Growing fuzzy inference neural network applied to the NN3 neural forecasting competition

Jiří Iša

***Abstract*—Growing fuzzy inference neural system (GFINN) is a fuzzy–neural network model. Its functionality can be expressed in a form of fuzzy if–then rules. The skill of the GFINN model to grow allows it to change its size and structure according to the training data. The resulting structure allows for a simple input features selection — not all input features have to be used in every fuzzy rule. The new algorithm for the computation of output weights runs much faster than least mean squares estimate, while the experiments performed in past show nearly identical performace of both methods. Furthermore, the new training method guarantees the output weights to remain inside the output values interval. This makes the extracted rules reasonable, unlike with least mean squares estimate. This text describes the GFINN model and its application to the NN3 neural forecasting competition.**

I. Introduction

In past, artificial neural networks have been successfully deployed in many domains. Their usage may become even more widespread, if their functionality becomes more transparent. For most neural models, their functionality is a black box. Nobody knows, how the network works, or why it does not. Fuzzy neural networks offer a possibility to express its own functionality in a form of human–readable fuzzy if–then rules. A human expert can verify, whether the obtained fuzzy rules seem reasonable. If they do not, the network parameters can be changed to reflect the problem. Having this extra requirement, to discover the fuzzy rules, in mind, the fuzzy neural network training may become more complex than a more traditional approach. This text introduces the Growing fuzzy inference neural model and applies it to the NN3 competition task.

II. Growing fuzzy inference neural network

The growing fuzzy inference neural network model (GFINN) has been developed by the author of this text [1], because the existing fuzzy neural models [2], [3] represent a very constraining top–restricted approach — the number of fuzzy rules is pre–specified. In some models the obtained rules can get pruned or merged, but it is never automatically recognized, that the task may be too complex for the given size of the network. The Growing neural gas model (GNG) [4], [5] offers a way out of these troubles. For the purpose of the GFINN model, the lifelong tendency of GNG to grow was extended by the ability to recognize the useless neurons and remove them from the network structure. The experiments performed so far have shown that the GFINN size stabilizes after a small amount of training cycles.

Jiří Iša is a student of Faculty of Mathematics and Physics at Charles University in Prague, Czech Republic; e–mail: jiri dot isa at matfyz dot cz

A. Structure of the network

GFINN is essentially a RBF–like neural network. It consist of an input layer, a self–organizing hidden layer and an output layer. The neurons in the input layer represent the input of the network. Every neuron n_i in the hidden layer has an attached reference vector $\overrightarrow{m_i} \in \Re^n$, where n is the input space dimensionality. Every hidden neuron n_i also has an attached variable $error_i$ used during training. In the single–output version of the network used in this text, the output neuron uses weights $\overrightarrow{w} = \{w_1, \ldots, w_k\}$, $w_i \in \Re$ corresponding to the connection to one of the k hidden neurons, to compute the network output y.

B. Recall

- Set the outputs of the input neurons to the values of the input pattern $\overrightarrow{x} \in \Re^n$.
- Calculate the hidden layers activation according to:

$$a_i(\overrightarrow{x}) = e^{-\frac{\|\overrightarrow{x}-\overrightarrow{m_i}\|^2}{2\sigma^2}}$$

  where the width σ is a parameter of the network.
- Calculate the normalized activation of the neurons in the hidden layer:

$$\nu_i(\overrightarrow{x}) = \frac{a_i(\overrightarrow{x})}{\sum_{j=1}^{k} a_j(\overrightarrow{x})}$$

- Compute the output of the network using the weighted average defuzzification:

$$y(\overrightarrow{x}) = \sum_{j=1}^{k} \nu_j w_j$$

C. Training

Such as it is common for RBF networks, the training consists of several consequent phases. The hidden layer is trained using self–organization and the output weights are trained independently. To improve the reference vectors of the hidden neurons with regards to the output, GFINN interleaves these to steps repeatedly. GFINN also adds the selection of the significant input features to the process.

The training algorithm skeleton is:

- Perform the self–organization step of the hidden layer for all training patterns $\overrightarrow{x}$.
- Use the training patterns and the current configuration of the hidden layer for the output weights adaptation and to obtain local error estimates $error_i$ used in the self–organization.

