}\cdot S_{t-1}\cdot S_{t-9} - 6.2\cdot10^{-2}\cdot S_{t-2}\cdot S_{t-5},$$

where S_t denotes sunspot activity in the year **t**.

In Fig. 1, the forecasting results on test set are presented together with actual sunspot activity values.

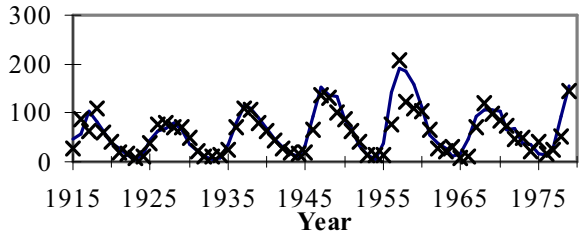

Figure 1. Sunspot activity values on test set: actual (line) and predicted (crosses).

Prediction of Deterministic Chaotic Time Series. The algorithm was applied to predict a well-known Logistic Parabola [de Groot and Wurtz, (6)], described by the formula

$$x_t = 1 - 2\cdot x^2_{t-1},$$

defined on the interval [-1..1]. Each pattern contained 10 successive values that were analysed by the algorithm to predict the value one step ahead. Both the train and test sets included 200 patterns. To make the task more difficult, the data were blurred by addition of white noise to the actual target values in the train set. GMDH algorithm successfully restored the correct formula with noise levels up to 30%.

Financial Prognosis. The algorithm was applied to a number of financial data to forecast different important market parameters. Typical values of **R** and **C** for this kind of problems are presented in Table 1.

Iris Classification. To perform classification of classical Fisher's iris data [Fisher, (7)], three GMDH models were created (one for each class). Activity of each model output served as measure of confidence that a pattern belongs to the corresponding class. Then the class of the pattern was determined by the maximal output activity among the three models. Use of this approach resulted in 2 misclassifications out of 75 train patterns and 2 misclassifications out of 75 test patterns.

15-bit Parity Problem. This problem was selected intentionally to demonstrate that learning of Boolean functions with many input variables, which is a challenge to neural networks approach [Kearns and Valiant, (8)], also cannot be solved by GMDH. The

possible reason why polynomial GMDH algorithm fails in this case is that it is difficult to create a representative set for training, as the modelled dependence is discontinuous everywhere, while we try to model it by a continuous function. Besides that, the Parity function obviously can hardly be expressed as a polynomial, as it contains modulus operation.

Table 1 - Estimations of GMDH performance.

Problem Name	R	C
Functional Approximation	0.91-0.98	0.96-0.99
Sunspot Activity	0.82	0.92
Logistic Parabola	0.99	1.00
Financial Prognosis	0.56-0.97	0.72-0.99
Iris Classification	0.79-0.98	0.90-0.99
15-bit Parity Problem	0.001	0.07

Discussion

The main conclusions that can be made are the following:

- GMDH is a powerful tool for mathematical modeling that can be used to solve a wide variety of different real-life problems.
- The most pronounced feature of GMDH is that it can choose the really significant input variables among dozens of these, thus actually reducing the dimension of the solved problem.
- One more useful property of the method is its ability to generate a formula directly expressing the algebraic dependence of the output on the inputs. This allows one to find out some important relations among the input variables: for example, whether there are cross-correlations between some variables, etc.
- GMDH can be also used for purposes of classification. However, it should not be used for approximation of complex Boolean logic functions.
- GMDH is hardly suitable for very large problems with a great number of inputs which are nearly equally significant. In this case, neural networks are probably the right choice.
- It is usually useful to perform scaling of all inputs and the output into ±variation. Solving problems with unnormalized variables is less computationally stable and may give worse results.
- The result (the quality of the obtained model) is strongly dependent on the type and coefficient (if any) of the selection criterion. These are the parameters that should be varied in the first turn.
- A new type of criterion was introduced in this work, which usually performs as well as or better than the other ones.
- It is very important to set the correct mode of optimization. It also may affect the result dramatically.

Conclusion

To summarize, GMDH is a powerful technique suitable to solve a wide range of different problems in mathematical modeling. Use of GMDH should be recommended to solve small and medium-sized problems with continuous output, especially to find out significant input variables and possible functional dependencies among them. GMDH should not be used for very large problems or for problems with discrete output.

References

1. Ivakhnenko A.G., 1971, 'Polynomial Theory of Complex Systems', IEEE Trans. Syst. Man & Cybern., SMC-1, 364-378.
2. Farlow S.J., ed., 1984, 'Self-Organizing Method in Modeling: GMDH Type algorithms', Statistics: Textbooks and Monographs, 54.
3. 3.Ivakhnenko A.G., Ivakhnenko G.A., and Muller J.A., 1994, 'Selforganization of Neuronets with Active Neurons', Pattern Recognition and Image Analysis, 4, 177-188.
4. Barron A.R., Xiao X., 1991, 'Multivariate Adaptive Regression Splines: Discussion', Ann. Stat., 19, 67-82.
5. Friedman J.H., 1991, 'Multivariate Adaptive Regression Splines', Ann. Stat., 19, 1-67.
6. De Groot C., Wurtz D., 1991, 'Analysis of Univariate Time Series with Connectionist Nets: A Case Study of Two Classical Examples', Neurocomputing, 3, 177-192.
7. Fisher R.A., 1936, 'The Use of Multiple Measurements in Taxonomic Problems', Ann. Eugenics, 7, 179-188.
8. Kearns M., Valiant L., 1989, 'Cryptographic Limitations on Learning Boolean Formulae and Finite Automata', Proc. 21st ACM Symp. Theory Comp., 433-444.