APPLICATIONS OF NEUROFUZZY TRAINING ALGORITHMS TO SIMULATION METAMODELLING

Galina V. Merkuryeva and Liana E. Napalkova
Department of Modelling and Simulation
Riga Technical University
1 Kalku Street
LV – 1658, Riga, Latvia
E-mail: gm@itl.rtu.lv, liana@factm.com

KEYWORDS

Simulation metamodelling, NeuroFuzzy training, approximation error.

ABSTRACT

NeuroFuzzy learning algorithm for simulation metamodelling is described in the paper. Here, the simulation experimental data are used to train a fuzzy neural network-based simulation metamodel and to generate a set of relevant decision rules. Regression type model is applied to define the structure of the metamodel training set and essential decrease of an approximation error of simulation output data is received. The research results in the paper are illustrated with a range of experiments performed.

INTRODUCTION

Simulation metamodelling is based on substitution of the current simulation model by its metamodel in order to perform further model manipulations such as conducting sensitivity analysis of the model, testing hypotheses regarding the real system, predicting of the model behaviour and optimisation of its parameters, etc. Usually, these actions lead to decreasing approximation accuracy of the investigated process necessary to make relevant conclusions and effective decisions.

Traditionally, statistical metamodels, such as regression ones, are used to approximate simulation model input-output relationships. Mostly, these models are applied to understand the behaviour of the simulation model and to perform its sensitivity analysis. Nevertheless, regressions models are not usable for extrapolation purpose. In general, regression models need a priori knowledge to define the kind of an appropriate function, and in many applications the resulting simulation metamodels don't ensure high approximation accuracy. In this sense, machine learning techniques, such as artificial neural network (ANN) - based models, could be more preferable. Moreover, learning techniques based on fuzzy computing well suit to deal with 'stochastic' data as they allow modelling soft data points.

ANN-based models provide universal means for data approximation and, at least theoretically, enable to discover relationships in simulation experiments with any accuracy degree. Nevertheless, in practice, these models require a great number of simulation experimental data. To solve this problem optimisation of the training set structure for generating a neural network is proposed in the paper. To define the training set structure dynamic or static regression type models are used. General features of this approach are described in the next section.

ANN-BASED APPROXIMATIONS

Let's consider a set of K data points from simulation experiments specified by a set of input variables, or factors $U = \{u_{i,k}\}, i = \overline{1,N}$, and output variable y_k, where N is a number of input variables, and $k = \overline{1,K}$.

Let's suppose, that causal relationships between simulation input and output variables are represented by simulation metamodel, or approximation f, as:

$$\hat{y}_k = f(u_{i,k}, \delta), \tag{1}$$

where δ represents an approximation error defined by an average linear error:

$$\delta = \frac{\sum_{k=1}^{K} \left| y_k - \hat{y}_k \right|}{K}. \tag{2}$$

In case of ANN-based approximations in (1), a set of K experimental points present a metamodel training set.

To decrease an approximation error δ, the training set structure could be transformed into another one (see, Figure 1) by including variables (e.g. lags, quadratic or interaction factors) that could be derived from the statistical analysis of simulation experimental data. As a result, ANN-based simulation metamodel f' with an approximation error $\delta' < \delta$ could be received:

Proceedings 19th European Conference on Modelling and Simulation
Yuri Merkuryev, Richard Zobel, Eugène Kerckhoffs © ECMS, 2005
ISBN 1-84233-112-4 (Set) / ISBN 1-84233-113-2 (CD)

$$\hat{y}_{k'} = f'(u_{i',k'}, \delta'), \qquad (3)$$

where $u_{i',k'} \in U'$, $U' \subseteq U \cup L$ is the modified training set structure, $L = \{l_{j,k'}\}$ is a set of derived variables, $j = \overline{1, M}$, $i' = \overline{1, N'}$, $N' \le M+N$, $k' = \overline{1, K'}$, $K' \le K$, and M is a number of derived variables, N' is a total number of variables selected.

