Acta Agraria Kaposváriensis (2007) Vol 11 No 2, 259-271
Kaposvári Egyetem, Állattudományi Kar, Kaposvár
University of Kaposvár, Faculty of Animal Science, Kaposvár

Complexity Reduction of Local Linear Models Extracted from Neural Networks

T. Kenesei, B. Feil, J. Abonyi
University of Pannonia, Department of Process Engineering, P.O. Box 158, H-8201, Veszprem, Hungary,

ABSTRACT

Nonlinear black-box models have become more and more important not only in research but also in industrial practice. However, their main disadvantage is that they are often too complex and not interpretable; therefore it is a hard and complex task to validate them by human experts. It is a challenge how a priori knowledge can be utilized and integrated into the black-box modeling approach. This could be a difficult multi-stage process. One of these steps can be the reduction of the identified model. It is also important from the viewpoint of overparameterization and to reduce the time and computational demand of the model. This article would like to show how model reduction techniques can be used for complexity reduction purposes by local models from neural networks. A possible method family is orthogonal techniques. These methods can roughly be divided into two groups: the rank revealing ones like SVD-QR algorithm and those that evaluate the individual contribution of the rule or local models, like the orthogonal least-squares approach (OLS). This later technique requires more computations, but for system identification purposes it is preferable as it gives a better approximation result. Apart from that, other methods can also be used to reduce the number of local models: the most similar models can be merged together. The analyzed methods are used for knowledge discovery purposes from neural networks.
(Keywords: model reduction, model transformation, knowledge discovery.)

ÖSSZEFOGLALÁS

Neurális hálózatok értelmezhetősége és annak javítása
Kenesei T., Feil B., Abonyi J.
Pannon Egyetem, Folyamatmérnöki Intézeti Tanszék, Postafiók 158, 8201, Veszprém

A nemlineáris fekete doboz modellezési technikák napjainkra különösen fontossá váltak nemcsak a tudományos kutatás, hanem az ipari alkalmazás területén is. Fekete doboz modellek lévén legnagyobb hátrányuk, hogy struktúrájuk, illetve paramétereik nem értelmezhetők. Ennek köszönhetően e modellek optimális struktúrájának meghatározása és validálása rendkívül nehéz. Szintén e fekete doboz jelleg miatt jelent nagy kihívást, hogy miként lehet a modellalkotás során előzetes információk felhasználásával javítani a modellezési teljesítményt. A fekete doboz modellek struktúrájának meghatározása tehát többlépcsős folyamat, mely általában egy komplex modell redukálásán alapul. A modell redukció a túlparaméterezés elkerülése miatt kiemelt fontosságú, továbbá használatával számítási idő nyerhető. Cikkünkben azt kívánjuk megmutatni, hogy redukciós technikák segítségével milyen módon lehetséges a fekete doboz modellek komplexitásának csökkentése. Az egyik lehetséges út az ortogonális technikák használata, melyek két további csoportba oszthatók: az ortogonális legkisebb négyzetek módszere (OLS), illetve a rangsoroló SVD-QR

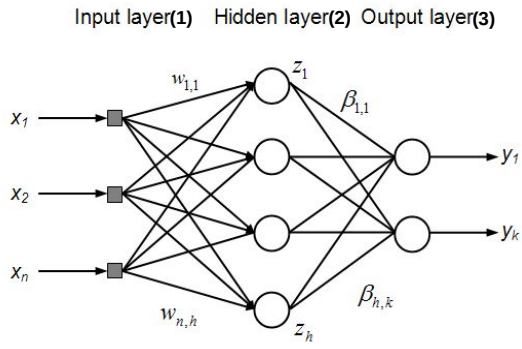
technikák. Az OLS sokkal számításigényesebb, de jó közelítést adó eredményei miatt preferált az identifikációs technikák használata során. A cikk egy, a neurális hálózatok értelmezhetővé tételére alkalmas eljárást mutat be, továbbá ismerteti az így kapott szabálybázis redukálását OLS segítségével. A bemutatásra kerülő eredmények a kidolgozott neurális hálózat részletes analízisére és redukciójára alkalmas technika széleskörű alkalmazhatóságát vetítik előre.
(Kulcsszavak: modell redukció, modell transzformáció, tudásfeltárás.)

INTRODUCTION

Nonlinear black-box models have become more and more important not only in research but also in industrial practice. One of the most often used types of black-box models is neural networks (NN). This type of nonlinear models can be used effectively for many purposes including business decision systems (*Setiono*, 2000), engineering and data mining (*Nelles*, 2001). It consists of simple but strongly connected units called neurons; and generally robust against the failure of single units. Neural networks can be feedforward or recurrent depending on the type of connections. In this paper only feedforward neural networks will be studied. In this type of network the neurons are organized into layers (input, hidden and output layers) as can be seen in *Figure 1*; and there are only connections between neurons in one layer to the following. Cybenko showed that any mapping from $\Re^n$ to $\Re^q$ could be achieved with two layers of hidden nodes, hence neural networks are universal approximators. Hornik showed that any mapping could also be achieved with an arbitrary degree of accuracy using only one hidden layer. The complexity of the network (number of parameters) depends on the number of hidden neurons, since the number input and output neurons are equal to the input and output of the system to be modeled, respectively. Based on these results NNs with one hidden layer will be used in the following.

