

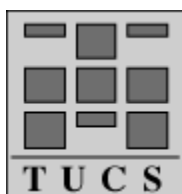
# **Advanced Data Forecasting using Retraining Neural Network Technique**

## **Iulian Nastac**

Turku Centre for Computer Science,  
Institute for Advanced Management Systems Research,  
Lemminkäisenkatu 14B, FIN-20520 Turku, Finland  
e-mail: Iulian.Nastac@abo.fi

## **Adrian Costea**

Turku Centre for Computer Science,  
Institute for Advanced Management Systems Research,  
Lemminkäisenkatu 14B, FIN-20520 Turku, Finland  
e-mail: Adrian.Costea@abo.fi



Turku Centre for Computer Science  
TUCS Technical Report No 542  
June 2003  
ISBN 952-12-1197-0  
ISSN 1239-1891

## **Abstract**

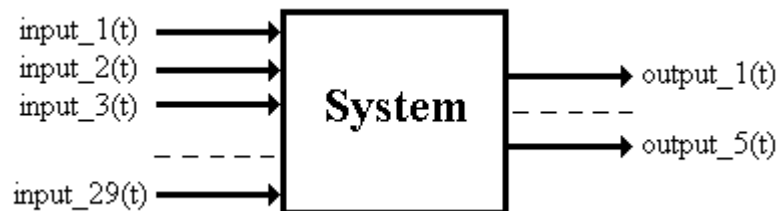
This paper is a report that describes a feedforward neural network architecture and an application of retraining algorithm in order to forecast relevant process variables representative for glass manufacturing, provided by EUNITE Competition 2003. The main purpose is to establish an optimum feedforward neural architecture and a well suited delay vector for data forecasting. The artificial neural networks (ANNs) ability to extract significant information provides valuable framework for the representation of relationships present in the structure of the data. The evaluation of the output error after the retraining of an ANN shows us that this procedure can substantially improve the achieved results.

**Keywords:** artificial neural network (ANN), forecasting, retraining

## 1. Introduction

Artificial Neural Networks (ANNs) are modelling tools having the ability to adapt to and learn complex topologies of inter-correlated multidimensional data. Constructing reliable time series models for data forecasting is challenging due to nonstationarities and nonlinear effects [7]. In this report, we present our feedforward ANN model which is useful in the case where there is a huge amount of data that imply the presence of correlations across time.

The goal of our research was to find a mathematical model describing the relationship between 29 input and 5 output variables of a glass manufacturing system (Fig. 1), depicted in the EUNITE Competition 2003 specifications [2].



**Fig. 1.** Multi-input-multi-output system for glass quality

All inputs and outputs vary dynamically, and there might occur large time-delays. Changing an input variable may result in an output change starting only a couple of hours later and going on for up to several days [2].

The raw data consists of 9408 rows (time steps) – one data set every 15 minutes during 14 weeks. For the first 12 weeks (8064 rows) both input and output data were given and used during training process. During the last 2 weeks (1344 rows) only input data were provided and, finally we predicted the corresponding 1344 rows of outputs.