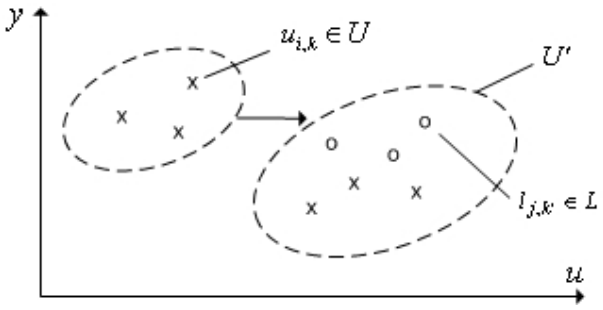

Figure 1: Transformation of the metamodel training set structure

The above-mentioned ANN–based models allow describing complex non-linear functional dependencies between simulation inputs and outputs in automatic mode. Both general and specialized ANN networks include training algorithms to set-up their parameters. However, general ANN networks cannot describe causal relationships in the simulation model because generated ANN model operates as "Black Box" also being poor from a standpoint of its verifiability. In this sense, specialized models, such as Adaptive-Network-based Fuzzy Inference System (ANFIS) could be more preferable for simulation metamodelling. It could be used to generate the decision rules in order to develop metamodelling knowledge base. In this case, simulation generated data are used to inquire and represent knowledge in the form of production rules as:

$$\begin{aligned} &\text{IF} \quad (u_1 \text{ is } A_1) \wedge (u_2 \text{ is } B_1) \\ &\text{THEN} \quad z_1 = p_1 u_1 + q_1 u_2 + w_1 \end{aligned} \qquad (4)$$

where A_1, B_1 are linguistic terms, p_1, q_1, w_1 are coefficients of linear equation, and z_1 is a real number.

To generate fuzzy productions (4), Sugeno inference multi-layer algorithm (Fuller 1995) is applied.

Layer 1 defines the degree to which the given input satisfies the linguistic terms, such as A_1, A_V, D_1, D_V.

Layer 2 computes the firing levels of the rules as:

$$\alpha_1 = A_1(u_1) \wedge D_1(u_N), \ \ldots, \ \alpha_R = A_V(u_1) \wedge D_V(u_N),$$

where V is a number of linguistic terms and R is a number of rules in a rule base.

Layer 3 normalizes these firing levels by the following formulas:

$$\beta_1 = \frac{\alpha_1}{\alpha_1 + \alpha_R}, \quad \beta_R = \frac{\alpha_R}{\alpha_1 + \alpha_R}.$$

Layer 4 performs the production of the normalized firing level and the individual rule output.

At last, layer 5 computes the overall output of the system:

$$z_0 = \sum_{r=1}^{R} \beta_r z_r^* .$$

NEUROFUZZY TRAINING ALGORITHM

To generate the training data set structure for ANN-based simulation metamodel, a dynamic regression is applied.

Dynamic regression models, generally called as Autoregressive Distributed Lag or ADL models (Pindyck and Rubinfeld, 1998), can be presented by the following equation:

$$y_t = \vartheta_0 + \sum_{j=1}^{P} \vartheta_j y_{t-j} + \sum_{i=1}^{N} \sum_{j=0}^{R} \varphi_{ij} u_{i,t-j} + \varepsilon_t \qquad (5)$$

where y_{t-j} is a vector of values of an output variable observed at the time $(t\text{-}j)$, ϑ_j, φ_{ij} are regression coefficients, N is a number of explanatory variables, P and R are lag lengths, and ε_t is an error term. Let note, that comparing with static regression models dynamic ones follow changes of the simulation model behaviour on time and enable to increase approximation accuracy of the underlying dependencies.

The corresponding NeuroFuzzy training (NTF) algorithm (Merkuryeva and Napalkova 2004) aimed to build ANN-based simulation metamodel includes the following steps:

1. ADL model identification.
2. Generating the training set structure according to ADL model.
3. Generating NeuroFuzzy metamodel structure by fuzzy production rule set.
4. Training simulation metamodel.
5. Simulation metamodel validation.

Step 1. ADL model is developed from simulation experimental data.

This step is generally splitted into two actions: 1) analysis of residuals for static regression model that focuses on the test of first-order serial correlation

(Heuchemer 2000), in this action evaluation of Durbin-Watson statistics is performed; 2) transition from a static regression to a dynamic one when first-order serial correlation occurs.

To correct the initial static model lagged variables are added. To detect a lag length an autocorrelation plot or a partial autocorrelation plot is used. To test a higher-order serial correlation an analysis of Box-Ljung statistics is performed. In order to exclude insignificant variables, the regression model is evaluated by R-squared, F-statistics and t-statistics.

