Modified GMDH Method and Models Quality Evaluation by Visualization

Pavel Kordík, Pavel Náplava, Miroslav Šnorek, Marko Genyk-Berezovskyj
Department of Computer Science and Engineering, CTU, FEE
Karlovo nám. 13, Prague, Czech Republic
kordikp@cs.felk.cvut.cz, naplava@fel.cvut.cz

KEYWORDS: Group Method of Data Handling (GMDH), Back-Propagation, Perceptron, Quasi-Newton method, Unconstrained Optimization

INTRODUCTION

The goal of this paper is to describe the Modified GMDH for inductive model generation and new visualization technique of model responses for the evaluation of model qualities.
Our method develops on a data set a group of models representing a complex system. The data set contains records of input and output variables of the system (description of system behaviour).
Inductive models are able to derive values of dependent output variable for all configurations of input variables (simulate the system behaviour).
The appropriate way for accessing the information enfolded in the model is to visualize its responses.

We introduce the visualization technique facilitating the evaluation of information included in the data set. It allows evaluating the quality of models too. Comparison of several models responses for the same values of input variables can indicate the quality of these responses. For any configuration of inputs, we can get not only the response (the estimated value of the output), but we would soon be able to determine the accuracy of this response too. This can grow into the big advantage of inductive models over the deductive ones.

Inductive Modeling

The most common modeling approach is the deductive one. The model is created using well known statistical methods such as Exploratory Data Analysis. The behaviour of modeled system is described using math equations.
We use the inductive approach that is not so spread. It is the data driven approach. The data nature is crucial for inductive modeling. It uses Artificial Intelligence methods for the automatic development of the black-box models on a data set.
These models simulate the behaviour of the system described by the data set.

MODIFIED GMDH

The Modified GMDH, which is being developed at our university, proceeds from GMDH introduced by Ivachknenko in 1966 [1]. It uses a data set to construct a model of a complex system. The model is represented by a network (see Figure 1). Layers of units transfer input signals to the output of the network. The coefficients of units transfer functions are estimated using the data set describing the modelled system.

Major modifications of the original GMDH are:

- The transfer function of the unit is of several types (linear, polynomial, logistic, etc.) and it can be provided by a perceptron network too. Each type of unit has its own learning algorithm for coefficients estimation. Which types of units are selected to make up the network depends just on the data nature.
- The number of unit inputs increases together with the depth of the unit in the network. Transfer functions of units reflect growing number of inputs.
- There exist interlayer connections in the network.
- The network construction process does not search all possible layouts of units[1]. It searches just the random subset of these layouts. The original GMDH produces one optimal model. Our method produces the group of models that are locally optimal, each for its specific subset of unit layouts.

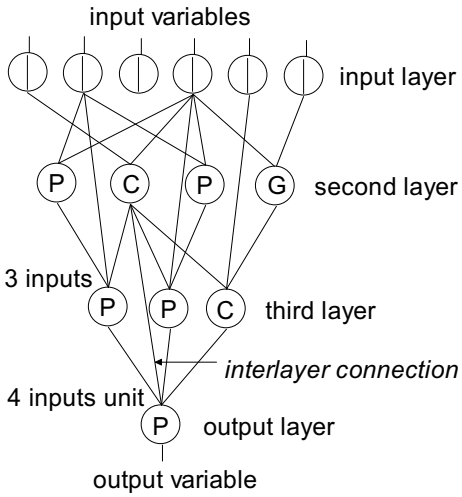


Figure 1. Modified GMDH network

The topology of the Modified GMDH neural network

The models (Modified GMDH neural networks) develop inductively on a data set. The data set includes vectors that consists of independent input variables (x1,x2,...,xn) and one dependent output variable y. During the learning process, forward multilayer neural network is developed in the following steps:

- The data set is split into the training and the testing data sets.
- In the input layer of the network n units with an elementary transfer function $y = x_i$ are constructed. These are used to provide values of independent variables from the learning set to following layers (hidden layers, output layer) of the network.
- When constructing a hidden layer an initial population of units is generated.

[1] When producing the initial population of a layer, it is not computationally maintainable to generate units for all combinations of inputs. It is possible for the original GMDH with invariable two inputs of the unit and no interlayer connections.

- There are several types of units in the initial population. Each type of unit has a unique transfer function and the learning algorithm. The type and inputs of the unit placed to the initial population of the layer is selected randomly.
- All units in the layer use its own learning algorithm to estimate coefficients using the learning set.
- The mean square error between the dependent variable y and the response of each unit is computed for all vectors from the testing set.
- Units are sorted out by the mean square error and just a few units with minimal error survive. The rest of units is deleted. This step guaranties that just the units with the best approximation ability are chosen.
- Further hidden layers are constructed while the weighted mean square error of units for the layer decreases.
- The output of the network is considered as the response of the best unit in the layer with the minimal weighted mean square error.