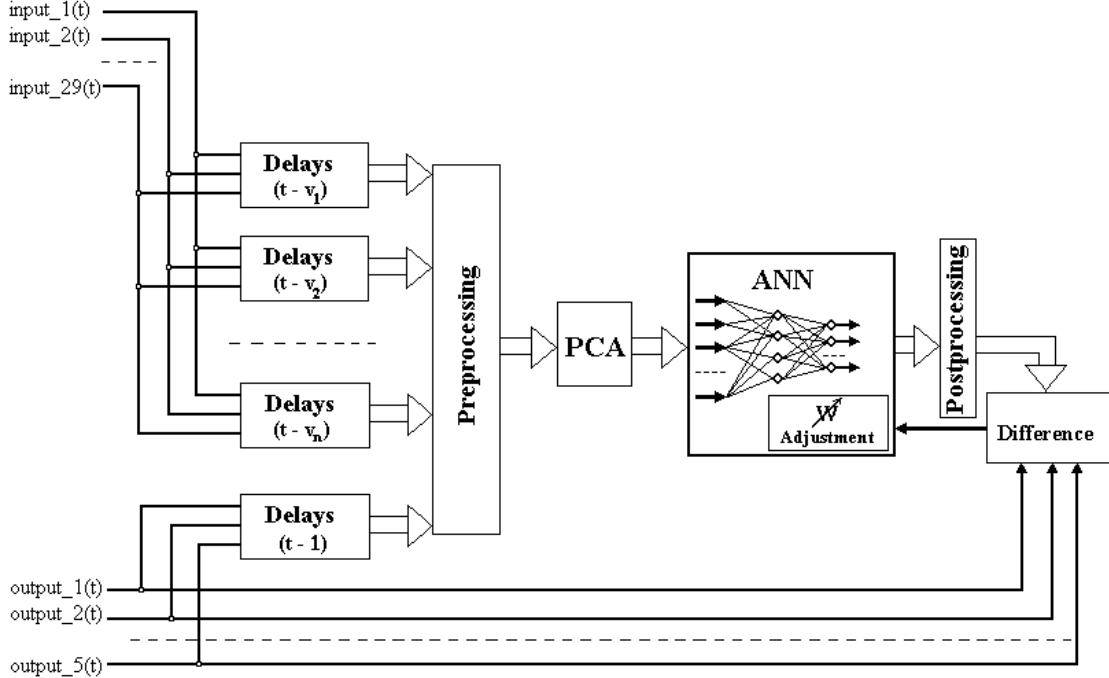
The structure of this paper is as follows. Section 2 presents the problem concerning model structure and data preprocessing. In next section we introduce the retraining technique and explain our approach. The main features of our experimental results are given in Section 4, where we discuss specific aspects. Section 5 presents the competition outcome and some remarks concerning the submitted model. Our conclusions are formulated in the final section of the report.

## 2. Model structure and data preprocessing

A good choice of the training data set is not a trivial task when one wants to make a good prediction. Data preprocessing and data selection remain essential steps in the knowledge discovery process for real world applications and greatly improve the network's ability to capture valuable information when correctly carried out [6] [7].

In Fig. 2 we present our idea in training a feedforward ANN to be a predictor. Delayed rows of the input data are used to construct representations of the current states. For learning purposes, the network inputs consist of many blocks with delayed

values of the glass manufacturing system inputs and one block with delayed system outputs. The ANN target outputs consist of the current values of the glass manufacturing system outputs. Therefore, the network tries to match the current values by adjusting a function of their past values.



**Fig. 2.** Training a feedforward ANN to be a predictor

At moment  $t$ , one output  $output\_i(t)$  is affected by the inputs from different past time steps. For example at moment  $t$  the output  $output\_i(t)$  is affected by all inputs ( $input\_1, \dots, input\_29$ ) at different past time steps:  $t-v_1, t-v_2, t-v_3$ , etc. We denote by *delay vector*,  $Vect\_In$ , a vector that includes the delays taken into account for the model:

$$Vect\_In = [v_1, v_2, \dots, v_n]$$

We used different delay vectors with  $n = 7, 8$  or  $9$  elements, whose values belong to intervals that can cover one to three days. The distribution of the elements was approximately similar to Gamma distribution. The elements of each vector were ascendingly ordered. Consequently the maximum value of any vector is  $v_n$ .

Beside the inputs from different past time steps, the outputs at moment  $t$  are affected, in our model, by the outputs from the previous time step  $t-1$ .

We designed a feedforward ANN with one hidden layer. The ANN model depicted in Fig. 1 restricts the total possible set of training (for model adaptation) to  $8064 \cdot v_n$  input-output pairs.

Once we have decided all the influences on the output at moment  $t$ , we have applied Principal Component Analysis (PCA) [4] [9] to reduce the dimensionality of the input space and to un-correlate the inputs. Before applying PCA we have preprocessed the inputs and outputs using normalization. We have applied the reverse process of normalization in order to denormalize the simulated outputs.

