Prediction of product quality in glass manufacturing process using LTF-A neural network

Marcin Wojnarski

Faculty of Mathematics, Informatics and Mechanics
Warsaw University
Banacha 2, 02-097 Warsaw, Poland
mwojnars@ns.onet.pl

1 Introduction

This report presents solutions of EUNITE Competition 2003 problem "Prediction of product quality in glass manufacturing process". The first solution is based on Local Transfer Function Approximator (LTF-A) neural network, while the next three solutions utilize simple rules to predict glass quality.

Despite advanced data preprocessing, LTF-A did not performed very well on the competition data. The reason for this was probably the difficulty of the problem and the lack of strong relations in the data ifself.

2 LTF-A neural network

Local Transfer Function Approximator (LTF-A) neural network can be used to approximate functions defined on $\mathbf{X} = \mathbf{R}^n$ into $\mathbf{R}$.

The general idea underlying this system is to approximate the target function separately on small subsets of the domain $\mathbf{X}$, and then put these local approximations together.

Very simple functions are used as local approximations – most often a constant or linear function (in this case the search for a local approximation is slightly similar to linear regression). Theoretically, many other classes of functions, also more complicated, could be applied.

The subsets which are the domains for local approximations are multidimensional ellipses. Every ellipse can have different shape (i.e. lengths of axes). They can also partially overlap – in this case the global approximation in a given point is defined as an average of all local approximations in this point.

An important feature of LTF-A is that the ellipses are fuzzy sets, with gaussian membership functions. This enables soft transitions between neighboring local approximations and gives a smooth global approximation.

Positions, axes lengths of ellipses and local approximations are chosen during training, adaptively, i.e. through small corrections after each presentation of a training pattern. The corrections are done in such a way that after next presentation of the same pattern the system response is more correct.

The architecture of LTF-A neural network expresses the above idea of local approximations. Every neuron in the network corresponds to one local approximation. A neuron is composed of two subunits – the so-called *decision neuron* (or *D-neuron*) and the *restriction neuron* (*R-neuron*). An R-neuron defines the ellipse which is the domain of approximation, while D-neuron defines the local approximation in itself.

Network structure is dynamical, it changes during training. New neurons are inserted when the network error is large. At the beginning there is only one neuron, the so-called *basis neuron*, which has the whole space $\mathbf{X}$ as a domain and approximates constant function equal to the mean of the output.

Neurons can also be removed during training. The reason for removal can be two-fold:

- a neuron increases average network error, or
- an R-neuron (i.e. the domain of local approximation) is very small, it does not contain almost any training patterns. Approximating on the basis of so few examples would be risky and could lead very often to overtraining.

Architecture of LTF-A neural network described above is the same as the so-called Sugeno, or Takagi-Sugeno-Kang, fuzzy system [1, 2]. Thus, LTF-A can be viewed also as a neuro-fuzzy system – it can be initialized like a fuzzy system and then trained like a neural network.

LTF-A is at the same time an extension of LTF-C neural network, which is designed to solve classification problems [3, 4]. System based on LTF-C won the 2^{nd} EUNITE Competition "Modeling the Bank Client's Behavior".

3 Data preprocessing

Provided data consisted 9408 time steps. For each of them 29 input values were given. For the first 8064 time steps 5 output values were also provided. The goal was to predict output values in the last 1344 time steps.

The data required thorough preprocessing, in order to eliminate situations which could negatively influence the network training process. Two main problems had to be fixed:

- Outliers. Most of attributes contained single values lying very far from their means. Such values were replaced with means of corresponding attributes. Outliers were removed from output attribute no. 4 and input attributes no. 2-28.
- Shifts. Most of input attributes contained short time ranges of significantly shifted values. These shifts were fixed by adding a constant which made the mean of shifted values the same as the mean of the rest of values in a given attribute. Shifts were fixed in input attributes no. 2-11, 13-23, 25-28.

In many cases outliers and shifts had to be found manually, by analysis of graphs of attribute values versus time.

4 Model structure

Four solutions of the competition problem were submitted. The first one was obtained with LTF-A neural network, while the rest were generated by very simple rules, which did not require training.

