

Soft Sensor for The Adaptive Catalyst Monitoring of a Multi-tube Reactor, Final Report for NiSIS Competition 2006

Martin Macas
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics
Technicka 2, Prague 6, 166 27, Czech Republic
Phone: +420 224 357 666, Fax: +420 224 923 677
email: lhotska@fel.cvut.cz

ABSTRACT: This report describes the solution of NiSIS Competition 2006 awarded by “best nature inspired concept” award.

KEYWORDS: Particle swarm optimization, Elman neural network, prediction.

INTRODUCTION

The objective of the competition is to create an adaptive mathematical model describing the relationship between 14 input variables and one output variables, all of them varying with time. Such a model probably has to be adapted to process state changes resulting from non-measurable influences. After adaptation to the current working point of the process the model should be able to predict the output variable over a certain time horizon, supposed the future behavior of the input variables is known. For detailed description see [1].

INPUT DATA PREPROCESSING

Original input data consist of 14 time series of measured features. The third feature representing *measured concentration of combustible component in combustible gas feed in mass fraction* was recognized as carrying no information was cancelled from the data set and thus just thirteen features were used for further processing.

First, the outlying values were replaced using following method. The 85th and 25th percentiles were computed for each feature defining the upper and the lower bound respectively. Next, all values higher than the 85th percentile were replaced by the percentile value. The same operation was done for the extremely low values using the 25th percentile.

In the next step of preprocessing, all features were normalized in order to have the zero mean and the standard deviation equal to 1. Further, the principal component analysis was applied in order to obtain less correlated features. Just first five principal components were used for subsequent processing. The input dimension was reduced from 14 to 5.

The next phase of preprocessing consisted of application of smoothing filters. The moving average filter averaging 100 points was applied on the feature time series. Finally, for each time point t in each i -th time series $X_i(t)$, the following two special features were extracted:

$$F_{1,i}(t) = \frac{1}{100} \sum_{\tau=t-100}^t X_i(\tau) \quad (1)$$

$$F_{2,i}(t) = \frac{1}{100} \sum_{\tau=t-100}^t ((X_i(\tau) - X_i(\tau-1))^2) \quad (2)$$

In this way, the data set with 10 features was obtained, which was further normalized into range $(-1;1)$ and used for model design and training.

POSTPROCESSING

Output data consisted of one time series representing the *catalyst activity*. The only preprocessing applied here was very slight smoothing (moving average from 10 points) and normalization into range $(-1;1)$.

MODEL STRUCTURE

The model used for prediction is based on neural network. The system which is to be modeled has many unknown delays between inputs and outputs; therefore, the model has to be able to handle with time context of data. That is why the recurrent neural network was supposed to be the suitable one. The special case of recurrent network was used – the Elman network. It has two layers with feedback from the first-layer output to the first layer input. This feedback enables the network to add the time context information into the modeling process. The schema of Elman network is depicted in Figure 2.

The model consists of hidden layer with sigmoidal neurons, output layer with linear neuron (just one was used here) and the context layer which is represented by a recurrent connection with delay. The delay in this connection stores values from the previous time step, which can be used in the current time step. The connections between output of hidden layer and the context units are fixed (the weights are set to 1). All the remaining connections are trainable. For detailed description of the Elman network see [2].