Units in the network can be of various types. It brings the data independence. Just the units of transfer function proper to data set nature survive.
We defined the following types:

- **Unit with a linear transfer function**
  According to the growing complexity scheme, the transfer function has the following structure:

$$y = a_1 x_1 + a_2 x_2 + ... + a_n x_n + a_{n+1},$$

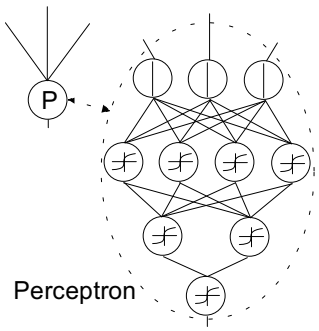
Where y is the output of the unit,
$x_1, x_2, ..., x_n$ are inputs to the unit,
$a_1, a_2, ..., a_n, a_{n+1}$ are coefficients computed by Gauss-Jordan Method or estimated by an optimisation method using whole learning set,
n is the number of the layer being created.

- **Unit with a polynomial transfer function**

$$y = \sum_{i=1}^{m} \left(a_i \prod_{j=1}^{n} x_j^{k_{i,j}} \right) + a_{m+1},$$

where y, x_i, a_i, n have the same meaning as in the previous type of the unit,
$k_{i,j} \in (0,1,... n)$ is random integer exponent (growing complexity),
$m = n + 1$ is experimentally derived number of terms. Coefficients are estimated by using unconstrained optimisation routines [2].

- **Unit with a perceptron structure**
  The perceptron network as a unit of the GMDH network has the following structure: n input neurons, small random number of hidden layers and neurons in each layer and one neuron in the output layer. The Back-Propagation Algorithm is used for setting up the weights of the perceptron network.



- **The unit with the logistic transfer function**
  This unit attempt to approximate the input-output relationship using the nonlinear logistic transfer function.

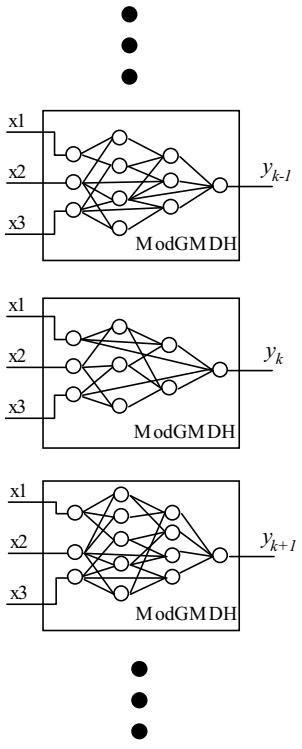
$$y = \xi\left(\sum_{i=1}^{n} a_i x_i + \Theta \right) \qquad \xi(\varphi) = \frac{1}{1 + e^{-\gamma\varphi}}$$

where y is the output of the unit,
a_i is the coefficient of the i-th input,
x_i is the value of the i-th input,
Θ is the bias of the unit,
ξ is the sigmoid function,
γ is the sensitivity coefficient of sigmoid function.

Coefficients $a_1, a_2, ..., a_{n+1}$ are estimated using the Quasi-Newton method.

Detailed information about the Modified GMDH can be found in [4].

The group of models and the quality evaluation



The Modified GMDH generates a group of models on a single training data set. The random processes influence the construction procedure. Weights and coefficients of units are randomly initialized. Transfer functions of many units types are defined pseudo-randomly when the unit is initialized. Inputs for units are selected pseudo-randomly, as well. It results in the fact, that the topology of models developed on the same training data set differs.
The question is how to determine the quality of models in the group.

We can compare the mean square error of models responses on the testing data set. To compute the mean square error we can use the following function. The testing set contains T $([x_1, x_2, ..., x_n]; o)$ vectors, with the meaning ([input_vector];relevant_output). When we put the input vector to the input of model, we get the response y.

The mean square error summarises the square deviation of the response from the desired value o for all vectors from the testing set:

$$\partial^2 = \frac{1}{T}\sum_{i=1}^{T}(y_i - o_i)^2$$

where y_i is the response of the model to the i-th input vector,

o_i is the output part of the i-th testing vector (desired response),

T is the number of testing vectors,

∂^2 is the mean square error of the model.