4.1 System based on LTF-A neural network

This system was composed of five LTF-A networks, each one predicting another output attribute. The networks predicted the *increase* of output values in time step $t+1$ (i.e. $output(t+1) - output(t)$) based on input and output values in time step t ($input(t)$ and $output(t)$). Thus, the networks had 34 input attributes.

The system was trained on 8063 examples corresponding to time steps when the output values were known. It was composed of 141 neurons in total.

No adaptation to current working point was applied.

4.2 Naive rules

Three solutions of the competition problem were generated by very simple models, which predicted $output(t+1)$ according to the following rules:

1. $prediction1 = output(t)$,
2. $prediction2 = \text{mean}(output)$.
   The mean was computed on the whole training data,
3. $prediction3 = \alpha(t) * prediction1 + (1 - \alpha(t)) * prediction2$.
   $\alpha(t)$ was a linear function of time, equal 1 at the beginning of the forecasting period and 0 at the end.

5 Results

Inaccuracy of the neural model was measured with the use of the Root Mean Squared Error divided by standard deviation of a given output attribute. This measure is very objective, as it is insensitive to attribute rescaling. It can be also easily interpreted: only if the error of a system is lower than 1, the system is better than a naive model which predicts always the mean of the output attribute.

Unfortunately, the system based on LTF-A had training error only slightly lower than 1 (≈ 0.98). This means that the network did not learnt anything, in fact.

Through modifications of training parameters it was possible to force LTF-A to fit better to the training data. However, this did not necessarily mean that resulting model would perform better during test, as well.

To check this, a test set was created, containing patterns no. 5001-8063 from the original training set. The first 5000 patterns formed a new training set, which was used to create a neural network. Although this network performed better on the training set, it had very poor accuracy (error over 1) on the test set.

This procedure was repeated several times, always with the same result. Other model structures were also tried using this method. Those models used different input attributes – exponential averages $\mathit{inputexp}_\alpha(t)$ and $\mathit{outputexp}_\beta(t)$ of previous values of *input* and *output* instead of $\mathit{input}(t)$ and $\mathit{output}(t)$.

An exponential average, e.g. $\mathit{inputexp}_\alpha(t)$, was computed as:

$$\mathit{inputexp}_\alpha(0) = \mathrm{mean}(\mathit{input})$$

$$\mathit{inputexp}_\alpha(t) = (1-\alpha) * \mathit{inputexp}_\alpha(t-1) + \alpha * \mathit{input}(t)$$

Different values of α and β, from 0.0005 to 0.1, were tried. Some models utilized also several kinds of averages together, with different values of α and β. However, all these models had test error rate over 1.

With such poor results as given above it is useless to present them on a diagram.

6 Conclusions

The unsatisfactory results of LTF-A are very odd. The network was unable to find any useful relationship in the data, despite advanced data preprocessing. This leads to a conclusion that either LTF-A cannot cope with such kinds of problems, or the provided data does not carry sufficient information to predict output parameters of the glass manufacturing process.

Probably the latter is the case, since up to now LTF-A performed very well in other real-world problems. Moreover, tha analysis of 2-dimensional graphs of k-th output attribute $(\mathit{output}_k(t+1) - \mathit{output}_k(t))$ against some specified input attribute, such as $\mathit{input}_i(t)$, $\mathit{output}_j(t)$ or $\mathit{inputexp}_{i,\alpha}(t)$, for various k, i, j or α, did not show any relationship, as well.

For that reason, in order to create a more accurate model of the glass manufacturing process, more training data is necessary, gathered during longer period of time and containing more attributes relating somehow to the predicted ones.

Also, disclosure of the meaning of the attributes and/or incorporation of expert's knowledge into model design would make better attribute selection/creation possible. This would help create better model, too.

References

1. Kruse, R., Gebhardt, J., Klawonn, F.: *Foundations of Fuzzy Systems*, Wiley, 1994.
2. Sugeno, M.: *Industrial applications of fuzzy control*, Elsevier Science Pub. Co., 1985.
3. Wojnarski, M.: LTF-Cimulator,
   Online: http://rainbow.mimuw.edu.pl/∼mwojnar/ltfcim.
4. Wojnarski, M.: LTF-C: Architecture, Training Algorithm and Applications of New Neural Classifier, *Fundamenta Informaticae*, **54**(1), 2003, 89–105.