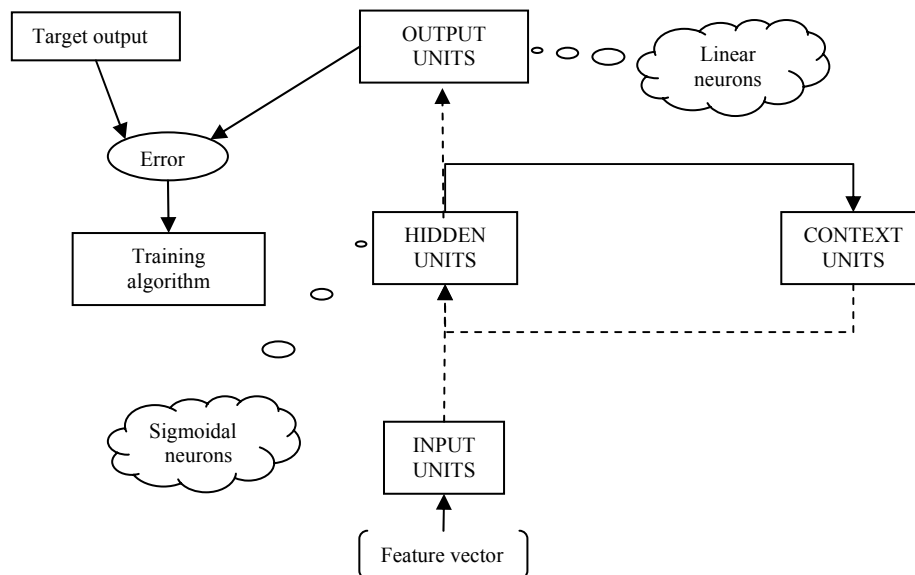


Figure 2: The schema of Elman recurrent network. The trainable connections are represented by dashed lines.

d (the weights are set to 1). All the remaining connections are trainable. For detailed description of the Elman network see [2].

TRAINING METHOD

The Elman network could be trained by any optimization algorithm. The most common method are gradient-based back-propagation algorithms. There are two main disadvantages. First, the gradient algorithms are often getting stuck in local optima. The second problem is computation of gradient. The contributions of weights and biases to errors via the delayed recurrent connections can not be computed and considered. Therefore these contributions are ignored and only *approximation* of the error gradient is used. Both the two problems are solved in solution described here using the global searching and not-gradient- based method. The Particle Swarm Optimization (PSO) method was used here for training the Elman network.

The PSO method is one of optimization method developed for searching global optima of some nonlinear function [3]. It has been inspired by social behavior of birds and fish. The method applies the approach of problem solving in groups. Each solution consists of set of parameters and represents a point in multidimensional space. The solution is called "particle" and the group of particles (population) is called "swarm". Each particle i is represented as a D -dimensional position vector $x_i(t)$ and has a corresponding instantaneous velocity vector $v_i(t)$. Furthermore, it remembers its individual best value of fitness function and position p_i which has resulted in that value. During each iteration t , the velocity update rule (3) is applied on each particle in the swarm. The p_g is the best position of the entire swarm and represents the social knowledge.

$$v_i(t) = wv_i(t-1) + \varphi_1 R_1(p_i - x_i(t-1)) + \varphi_2 R_2(p_g - x_i(t-1)), \quad (3)$$

The parameter w is called inertia weight and during all iterations decreases linearly from w_{start} to w_{end} . The symbols R_1 and R_2 represent the diagonal matrices with random diagonal elements drawn from a uniform distribution between 0 and 1. The parameters φ_1 and φ_2 are scalar constants that weight influence of particles' own experience and the social knowledge.

Next, the position update rule (4) is applied:

$$x_i(t) = x_i(t-1) + v_i(t). \quad (4)$$

If any component of v_i is less than $-V_{max}$ or greater than $+V_{min}$, the corresponding value is replaced by $-V_{max}$ or $+V_{max}$ respectively. The V_{max} is maximum velocity parameter. The update formulas (3,4) are applied during each iteration and the p_i and p_g values are updated simultaneously. The algorithm stops if maximum number of iterations is achieved.

Each particle corresponds to one particular set of network weights and biases. The topology of network (numbers of neurons in layers) were fixed and were set experimentally. Therefore, each particle corresponds to one network. The evaluation of each particle (fitness) was done using network training error.

The training of model (network) for prediction of STEP 1 was performed using random initialization of swarm. The parameters of PSO were set as following:

Parameter	Value
φ_1 and φ_2	2
V_{max}	0.5
w_{start}, w_{end}	0.9, 0.4
Initialization range	$(-0.5; 0.5)^{\text{dimension}}$
Swarm size	30 particles
Iterations	2000
Hidden units	10
Output units	1
Context units	10
Input units	10

The mean squared error between the actual (target) catalyst activity and the network response was used as fitness. Important property of predicted system is its non-stationarity. Therefore, the training data did not consist of the whole time series, but only a window (here referred as training window) foregoing to the prediction point must be considered. The length of the training window was set experimentally to 3300 samples (138 hours) as it is depicted in Figure 3.