When we compute mean square error of all models in the group, we can usually find a few models with the error much bigger than the rest of the group. The quality of these models is lower (in the area of the input space defined by the presence of testing vectors). These models were not successfully generated and should be eliminated from the group.
The remaining models have always approximately the same mean square error.

To check if models in the group are equivalent, we designed method facilitating visualization of their responses.

VISUALIZATION OF MODEL RESPONSES

By visualization of model responses we can access the information abstracted by the model from a data set. The easiest way to visualize how the model approximates the system is to change values of input variables and record the output of the network.

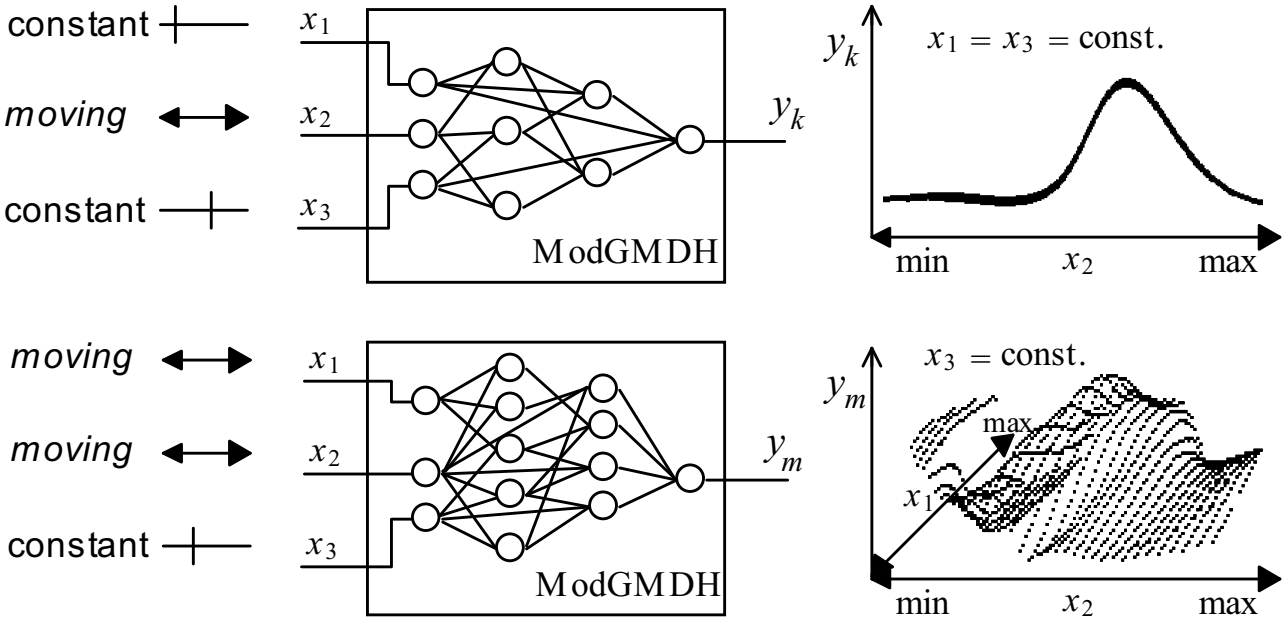


Figure 2. The principle of visualization of Modified GMDH model responses

The basic principle of visualization techniques is illustrated on the Figure 2. When we vary just one input variable whereas others stay constant, we can plot a curve. The curve shows us the influence of selected input variable to the output variable in the configuration specified by the others input variables. If we change the input configuration, the shape of the curve changes often too.
If we vary two of the input variables whereas others stay constant, we can plot a surface. The surface represents the relationship between two input variables and the response of the model in the configuration defined by the constant inputs.

The intersection in one dimension of the input space (curve)

If we assume input variables as dimensions of the input space, we can display the input vectors as points in the input space.

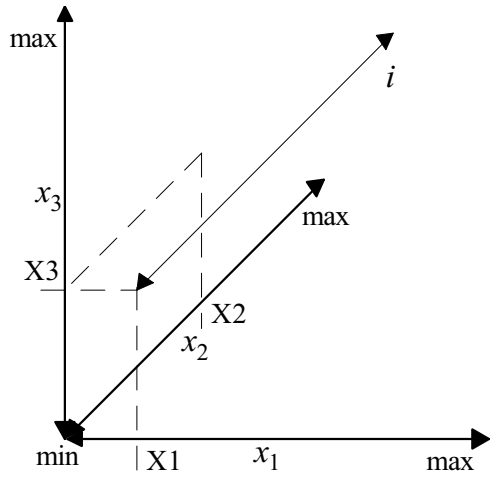


Figure 3. The intersection of the input space in one dimension.