### 3. Training Procedure

As basic training algorithm we have used the Scale Conjugate Gradient (SCG) algorithm [5] [9]. In order to avoid the over-fitting phenomenon we have applied the early stopping method (*validation stop*) during the training process [9].

The accuracy of the result was improved applying in a special way the *retraining technique* [6] [8], which is a practical information extracting mechanism directly from the weights of a reference ANN which was already trained and is perfectly functional at the present time. Briefly, the retraining procedure reduces the reference network weights by a *scaling factor*  $g$ ,  $0 < g < 1$ . These reduced weights are used as initial weights for a new training sequence, with the expectation for a better error as we can see in the following:

- Training an Artificial Neural Network in a natural way with *validation stop* starting with the weights initialized to small uniformly distributed values
- Reduction of the first network weights with a *scaling factor*  $g$  ( $0 < g < 1$ )
- Retraining the network with the new initial weights
- Compare the *validation error* (or training error) in both cases

The data that we used for our model consist of **8064- $v_n$**  input-output pairs. As splitting criterion we have randomly chosen approximately 85% of the data for training set and the remaining for validation.

Next we describe the *three steps* performed to refine our model:

1. First step was performed in order to decide the proper number of hidden neurons ( $N_h$ ). Each of the trainings started with the weights initialized to small uniformly distributed values [3] [6]. We chose the best model according to the smallest error between the desired and simulated outputs. This error was calculated for **8064- $v_n$**  data that include both training and validation sets. We have tested several ANN architecture with  $N_h$  around the geometric mean [1] of input neuron number ( $N_i$ ) and output neuron number ( $N_o$ ):

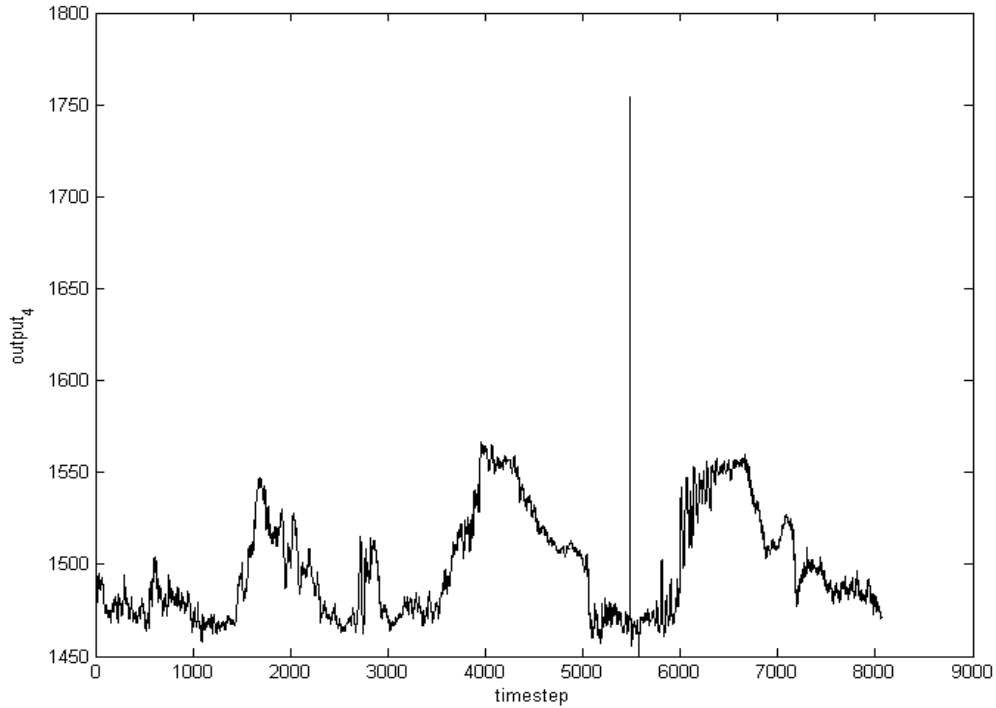
$$\sqrt{N_i * N_o} - 5 \leq N_h \leq \sqrt{N_i * N_o} + 5$$

2. Secondly, we have applied the retraining technique using the ANN architecture (with its associate training and validation sets) obtained in the previous step. We have applied this technique for each value of  $g$  ( $g = 0.3, 0.4, \dots, 0.9$ ), keeping the neural networks that performed the minimum error as the reference network. We repeated this step three times.
3. In this step we have also applied the retraining technique, the only difference from the previous step being that we randomly reconstructed the training and validation sets before each retraining sequence.