Let's assume that residuals are described by the second-order AR(2) process. The static regression model with an AR(2) error could be transformed into the dynamic ADL model by adding two lags of y_t and $u_{i,t}$, $i = \overline{1, N}$:

$$y_t = \vartheta_0 + \sum_{j=1}^{2} \vartheta_j y_{t-j} + \sum_{i=1}^{N} \sum_{j=0}^{2} \varphi_{ij} u_{i,t-j} + \varepsilon_t . \qquad (6)$$

Step 2. Training set structure U' is defined and statistical data for variables included in the training set are generated. Training set structure for the ADL model (6) is presented as follows:

$$\{< y_t, y_{t-1}, y_{t-2}, u_{1,t-1}, u_{1,t-2}, ..., u_{N,t-1}, u_{N,t-2} >\}. \qquad (7)$$

Step 3. To generate ANN-based fuzzy metamodel, the following tasks are performed: 1) the type and number of membership functions assigning to each input and output variable are defined; 2) the form of production rules (4) is defined. The set K' of data points for each input $u_{i',k'}$ and output variable $y_{k'}$ is divided into clusters. The number of clusters corresponds to the number of terms for each variable. Numerical values of the rules consequent and premise parameters are estimated at the next step.

Step 4. The ANFIS training algorithm is used to identify the rules consequent and premise parameters. This algorithm adjusts the consequent parameters in a forward pass and the premise parameters in a backward pass by using least-squares method and gradient descent method, respectively (Figure 2).

Step 5. The approximation error to validate the resulting metamodel is estimated using equation (2). In a case of $\delta' \geq \delta$, the structure of the training set U' is corrected.

Thus, ANN-based simulation metamodel approximates the target functional dependency with a piece-wise linear function and could be described by a set of production rules like:

$$\text{IF } g(u_{i'} \text{ is } A_v) \text{ THEN } \hat{y} = f'(u_{i'}) , \qquad (8)$$

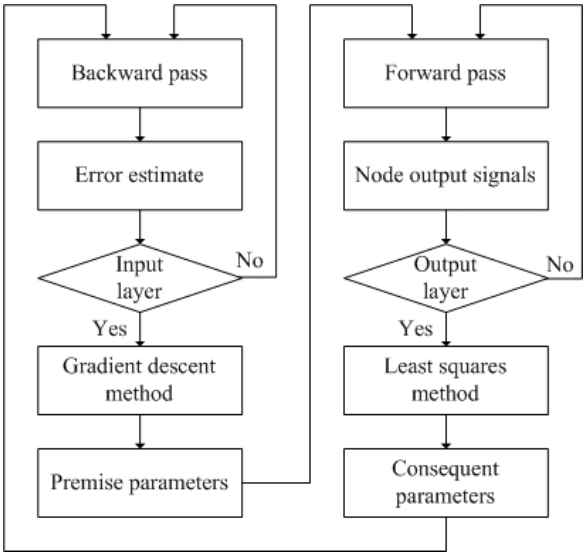

Figure 2: Adjusting production rules parameters

These rules define linear regions for a metamodel, where *g* is a logical function that combines propositions of the conditions; and *f* is a function of simulation input variables $u_{i'} \in U'$ that produces a simulation metamodel output variable. The next section presents application and testing results of NTF algorithm.

NFT ALGORITHM TESTING

The simulation study in (Merkuryeva and Shires 2004) is aimed to improve business operations in the planning department in a medium-sized company. The high-level business/manufacturing simulation model is developed to analyse processing of incoming inquiries of two different types and planning production orders already confirmed by customers. The following controllable variables are defined in this model: 1) inquiries processing time, 2) planning time for orders confirmed. The time between arrivals of inquiries, customer response time, the probability an inquiry becoming confirmed or becoming an order, and order processing time at production stages are regarded as environmental variables. The following simulation output variables are analysed: 1) average lead-time, 2) total revenue, 3) utilization of planners, etc.

To uncover relationships between the enquiries processing time, the orders planning time and an average lead-time, ANN-based simulation metamodels without and with NTF algorithm were built and tested. Two tests for two types of inquiries are performed to illustrate preferability of application NTF algorithm.

In the *Test 1* performed for the first type of inquiries (i.e. Pharmaceutical ones) the training set structure U consists of two input variables $u_{1,\,k}$, $u_{2,\,k}$ and output variable y_k , denoting inquiry processing time, order planning time, and an average lead-time in the system,

correspondingly. Each input variable is described by four triangular membership functions (Figure 3).

First, to train ANN-based metamodel without NTF algorithm, 17 data points $\{<y_k, u_{1,k}, u_{2,k}>\}$ generated from simulation experiments are used. As a result, 16 production rules, like (9), are derived that approximate the input-output behaviour of the basic simulation model:

$$\text{IF} \quad (u_1 \text{ is } A_2) \wedge (u_2 \text{ is } B_3)$$
$$\text{THEN} \quad \hat{y}_{4'} = -1549u_1 + 1037u_2 + 98.38 \qquad (9)$$

where A_2, B_3 presents specific membership functions and coefficients present consequent parameters.