ADAPTATION OF THE TRAINED MODEL

The prediction model has to be adapted to process state changes resulting from non-measurable influences. One possibility is to shift the training window in time where the prediction is required and to train new network. However, it is probable, that the modeled system is not completely different in the new prediction point, its dynamics is very similar and it is enough to just slightly adapt the trained model to the new working point. This problem is solved in terms of swarm initialization. In contrast to the basic PSO algorithm described above, during adaptive process, the swarm is not initialized totally randomly. It is split into two parts, the first part is reinitialized randomly and the second part is preserved. This method preserves some information from the old model and simultaneously increases the

diversity of the population and possibility of finding new optima. The partly reinitialized swarm is further used in common PSO described above. The Figure 4 shows the time ranges of training data. The first swarm was randomly initialized, the swarms of step 2-4 were initialized partly. Figure 5 shows how the information is preserved during all adaptation processes.

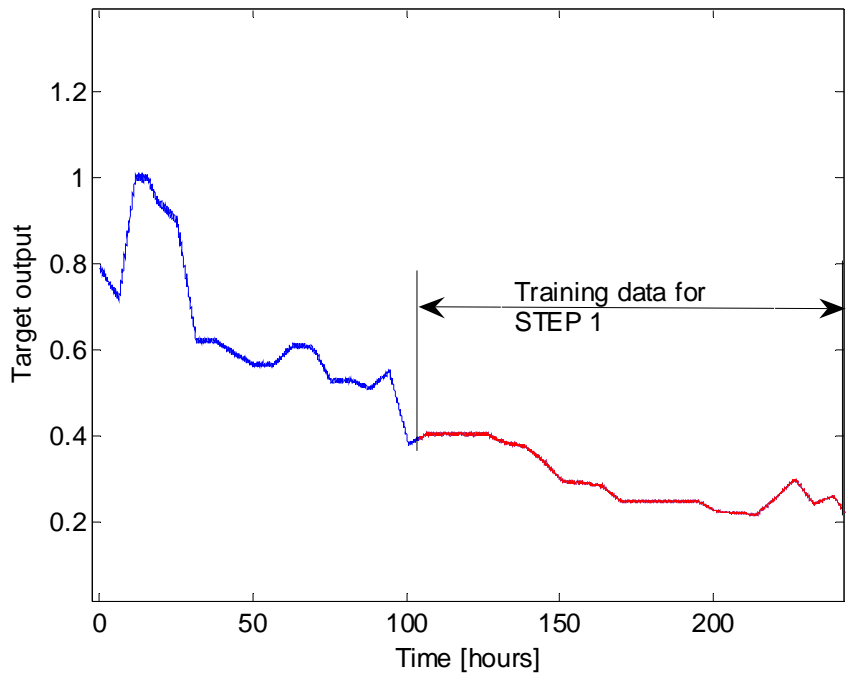


Figure 3: Only a part of the whole data set was used for training due to the non-stationarity of modeled system. The input data were split in the same manner