The rule used to choose the best model during each step was the mean square error of the differences between real and simulated outputs of **8064- $v_n$**  data that include both training and validation sets.

At each of these 3 steps we obtained a new model. Consequently for one single delay vector we had 3 models. We have applied iteratively the 3 steps above for different delay vectors. In total we have used 12 different delay vectors obtaining 36 models.

We have discovered that for the 4-th output (Fig. 3) there are three anomalous extreme values (during timesteps 5490, 5491 and 5492) that could negatively influence the training process.



**Fig. 3.** The provided data for the 4-th output

Consequently, for half of our models we have decided to level these values by having them decreased from  $output\_4(5489)$  to  $output\_4(5493)$ , in order to improve the forecast precision (see Tab. 1).

	Initial values	Level values
$output\_4(5489)$	1471.94130808653	1471.94130808653
<b><math>output\_4(5490)</math></b>	<b>1754.4285996792</b>	<b>1467.8</b>
<b><math>output\_4(5491)</math></b>	<b>1754.4285996792</b>	<b>1463.7</b>
<b><math>output\_4(5492)</math></b>	<b>1754.4285996792</b>	<b>1459.6</b>
$output\_4(5493)$	1455.63110951847	1455.63110951847

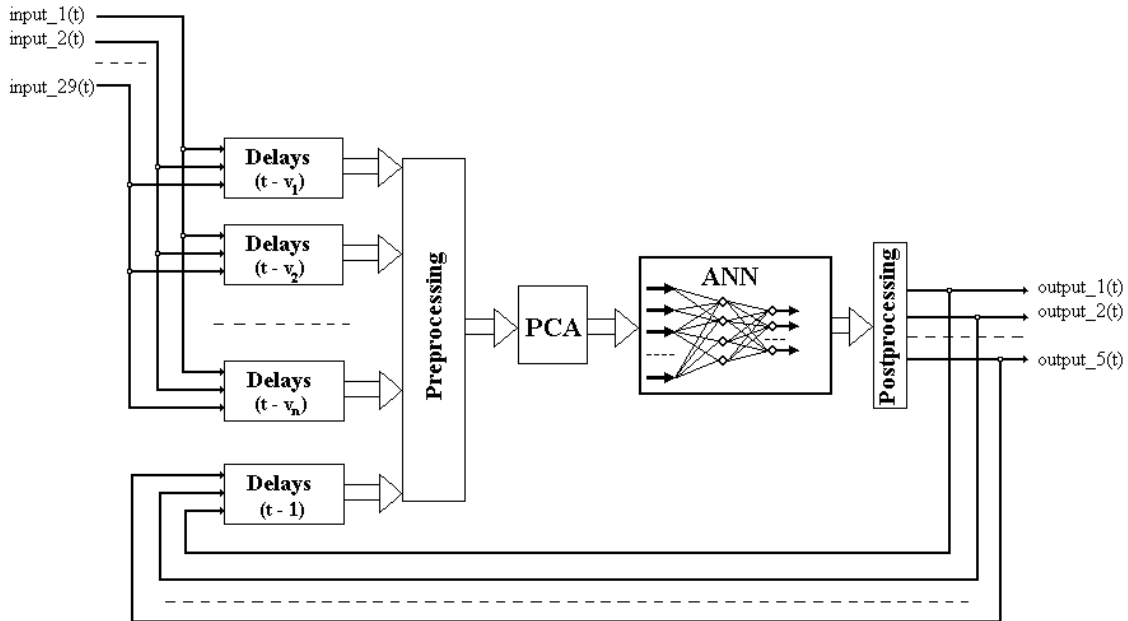
**Tab. 1.** Changing abnormal initial values with level values

The criterion to choose one of these 36 models was the minimum value of ERR (according to the formula provided in the Eunite-Competition 2003 specifications [2]; see next section), but used for five intervals in which we had the real values of the outputs.