In general, the potential number of production rules that represent metamodelling knowledge base is equal V^N, where N, V is a number of inputs variables and membership functions for each input, correspondingly. Let's note, that some of that rules could have empty THEN part.

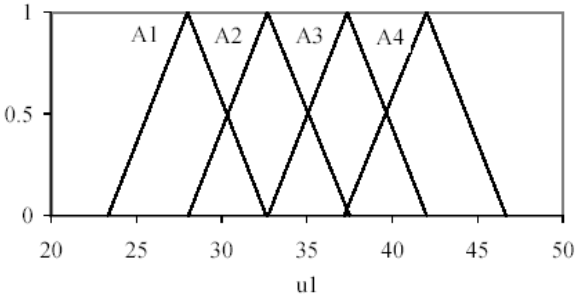

Figure 3: Triangular membership functions for input variable 'inquiry processing time'

Training process required 100 iterations. The estimate of approximation error is equal to 1.388.

Second, to build ANN-based metamodel with NFT algorithm, the regression-correlation analysis is performed. The following regression equation is received:

$$\hat{y} = 9277.03 - 21.05 * u_1 + 4.83 * u_2 + 0.62 * (u_1)^2 + 0.41 * u_1 * u_2$$

From the last equation the training set structure U' is defined and it includes four input variables as follows:

$$\{<y_{k'}, u_{1',k'}, u_{2',k'}, u_{3',k'}, u_{4',k'}>\}, \qquad (10)$$

where $u_{1',k'} = u_{1,k}$, $u_{2',k'} = u_{2,k}$, $u_{3',k'} = (u_{1,k})^2$, $u_{4',k'} = u_{1,k} \times u_{2,k}$ and $K' = K$.

Number of triangular membership functions assigned to each input variable is equal to 4. Training process consists of 100 iterations or epochs (Figure 4). As a result, 256 production rules are generated that approximate the input-output behaviour of the basic simulation model and represent a new metamodelling knowledge base. The sample production rule is given below:

$$\text{IF } (u_{1'} \text{ is } A_2) \wedge (u_{2'} \text{ is } B_2) \wedge (u_{3'} \text{ is } C_2) \wedge \wedge (u_{4'} \text{ is } D_1) \text{ THEN}$$
$$\hat{y}_{4'} = 0.1004u_{1'} + 0.2574u_{2'} + 3.247u_{3'} + +8.527u_{4'} + 0.00312, \qquad (11)$$

where A_2, B_2, C_2, D_1 are inputs membership functions, and coefficients present consequent parameters.

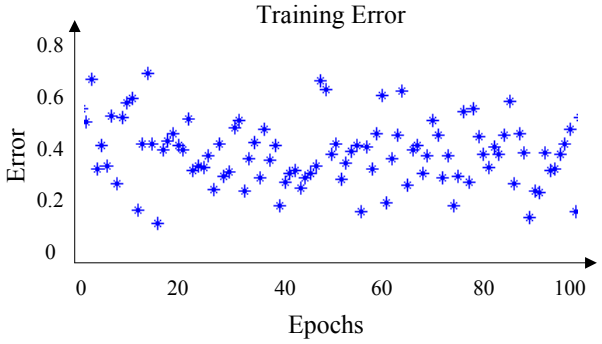

Figure 4: ANFIS training process (Test 1)

ANFIS testing results while comparing training data with metamodelling output data are illustrated in Figure 5. In this case, the estimate of approximation error δ' according to formula (2) is equal to 0.4489. It means that $\delta' < \delta$.

In the *Test 2* the results of simulation experiments for the second type of inquiries (i.e. Personal care inquiries) are used. ANN-based simulation metamodels without and with NTF algorithm are built and tested.

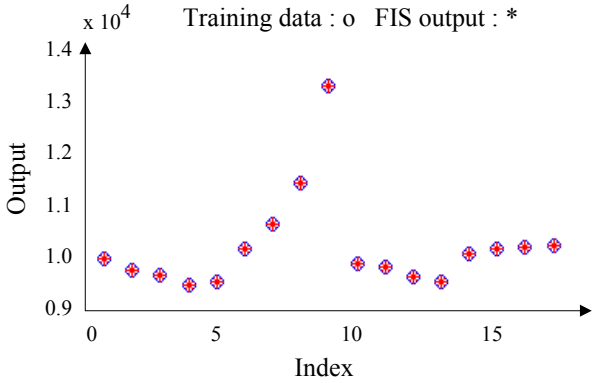

Figure 5. ANFIS testing results

Training set structure for the first metamodel built without NTF algorithm corresponds to one in the first test. Training set structure for the metamodel built with

NFT algorithm includes additional derived variables. Each input variable is described by three triangular membership functions.