For three independent input variables we get the three dimensional input space (Figure 3). The input vectors V located on the intersection *i* of the input space, are used as the inputs for the model. The output of the model plots a curve. The curve expresses the relationship between input variable x_2 and the dependent output variable y for the configuration x_1=X1, x_3=X3.

V[X1,*i*,X3]; X1,X3 = invariable, $i \in \langle x_2\text{min}, x_2\text{max} \rangle$;
$y = f(x_2)$; x_1=X1, x_3=X3;

The intersection in two dimensions of the input space (surface)

Just as in the previous technique, we can visualize responses for mutual combinations of two independent input variables (Figure 4). For the plane intersection *ij* of the input space the shape of the surface expresses the relationship between independent input variables x_1, x_2 and the dependent output variable y in the configuration x_3=X3.

V[j,i,X3]; X3 = invariable, $j \in \langle x_1 \min, x_1 \max \rangle$, $i \in \langle x_2 \min, x_2 \max \rangle$;
$y = f(x_1, x_2)$; x_3=X3;

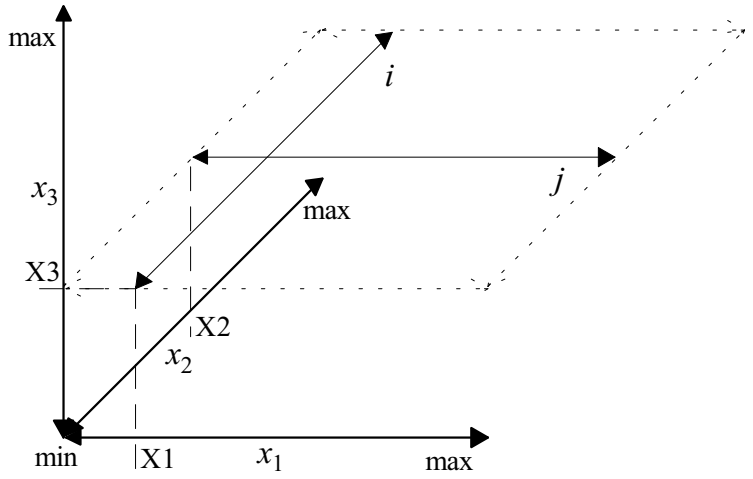


Figure 4. The plane intersection of the input space in two dimensions

To be able to see, how models approximate the training data, we need to find a projection of training vectors to the graph of the input-output relationship. We developed the following method.

Projection of data vectors into the Input-Output relationship graph

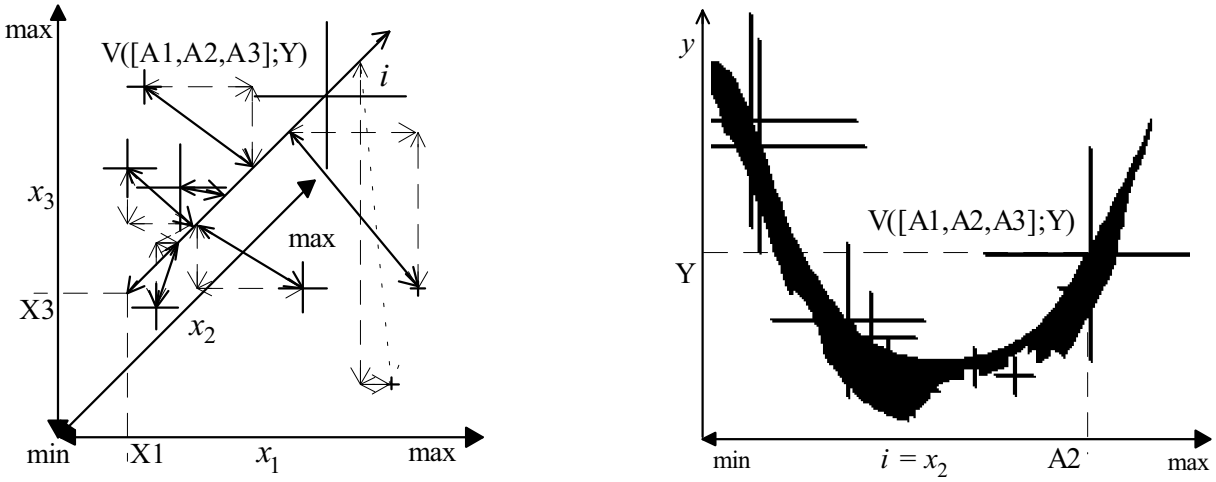


Figure 5. The projection of data vectors into the IO relationship graph

Each data vector consists of two parts. The first is the input vector and the second is the requested output for this input vector.
We plot the crosses representing data vectors to the graph (Figure 5). The position of the cross is given by the value of the vector for the dimension of intersection (x_2) and by the output value [A2,Y].
We compute the Euclid distance of input part of each data vector V from the axis of the input space intersection.