## 4. Forecasting Model

In our model the outputs at moment  $t$  are affected by the outputs from the previous time step  $t-1$ . During the training phases we always used the real data at the input.

In the last part we tried to predict the 1344 values of outputs in a sequential mode. Therefore (see Fig. 4), in order to produce the outputs at the timestep ( $t$ ) the neural network used as input, beside the real inputs of the glass manufacturing system from different past time steps, the estimated outputs ( $t-1$ ), which was calculated at the previous step ( $t-1$ ) using the outputs ( $t-2$ ), and so on. Applying this iterative process, a forecast may be extended as many steps as required, but in this case, each step may increase the forecasting error.



**Fig. 4.** Output forecasting

In order to choose the best model among the 36, we have split the output data corresponding to timestep ( $v_n+1$ ) till 8064 in five distinct intervals. The testing intervals are as follows: 400 - 1800; 1850 - 3250; 3300 - 4700; 4750 - 6150 and 6200 - 7600.

We have computed for each interval the error ERR (the error provided in the competition specification [2]):

$$ERR = \frac{1}{5} \sum_{i=1}^5 \frac{100}{N} \sum_{n=1}^N \frac{|O_{Rni} - O_{Fni}|}{|O_{Rni}|} \cdot f(n)$$

where  $N = 1344$  (number of timesteps)

$O_{Rni}$  = real output timestep  $n$  of output  $i$

$O_{Fni}$  = forecasted output timestep  $n$  of output  $i$

and  $f(n) = \frac{500}{500+n}$  a weight function decreasing with the number of timestep  $n$ .

As a measure of model quality we calculated ERR\_M as the mean of 5 ERR errors. Then for each of the 36 models we have one ERR\_M value. All the 36 models were tested against each other in terms of ERR\_M. We have chosen the model with the smallest ERR\_M.

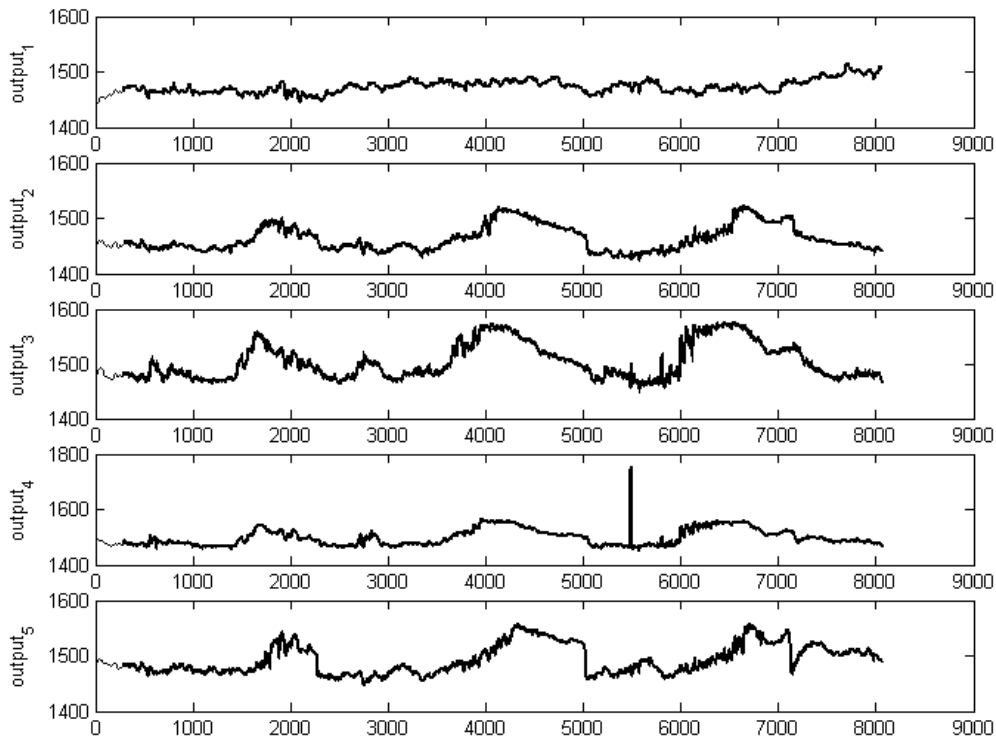
Finally, the parameters of the model reported to EUNITE and derived from step 2 were:

- $Vect\_In = [10\ 20\ 35\ 55\ 80\ 120\ 185\ 290]$
- Number of hidden neurons = 35
- ERR\_M = 0.3602

We noticed that the previous model was among the ones that did not use the level values between output\_4(5489) and output\_4(5493). Even if the fact was a little bit surprising we decided to adopt this model.

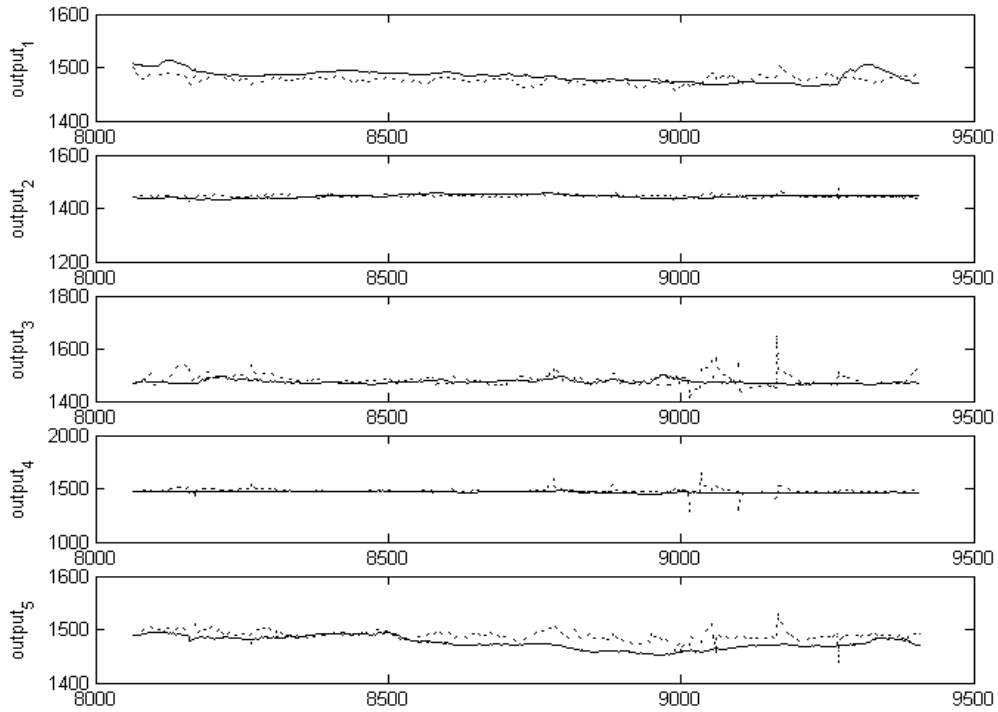
Applying an iterative process (Fig. 4), starting with time step 8065, we finally obtained the desired outputs that correspond to time steps 8065-9408.

The outputs of the model submitted to the EUNITE Competition 2003 are presented in the following figures:



**Fig. 5.** Process outputs. **Real data** (thin lines that correspond to time steps 1-8064) and **neural network values** (thick lines that correspond to timesteps 291-8064) after training process. Thick lines cover very well the thin lines