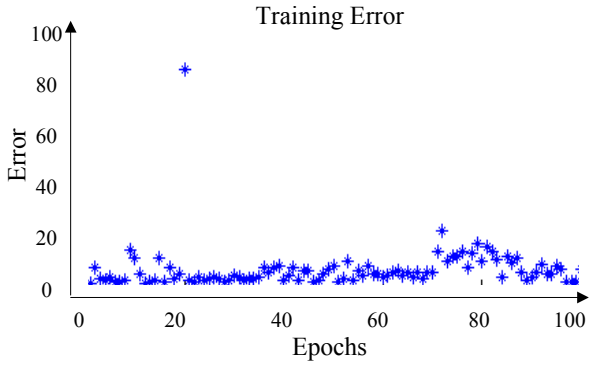

Figure 6: ANFIS training process (Test 2)

After the training process in 100 iterations (Figure 6) corresponding approximation errors δ and δ' are estimated and they are equal to 12.038 and 2.644, respectively, that gives $\delta' < \delta$. Corresponding ANFIS testing results are presented in Figure 7.

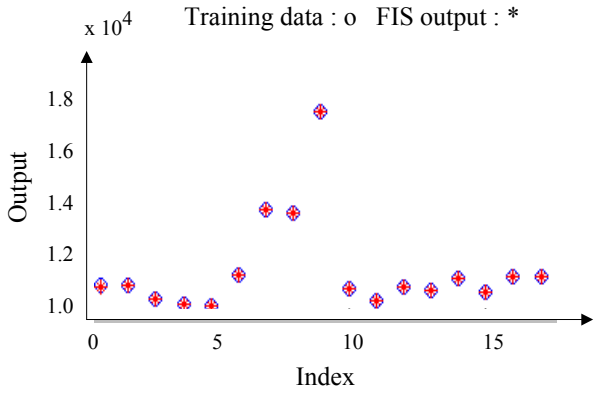

Figure 7. ANFIS testing results

Table 1: Comparative testing results

	Metamodel built without NFT (δ)	Metamodel built with NFT algorithm (δ')
Test 1	1.388	0.449
Test 2	12.038	2.644

The results of above tests (Table 1) shows that, ANN-based simulation metamodels built with NeuroFuzzy training algorithm provide lower approximation error of input-output relations in basic simulation models.

CONCLUSIONS

The developed NFT algorithm for simulation metamodelling is described in the paper. It is based on application of regression models to modify the training set structure of ANN-based fuzzy metamodels. It is indicated that application of NTF algorithm leads to increasing the degree, to which metamodel accurately approximates underlying dependencies in the basic simulation model. Moreover, this algorithm doesn't require expensive and time-consuming experiments.

REFERENCES

Esbensen, K. 1994. "Multivariate analysis in practice". In: CAMO, Trondhein.

Fuller, R. 1995. "Neural Fuzzy Systems". In: Åbo Academy University.

Heuchemer, S. 2000. "Analyse eines Regressionsmodells". In: Institut für Allgemeine Wirtschaftsforschung.

Merkuryeva, G. and Napalkova, L. 2004. "Applications of NeuroFuzzy training algorithms to analysis of business processes". In: *Scientific proceedings of Riga Technical University*. Riga, 141-148.

Merkuryeva, G. and Shires, N. 2004. "SIM-SERV Case study: Simulation-based production scheduling and capacity optimisation". In: *Scientific proceedings of European Simulation Multiconference*. Magdeburg.

Pindyck, R. and Rubinfeld, D. 1998. "Econometric Models and Econometric Forecasts". In: McGraw-Hill, 4th edition.

Tong, L. 2003. "Inference of Structural Econometric Models: An Unified Approach". In: Indiana University.

AUTHOR BIOGRAPHIES

GALINA MERKURYEVA is Professor at the Department of Modelling and Simulation, Riga Technical University. She holds Doctor of Engineering (Dr.sc.ing) from Institute of Electronics and Computer Techniques of Latvian Academy of Sciences and Doctor of Technical Sciences (Dr.tech.sc.) from Institute of Control Sciences of Russian Academy of Sciences. G. Merkuryeva has professional interests in discrete-event simulation, simulation metamodeling, decision support tools development, and simulation based training.

LIANA NAPALKOVA holds a BSc. degree in Computer Science from Riga Technical University. Graduated from Herder School in Riga, Latvia. Currently, she is a research assistant at Department of Modelling and Simulation and develops Master thesis on neural network-based simulation metamodelling.