Figure 1

Multilayer neural network



Source (*Forrás*): (*Benitez*, 1997)

1. ábra: Többrétegű neurális hálózat

Bemeneti réteg(1), Rejtett réteg(2), Kimeneti réteg(3)

However, the main disadvantage of NNs is that they are often too complex and not interpretable. Complexity and interpretability issues are connected with each other: often a relatively simple cross validation method can be used to determine the proper number of hidden neurons but several problems still remain. As *Duch* (2003) showed a simple performance measure (mean square error by function approximation, ratio of wrongly classified samples by classification) is not enough by itself because two NNs with the same performance can have highly different behavior. Other problem is how a priori knowledge can be utilized and integrated into the black-box modeling approach, and how a human expert can validate the identified NNs or more favorably, follow the identification process to interfere in it if it is needed (e.g. to avoid overparameterization or overlook the possible soft or crisp constrains).

To overcome these problems, there are some strategies in the literature:

1. *Visualization of neural network behavior.* This approach utilizes the natural pattern recognition capability of human expert. It aims to draw a two dimensional map that is in connection with the behavior of NN in a specific way.
2. *Transformation of NN.* The aim of this type of methods is to convert NN into a more interpretable form. Because NN is a black-box, other black-box models should be used that are closer to human thinking. A good approach is to extract rules from NN functions and parameters, and represent them as fuzzy (linguistically sound) if-then rules.
3. *Model reduction.* This approach does not aim to give a 'picture' of NN responses for specific inputs and behavior, but overcome complexity problems with the determination of 'importance' of hidden neurons and weights, remove the insignificant ones, and/or merge the similar ones. It is also important from the viewpoint of overparameterization and to reduce the time and computational demand of the model. Naturally, it can be combined with the above mentioned approaches.

The Related works section gives a brief introduction and overview about these methods. The Complexity reduction subsection contains a combined approach used in this paper to get reduced rule based model from NN, and gives viewpoints for future work. Application examples and discussion section describes some illustrative examples, while the Summary section concludes the paper.

RELATED WORKS

Structure of neural networks

This subsection gives a brief introduction how NNs work. This description is mainly based on *Benitez* (1997); for a detailed discussion see *Nelles* (2001). Let us consider the NN in *Figure 1*. It has n input ($x_1,\ldots,x_n$), h hidden ($z_1,\ldots,z_h$), and m output ($y_1,\ldots,y_m$) neurons. Let τ_j be the bias for neuron z_j. Let w_{ij} be the weight of the connection from neuron x_i to neuron z_j, and β_{jk} the weight of the connection from neuron z_j to neuron y_k. The $\Re^n \rightarrow \Re^m$ function the net calculates is $F(x_1,\ldots,x_n) = (y_1,\ldots,y_m)$ where

$$y_k = g_A\left(\sum_{j=1}^{h} z_j \beta_{jk}\right) \tag{1}$$

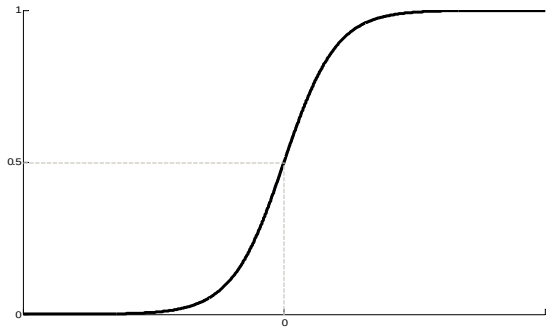
with

$$z_j = f_A\left(\sum_{i=1}^{n} x_i w_{ij} + \tau_i\right) \tag{2}$$

Where g_A and are activation functions. In several applications the output activation functions are linear ones, and the usual choice for the hidden activation function is the logistic function: $f_A(x)=1/(1+e^{-x})$ (*Figure 2*).

Figure 2

Logistic function



2. ábra: Logisztikus függvény

NN visualization methods

The output of the hidden neurons z_j can be seen as an h dimensional vector that represents the range the neurons work in. If a 'hidden variable' z_j is close to zero or one, the neuron is saturated (*Figure 2*). If a hidden neuron gives values near zero or one for almost all inputs, hence it does not fire or fires all the time, it is useless for the problem. The distribution of these h dimensional data can represent the NN behavior for a human expert. Unfortunately, in several cases there is a need for more than two or three hidden neurons. In these cases a projection or dimensionality reduction technique has to be used. Principal Component Analysis (PCA) is a linear technique; therefore the information loss may be more than the admissible level. Other (topology or distance preserving) projection techniques like Multidimensional Scaling, Sammon method, Isomap or Locally Linear Embedding can be used for that purpose. For more details see *Abonyi* (2007) and the references within.