The size of the cross that represents the data vector is inversely related to its distance from the axis of intersection.

$$Size = \frac{1}{\max(H, Dist)}, H > 0,\ Dist = \sqrt{(\text{A1-X1})^2 + (\text{A3} - \text{X3})^2},$$

where
- H is small number to limit the cross size,
- $Size$ is the size of the cross in the graph,
- $Dist$ is the Euclid distance of the vector from axis of intersection,
- A1,..,A3 are values of the data vector in input dimensions,
- X1,..,X3 are the constant values of model inputs (input configuration).

The upper part of the curve in the graph (Figure 5) shows responses of the model for input vectors located on the axis of intersection. The thickness of the curve represents the density of data vectors in the input space. The more vectors are present and in defined neighborhood, the higher the density is.
You can consider the quality of the model simply by looking at the IO relationship graph and checking how it approximates the training or testing vectors.

EXPERIMENTS ON THE ARTIFICIAL DATA SET

First experiments were performed on an artificial data set to show the functionality of our method. We generated small data set (40 data vectors) describing the artificial system. The output variable depends on input variables in conformity with the following equation:

$$y = \frac{\sinh(x_1 - x_2) + x_1^2 (x_2 - 0.5)^2}{2},$$

where y is the dependent output variable, x_1, x_2 are input variables.

Our Modified GMDH method developed a group of models (networks) on the data set. Models have different inner structure but the responses are almost the same.

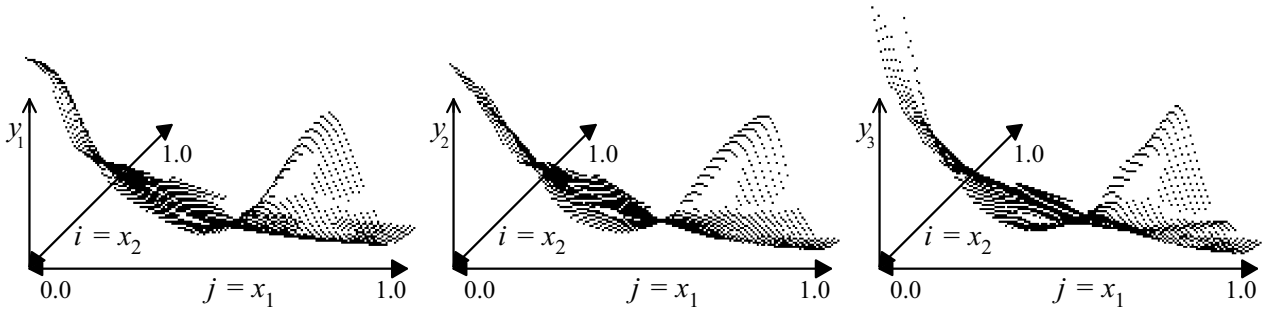


Figure 6. Group of models developed on the Artificial data set

Training data vectors are distributed uniformly in the area x_1, $x_2 \in \langle 0,1 \rangle$. Whereas in the center of the area we can notice a good match of models, at the borders there is remarkable difference in responses of models.

The IO relationship graph for the group of models

We plot responses of models in one graph, to be able to compare how these differ for particular input vectors.
To explore responses of models in areas where data vectors are defined, on the borders of such areas and outside where no training vectors are present, we prolong the intersection of the input space.