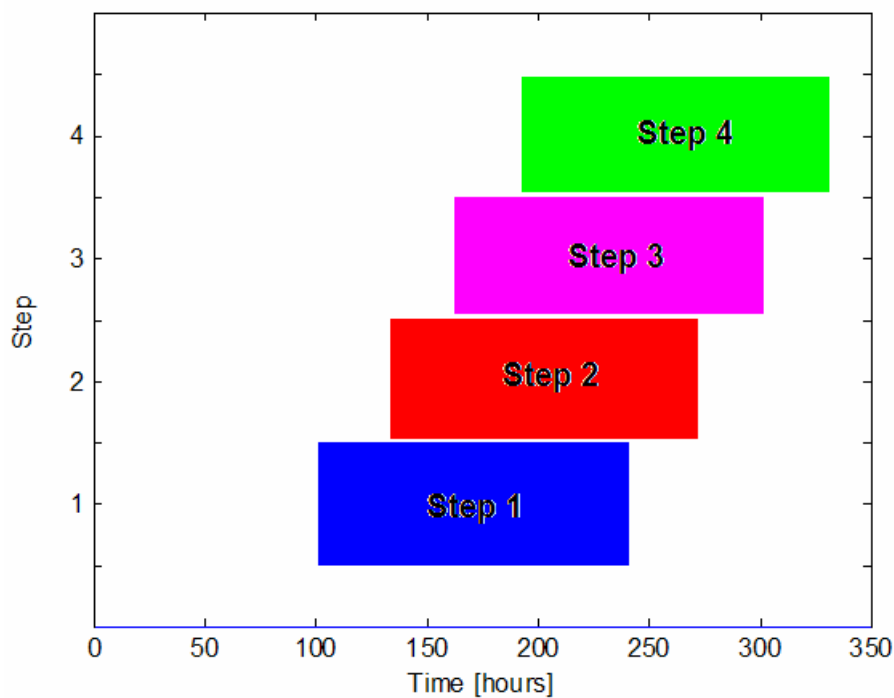


Figure 4: Just a part of the whole data set was used for training due to the nonstationarity of modeled system. The input data were split in the same manner

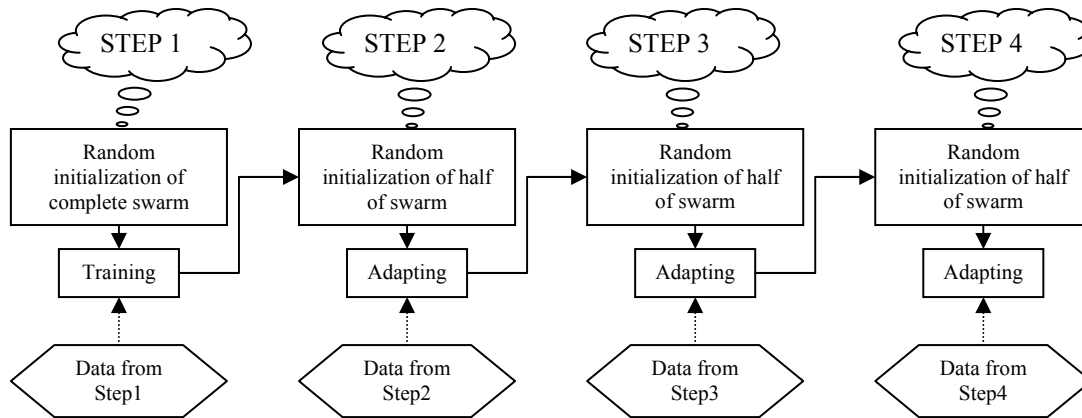


Figure 5: The diagram of solution

RESULT AND CONCLUSIONS

The described approach was used for prediction of the catalyst activity in 4 steps. The result of first validation experiment, where the whole training data set of STEP 1 was split to training and testing set is depicted in Figure 6. The green curve is the validation part of the data set and the blue curve is the model output.

The final result of all steps except the result of STEP 4 is shown in Figure 7. It could be seen that the prediction differs significantly in several points. These differences are probably caused by errors in input data (measurements) or by fixed training window used for adaptation.