- If the stop condition has no been met yet (fixed amount of cycles, stable network size, training error threshold achieved, ...), repeat from the first step.
- Perform the significant input features selection.

1) Self–organization of the hidden layer: Because there is no need for a low–level visualization of input patterns, GFINN uses a much more loose structure than the standard regular Kohonen's neural grids are [6] — the Growing neural gas [4], [5] (GNG). GNG is designed to grow infinitely, or for a fixed amount of training cycles. For the purpose of GFINN we have extended it with the removal of insignificant neurons. A neuron n_i is considered ε–insignificant if:

$$\forall n_j \in neighbors(n_i) : \ |w_i - w_j| \leq \varepsilon$$

If an ε–insignificant neuron n_i is removed from the network, its activity is, due to the locality of the activation of the neurons, overtaken by its neighbors with almost identical influence on the network output. ε–insignificant neurons should be removed every few learning cycles, the same way the new neurons are added in the basic training algorithm [4].

We also suggest to use a modified learning rule for the adaptation of the hidden layer reference vector [1]:

$$\overrightarrow{m_i}(t+1) = \overrightarrow{m_i}(t) + \alpha(t)|1 - (w_i - \widehat{y}(\overrightarrow{x}))|[\overrightarrow{x} - \overrightarrow{m_i}(t)]$$

t is a discrete time step, $\alpha(t)$ a learning rate in time t. The $|1-(w_i-\widehat{y}(\overrightarrow{x}))|$ item is added to reduce the attraction of a neuron n_i, if the requested output $\widehat{y}(\overrightarrow{x})$ is very different from the rule, which this neurons represents (with a consequent w_i).

2) Fuzzy–optimal output weights computation (FOPT): Two methods used to obtain the output weights are widely used for RBF networks: gradient descent and least mean squares estimate (LMSE). It has been observed [1] that a) it is difficult to make the gradient descent react to the changes of the underlying hidden layer structure while not overfitting, b) LMSE is computationally expensive and can suggest output weights (rule consequents) outside of the output values interval.

We suggest to use a local error function instead of the standard sum of network error squares. This new error function reflects fuzzy error appearing on every hidden neuron n_i:

$$E_i^{FOPT} = \frac{1}{2} \sum_{(\overrightarrow{x},\widehat{y}) \in T} \nu_i(\overrightarrow{x})(\widehat{y} - w_i)^2$$

where T is a training set consisting of input–output training pairs $(\overrightarrow{x}, \widehat{y})$.

Then the partial derivative of E_i^{FOPT} with respect to w_i can be computed easily:

$$\frac{\partial E_i^{FOPT}}{\partial w_i} = - \sum_{(\overrightarrow{x},\widehat{y}) \in T} \nu_k(\overrightarrow{x})(\widehat{y} - w_i)$$

Requiring this partial derivative $\frac{\partial E_i^{FOPT}}{\partial w_i}$ to equal to zero, we can conclude with the following equation to determine w_i:

$$w_i = \frac{1}{\sum_{(\overrightarrow{x},\widehat{y}) \in T} \nu_i(\overrightarrow{x})} \sum_{(\overrightarrow{x},\widehat{y}) \in T} \nu_i(\overrightarrow{x})\widehat{y}$$

From this formula, it can be seen, that it may be feasible to use weighted averages of the output values of the training patterns as the output weights.

The resulting scheme differs in effect from the standard computations of output weights, because a different error function is used. The local fuzzy error functions E_k^{FOPT} do not have the global impact of the global error function. Due to the weighted average output computation, the GFINN model can also be considered to represent a voting scheme, where every voter (hidden neuron) also estimates its own vote strength and specializes itself for a subset of the input space.

3) Significant input features selection: One of the great benefits of the GFINN representation is, that it supports a straightforward method for the selection of significant input features. As a consequence, the induced rules do not have to use all the input features. The features they actually use may vary for different rules.

The locality of the gaussian activation allows to determine a set of test patterns, which can be used to measure a relevancy of an input feature for a hidden neuron [1].