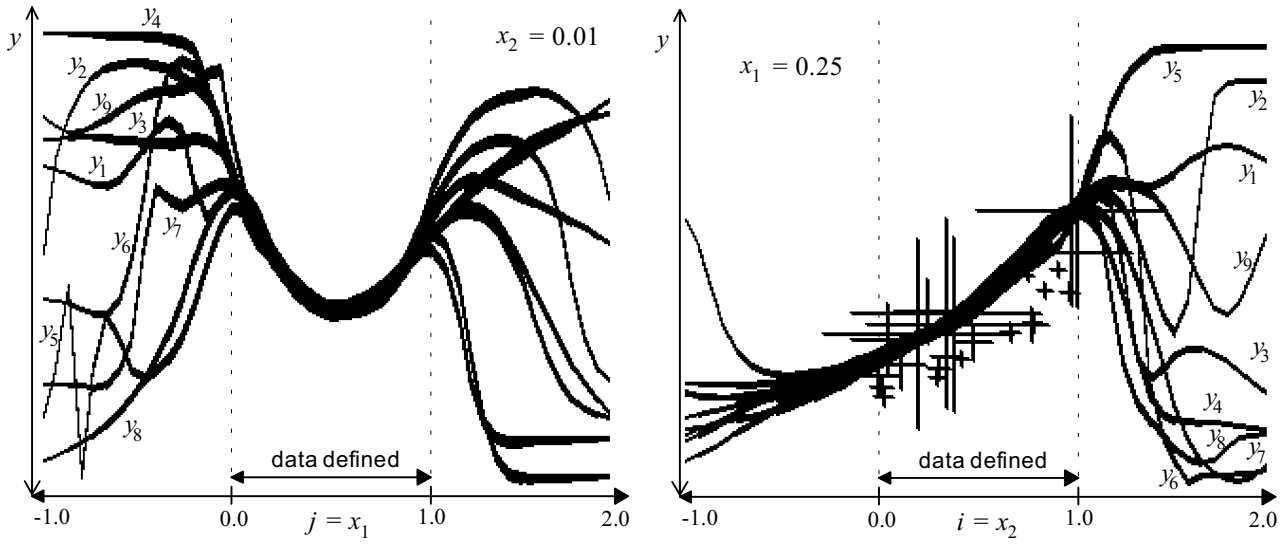


Figure 7. Responses of nine Modified GMDH networks developed on the Artificial data set

When models are randomly initialised, responses of these models are chaotic before the learning process begins. The iterative learning process is shifting their responses towards to training data outputs. After the learning phase, models approximate the training data.
The learning iteration process affects just the areas where some training vectors are present. With growing distance from the vectors the influence of the learning process declines. Responses of models stay chaotic within the areas far from training data vectors.

THE QUALITY EVALUATION OF THE MODEL RESPONSES

When you look at the Figure 7, you can find areas where models have compromise response and the areas where the models responses considerably differ. The areas of the compromise response correspond to the areas where some training vectors are present.

We found out that if there is a satisfactory density of training vectors with suitable distribution (uniform) in the input space, there would exists a compromise response of models in the output space. On the other hand when there are not enough vectors in the input space, the responses would differ.

This fact can be used to estimate the quality of the models responses.

The quality of models for any configuration of input variables depends on the models responses dispersion for this input vector.

We demonstrate in an example how important information carries the dispersion of models responses.

Imagine we have developed an inductive or deductive model on a data set. Later on we would like to get response for an input vector. How we can determine the quality of the model response for this input vector?
If we have training data, we can compute the distance of the input vector from training vectors. Using this distance we can estimate the quality. If the distance is small the response for the input vector will be good, if it were big the response would have been probably bad.

When we use the Modified GMDH to generate a group of models, we do not need the training data set any more. If models responses for the input vector considerably differ, we can say the quality of models is bad for this input vector. If responses match, models are of good quality at this area of the input space.

This approach has big advantage against computation of the vector distance from the training data. It takes into consideration the importance of the particular input variables in the neighbourhood of the input vector.

If training vectors differ a lot from the input vector in the value of an unimportant input variable, the quality of the model is still high.
On the other hand the small distance of the input vector from training data in very important dimension can entail the models quality slump.

THE REAL-WORLD APPLICATION: MANDARIN TREE WATER CONSUMPTION

The main object of our experiments is the data set (provided by the Hort Research, New Zealand) describing water consumption of a mandarin tree.
The mandarin tree is the complex system influenced by many input variables (water, temperature, sunshine, humidity of the air, etc.).
Our data set consists of measurements of these input variables and one output variable – the water consumption of the tree. It describes how much water the tree needs in specific conditions.

We used 2500 training vectors (11 input variables, 1 output variable) to generate group of models using Modified GMHD method.
From this group of eight models we eliminated one because of to high mean square error on the testing set (1500 vectors).
Seven models that left in the group had approximately equal mean square error.