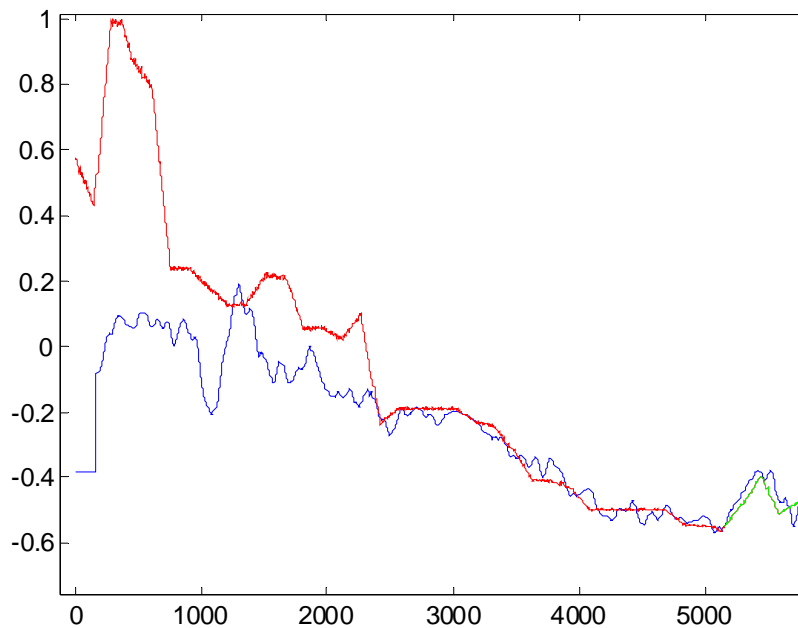


Figure 6: The first validation of the model

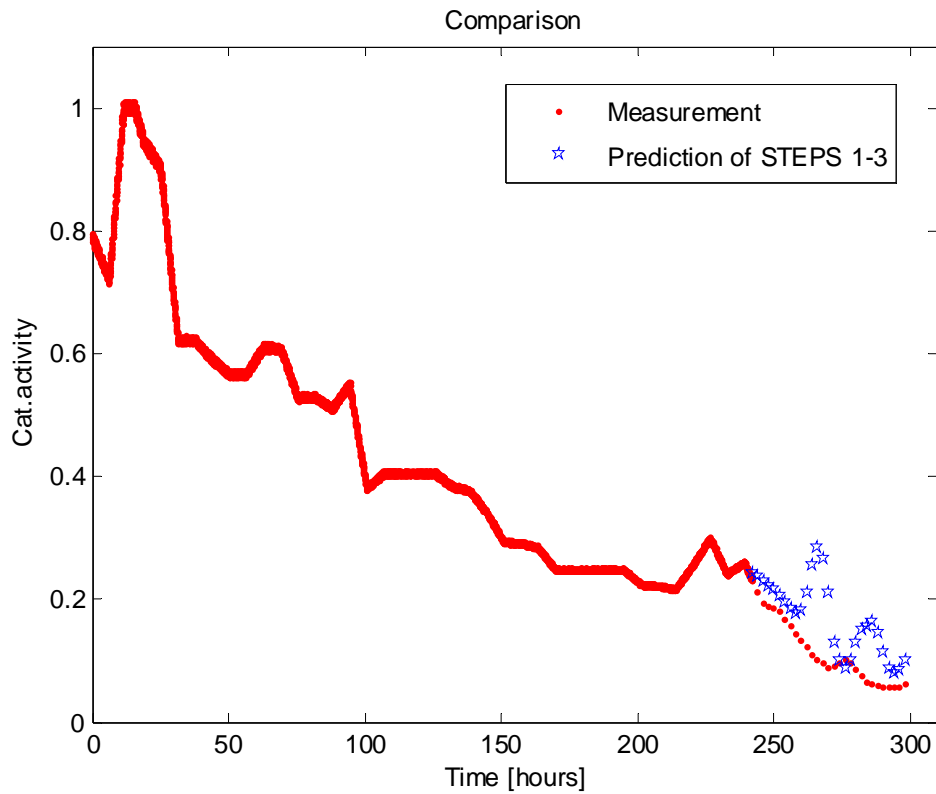


Figure 7: The final result of steps 1-3

The approach is highly inspired by nature from two points of view. First, the neural network model was used inspired by neural nets in living organisms. Moreover, the optimization algorithm was used for training and adaptation, which is inspired by movement of groups of creatures (fish, birds, insects) and even has a socio-cognitive metaphor in human decision making (individual and social knowledge).

REFERENCES

- [1] www.nisis.de
- [2] Elman, J. L., "Finding structure in time," Cognitive Science, vol. 14, pp. 179-211, 1990.
- [3] Kennedy J, Eberhart RC. Particle swarm optimization. In: Proc. IEEE International Conference on Neural Networks, 1995:1942-1948.