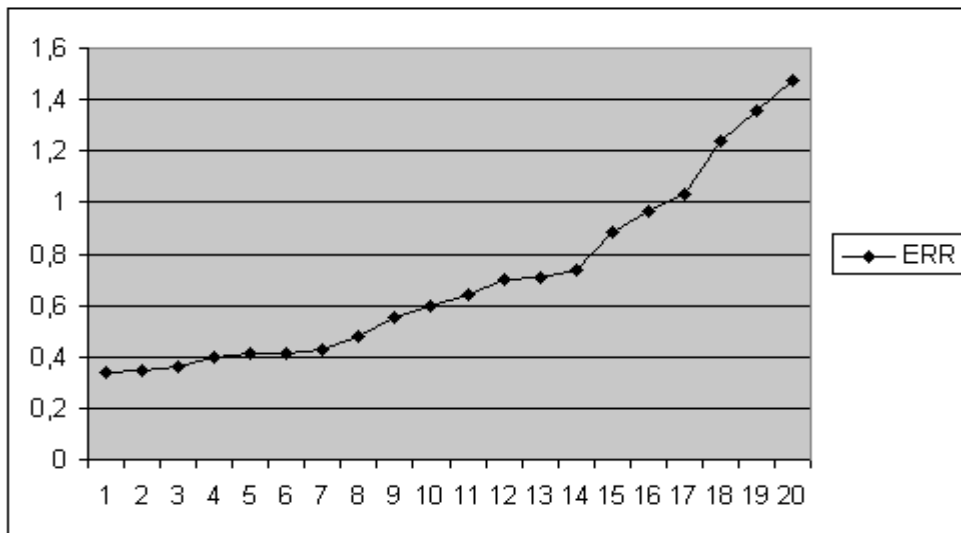




**Fig. 6.** Real outputs (solid lines) provided at the end of the competition and predicted outputs (dotted lines) that correspond to timesteps 8065-9408

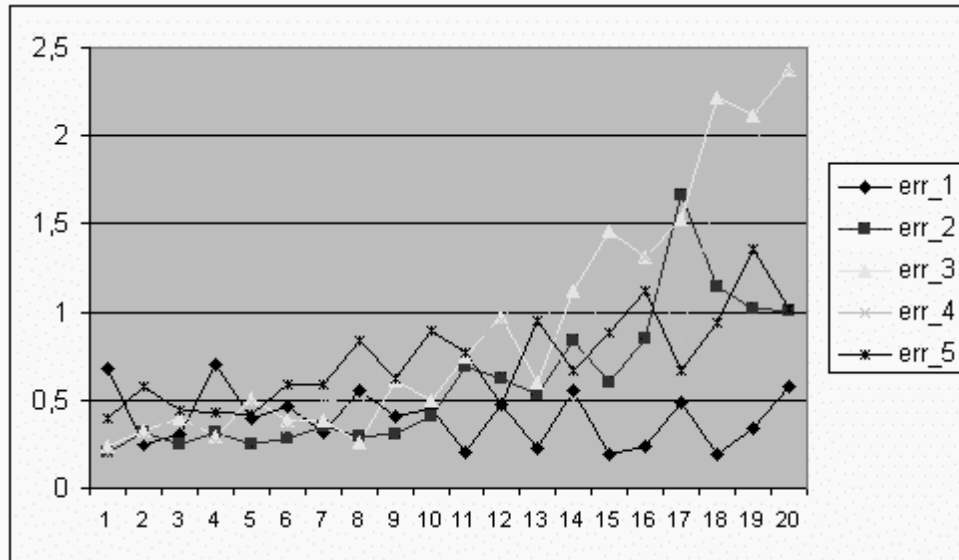
## 5. Competition Outcome

After the evaluation of 20 solutions [2] we have obtained the third position according to the defined error criteria (Fig. 7).



**Fig. 7.** Total error of all 20 solutions using the defined error criteria (provided by EUNITE [2])

There is a small difference among the first three winners. But, on the other hand, our solution is by far the best one in conformity with the variance of all five output errors (see Fig. 8).



**Fig. 8.** Error of all five outputs for the 20 solutions. You find for each solution (No 1 till 20) five markers indicating the error of the five outputs (provided by EUNITE [2])

At the end of the competition, the real outputs (for timesteps 8065-9408) allowed us to estimate ERR for all 36 models. We discovered that there were three models better than the one that we submitted (see Tab. 2). All of them used during the training process the level values between output\_4(5489) and output\_4(5493).