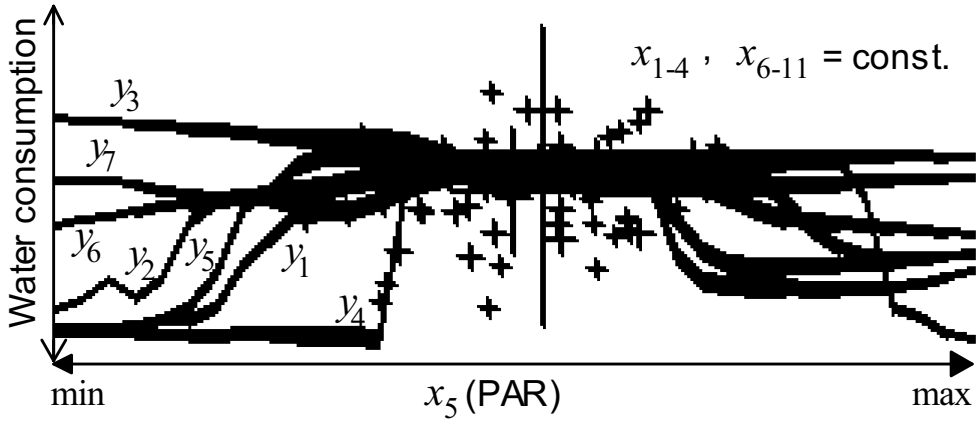


Figure 8. The dependence of tree water consumption on the PAR variable

The IO relationship graph (Figure 8) shows responses of these seven models. The intersection of the input space is parallel to the dimension x_5 - input variable PAR.

The variable PAR, as the majority of natural variables, has the normal distribution. This distribution is not very suitable for the shape definition because the values concentrate in a small dense cloud. If we expose the tree to very different conditions, we would achieve the cloud to get bigger and the shape of curves in the graph to be better defined.

When we look at the figure, all we can say is that there is an interval defined by the data cloud, where responses of models match. Unfortunately the interval is too small and we are not able to determine how the change of the PAR variable influence the water consumption for the conditions specified by the others input variables.

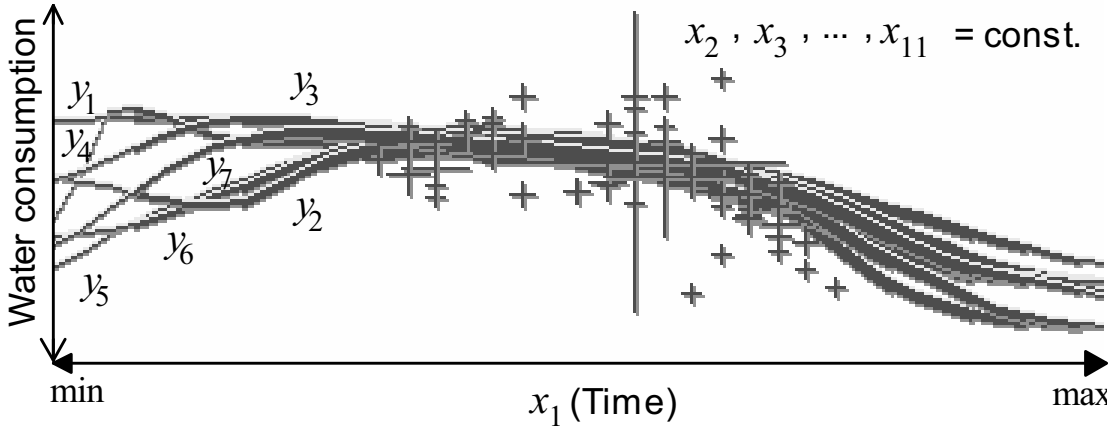


Figure 9. The dependence of tree water consumption on the Time variable

The Figure 9 shows the relationship graph analogous to that in the previous figure. The PAR variable is replaced by the variable Time. The variable Time is monotonous, does not influence the tree and therefore should not be considered as the input variable for the model [3]. Curves from the figure have no physical meaning. The reason why we present the graph is the uniform distribution of the Time variable.

You can see the shape of the relationship is much better defined than for variable PAR with normal distribution.

Theoretically, if have some input variable we can control (fertiliser, watering, chemical, etc.), this variable can be included in the data set with the uniform distribution (the use of various quantities when treating the tree). Then the IO relationship graph can tell us, what influence would have increasing or decreasing of some input variable (watering) on the output (water consumption) in any conditions (configuration of other input variables). This can be the method facilitating the search for the optimal treatment of the tree.

IMPLEMENTATION OF THE IDEAS DESCRIBED ABOVE

The java application has been created. It allows the real-time simulation of the group of Modified GMDH models. You can for example drag the scrollbar representing some input variable and watch the visualization of models responses. New types of unit and learning algorithm can be easily integrated to the system. The application implements powerful configuration option.