D. Rules extraction

Every hidden neuron n_i corresponds to a fuzzy rule [7]:

$$If \ \overrightarrow{x} = \overrightarrow{m_i} \ Then \ y = w_i$$

Because of the input features selection, the general rule may become smaller:

$$If \ x_{f_1} = m_{f_1} \ \wedge \ \ldots \ x_{f_p} = m_{f_p} \ Then \ y = w_i$$

where $f_1, \ldots, f_p$ are the selected input features for the respective rule.

III. Application to the NN3 neural forecasting competition

A. Data preprocessing and postprocessing

For simplicity, a floating window of twelve last values is used as an input of the network, which predicts the next value. The modified learning rule assumes the input values to be normalized to the $\langle 0, 1 \rangle$ interval:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

with x_{min} and x_{max} being extreme values of the input feature x in the training set.

To obtain the predicted value, the network output is denormalized using $\widehat{y}_{min}$ and $\widehat{y}_{max}$ as the normalization boundaries.

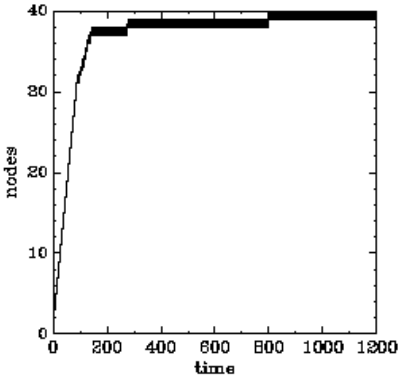

Fig. 1. Illustrational network growth for one of the series.

B. Network configuration and training

A new neuron is added to the network after every third pass through the training set. In the same time 0.001–insignificant neurons are pruned.

Figure 1 shows an illustrational growth of the GFINN network for one of the predicted series.

For every series in the competition a new network is trained, using the same parameters. The network grows by itself as neccessary for the given task. The whole known time series is used as a training set.

C. Far future forecast

The network is configured and trained to forecast a single value. Eighteen future values are requested for every series in the NN3 competition. The designed GFINN model predicts the values one–by–one. The newly predicted value is used as the last variable in the input pattern used for the next prediction (Fig. 2).

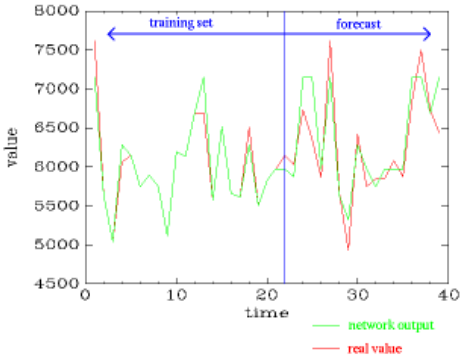

Fig. 2. Illustrational network prediction.

REFERENCES

[1] J. Iša, "Artificial neural networks for clustering and rule extraction," Master thesis, Faculty of Mathematics and Physics, Charles university in Prague, Czech Republic, 2007.

[2] T. Nishina and M. Hagiwara, "Fuzzy inference neural network," *Neural Computing*, vol. 14, no. 3, pp. 223–239, 1997.

[3] J.-S. R. Jang, "ANFIS: Adaptive–network–based fuzzy inference system," *IEEE Transactions on Systems, Man & Cybernetics*, vol. 23, pp. 665–685, 1993.

[4] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds. Cambridge MA: MIT Press, 1995, pp. 625–632. [Online]. Available: citeseer.ist.psu.edu/fritzke95growing.html

[5] M. Martinetz and K. J. Schulten, "A "neural–gas" network learns topologies," in *Proceedings of International Conference on Articial Neural Networks*, O. S. T. Kohonen, K. Mkisara and E. J. Kangas, Eds., vol. 1, 1991, pp. 397–402.

[6] T. Kohonen, *Self–organizing maps, Third extended edition*. Springer, 2001.

[7] J.-S. R. Jang and C.-T. Sun, "Functional equivalence between radial basis function networks and fuzzy inference systems," *IEEE Transactions on Neural Networks*, vol. 4, pp. 156–159, 1993.