	Models		
	3_PCA_II	5_cuPCA_II	5_cuPCA_III
Number of hidden neurons	39	42	42
Vect_In	[10 20 30 45 65 95 145 210 330]	[12 22 31 39 47 55 66 79 98]	[12 22 31 39 47 55 66 79 98]
Result of	Step 2	Step 2	Step 3
<b>ERR</b>	<b>0.3629</b>	<b>0.3707</b>	<b>0.3788</b>
err_1	0.4159	0.4361	0.4586
err_2	0.2593	0.2994	0.3059
err_3	0.4294	0.3729	0.4991
err_4	0.3902	0.2904	0.2639
err_5	0.3196	0.4549	0.3663
Standard deviation of all five output errors	<b>0.0718</b>	<b>0.0757</b>	<b>0.0994</b>

**Tab. 2.** Parameters of the better (unsubmitted) models

This means that our idea to choose the best model in terms of ERR\_M was not perfect. More than five distinct intervals used to compute ERR should improve the criterion of the selection.

## 6. Conclusions

In this paper, we have designed a neural network tool for data prediction. Our method exploits the input-output dependence across time using a delay vector. We employed the PCA procedure in order to reduce the dimensionality of the input space and to uncorrelate the inputs. The learning process was refined applying the retraining procedure.

It is important to study the shapes of the graphs before training in order to level some unnatural values. The decision to submit a model without level values between output\_4(5489) and output\_4(5493) was not the best. Choosing the best model is not an easy task. It should be improved using more data that cover all potential situations as much as possible.

We were limited by the memory and speed of our computer (512 Mb of RAM and Pentium 4 CPU 1.7 GHz). We are definitely convinced that using vectors with more than 9 elements we can increase the performance of our tool. At the same time there are other efficient algorithms like Levenberg-Marquardt or Bayesian regularization that also necessitate a powerful machine to solve the problem. It is very easy to change in our tool the SCG algorithm with one of these because at the basic level the architecture and the retraining procedure are independent of the training algorithm.

We noticed that the retraining technique significantly improved the achieved result.

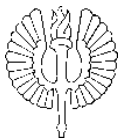
## References

- [1] Basheer, I.A., Hajmeer, M.: Artificial neural networks: fundamentals, computing, design, and application, Journal of Microbiological Methods, Elsevier Science, Vol. 43 (2000) 3-31
- [2] Eunite Competition 2003: Prediction of product quality in glass manufacturing, available on <http://www.eunite.org/eunite/events/eunite2003/competition2003.pdf> (2003)
- [3] Hagan, M.T., Demuth, H.B., Beale, M.: Neural Networks Design, MA: PWS Publishing, Boston (1996)
- [4] Jackson, J.E.: A user guide to principal components, John Wiley, New York (1991)
- [5] Moller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning, Neural Networks, Vol. 6 (1993) 525-533
- [6] Nastac, I.: Contributions in Technical Systems Quality Modelling through the Artificial Intelligence Methods, Ph.D. dissertation, Polytechnic University of Bucharest (2000)
- [7] Nastac, I., Koskivaara, E.: A Neural Network Model for Prediction: Architecture and Trainig Analysis, TUCS Technical Report 521, Turku Centre for Computer Science, (2003) (also available on <http://www.tucs.fi/Research/Series/serie.php?year=2003&type=techreport>)

- [8] Nastac, I., Matei, R.: A Retraining Improvement of Feedforward Neural Networks, TUCS Technical Report 504, Turku Centre for Computer Science, (2003) (also available on <http://www.tucs.fi/Research/Series/serie.php?year=2003&type=techreport>)
- [9] Neural Network Toolbox User's Guide, The MathWorks, Inc., Natick (2001)

Turku Centre for Computer Science  
Lemminkäisenkatu 14  
FIN-20520 Turku  
Finland

<http://www.tucs.fi/>



University of Turku

- Department of Information Technology
- Department of Mathematics



Åbo Akademi University

- Department of Computer Science
- Institute for Advanced Management Systems Research



Turku School of Economics and Business Administration

- Institute of Information Systems Science