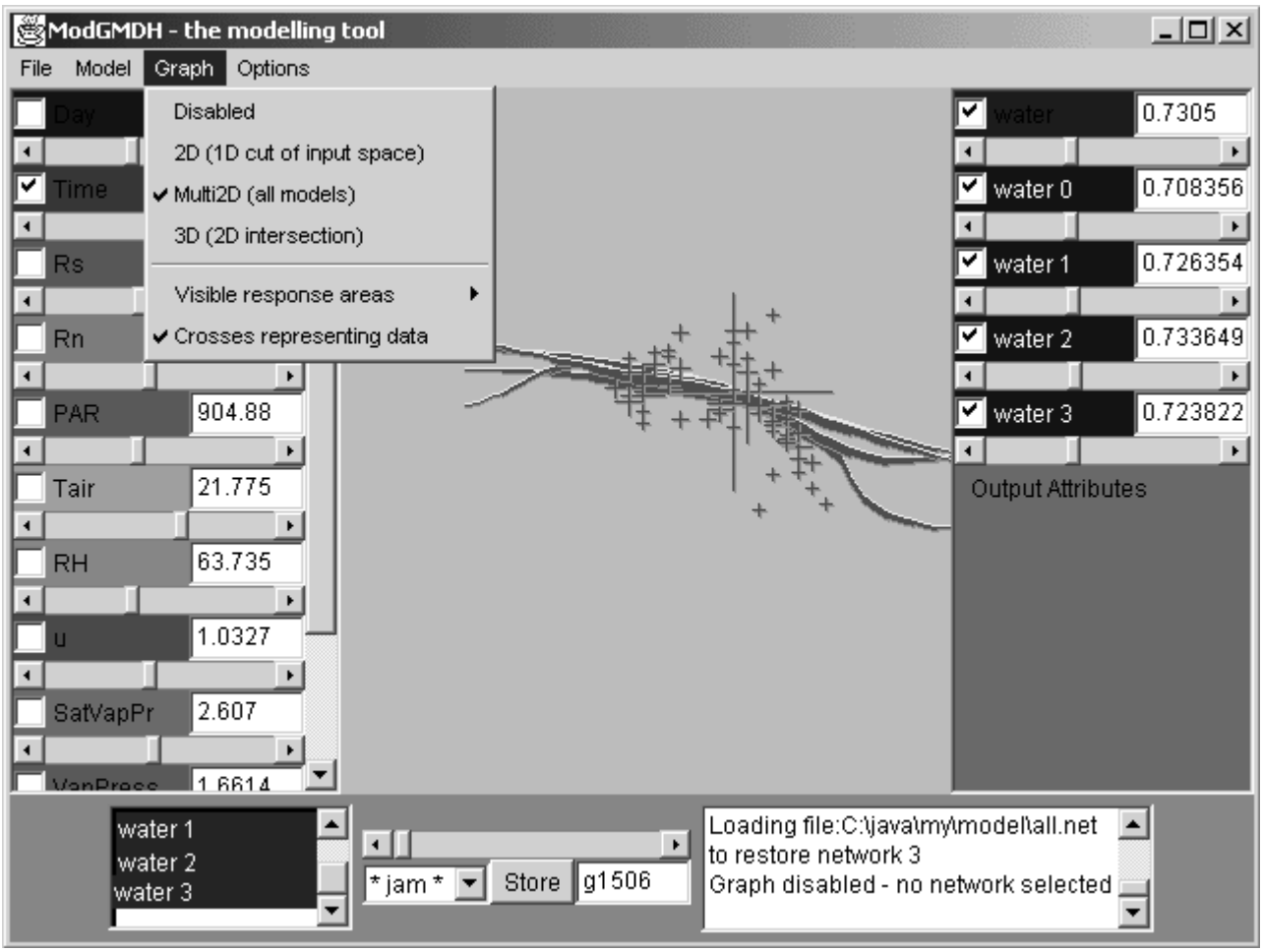


Figure 10. The screenshot of the ModGMDH java application

CONCLUSION

We presented the modified GMDH method that can be used to generate group of models of a system. The responses of these models can be visualized using techniques introduced above.
The results of visualization showed to be profitable for the knowledge mining and for models quality evaluation too.
Our next goal is to explore the relationship between the dispersion in models responses for the specified input vector and the models quality (accuracy) in this area of the input space.

REFERENCE

[1] Madala, Ivakhnenko: Inductive Learning Algorithm for Complex System Modelling; 1994, CRC Press, Boca Raton
[2] R.B. Schnabel, J.E. Koontz, and B.E. Weiss: A Modular System of Algorithms for Unconstrained Minimization, Report CU-CS-240-82, Comp. Sci. Dept., University of Colorado at Boulder, 1982.
[3] Dorian Pile: Data Preparation for Data Mining; 1999, Kaufman publishing
[4] Kordík, Náplava, Šnorek, Genyk-Berezovskij: The Modified GMDH Method Applied to Model Complex Systems; proceedings of ICIM'2002 Conference, Ukraine, Lviv
[5] www.gmdh.net