However, there are some special visualization methods for NNs. *Duch* (2003) proposed an approach for visualization of NNs applied on classification problems. His method can be applied for problems with K classes if the output is coded as a K length vector: (1,0,…,0) means the first class, (1,0,…,0) the second and so on. In this case the classes are represented by the corners of the K dimensional unit hypercube. The approach proposed by Duch maps the *NN output* into two dimensions, basically 'flattens' the hypercube into two dimensions. This approach was thought over and applied on *the output of the hidden neurons* z_j in *Duch* (2004a, 2004b). This method was straightforward from the former one because the hidden variables (the activation functions) take values from [0,1], therefore the h dimensional vectors are located within the unit hypercube. This method can be used not only for classification but also for function approximation purposes as well. Based on this latter approach a picture of the behavior of the hidden units, their firing strength and activation or saturation level can be obtained. The main drawback is that the number of classes/hidden neurons is limited. To keep the figures simple and interpretable, only 3…6 variables can be used.

Interpretation of NNs

Another strategy for 'opening' a NN is to convert it into a rule based model. These 'linguistically sound' rules are often fuzzy if-then rules, and are close to human thinking: IF *a set of conditions is satisfied*, THEN *a set of consequences is inferred*. Fuzzy logic provides a tool to process uncertainty, hence fuzzy rules represents knowledge using linguistic labels instead of numeric values, thus, they are more understandable for humans and may be easily interpret (*Benitez*, 1997). If NNs can be transformed into rules, then it makes possible to overlook and validate the trained NN, and build in a priori knowledge to the network. The crucial question is what the connection is between the several types of neural networks and fuzzy rule based systems.

Under some conditions, the equivalence of normalized radial basis function networks (RBF) and Takagi-Sugeno fuzzy models can be obtained (*Nelles*, 2001). However, in this paper multilayer perceptron (MLP) type neural networks with logistic hidden activation function (see Structure of neural networks subsection) are dealt with (in the following the notation NN will be used for MLP type networks). An approach for NNs with **tanh** activation function is presented in *Setiono* (2000) for function approximation purposes, but it should be noted that it is an approximation: the rule based model is not identical to the original trained NN, therefore information transfer in the 'opposite' direction, i.e. from the rule base to the NN can be problematic. An interesting result was given by *Benitez* (1997) who proved the equality of NNs with logistic activation function and a certain type of fuzzy rule based model called fuzzy additive system (FAS). For that purpose, a new fuzzy logic operator had to be introduced. Because of the equality (which is stronger than equivalence), if a method can be applied on a FAS for a certain purpose (e.g. rule base reduction), then it is also applicable to the NN as well and vice versa. In the following, this equality relation is discussed based on *Benitez* (1997).

FAS employs rules in the following form:

$$R_{jk}\text{: } \mathbf{If}\ x_1 \text{ is } A_{jk}^1 \text{ and } \ldots \text{ and } x_n \text{ is } A_{jk}^n \ \mathbf{then}\ y_k \text{ is } p_{jk}(x_1,\ldots,x_n) \tag{3}$$

where $p_{jk}(x_1,\ldots,x_n)$ is a linear function on the inputs. In FAS's, the inference engine works as follows: for each rule, the fuzzified inputs are matched against the corresponding antecedents in the premises giving the rule's firing strength. It is obtained as the *t*-norm (usually the minimum operator) of the membership degrees on the rule if-part. The overall value for output y_k is calculated as the weighted sum of relevant rule outputs. Let us suppose multi-intput single-output fuzzy rules, having l_k of them for k th output. Then y_k is computed as

$$y_k = \sum_{j=1}^{l_k} v_{jk} \cdot p_{jk}(x_1,...,x_n) \tag{4}$$

where v_{jk} is the firing strength of j th rule for k th output.

To decompose the multivariate logistic function to form the rule antecedents in the form of (3) with univariate membership functions, a special logic operator has to be used instead of *and*: interactive-or or *i*-or:

$$a * b = \frac{a \cdot b}{(1-a)\cdot(1-b) + a \cdot b}\cdot \tag{5}$$

To get a clearer idea of *i*-or behavior, see *Figure 3*, which represents the surface defined by the *i*-or operator. Using this * operator, the interpretation of NNs whose hidden neurons have biases as follows. It can be checked that

$$f_A\left(\sum_{i=1}^{n} x_i w_{ij} + \tau_j\right) = f_A\left(x_1 w_{1j} + \tau'_j\right) * \ldots * f_A\left(x_n w_{nj} + \tau'_j\right) \tag{6}$$

where $\tau'_j = \tau_j/n$. In (6), the first term corresponds to the fuzzy proposition "$\sum_{i=1}^{n} x_i w_{ij} + \tau_j$ is A". Likewise, $f_A(x_i w_{ij}+\tau'_i)$ corresponds to proposition "$x_i w_{ij}+\tau'_i$ is A" or in a similar form "$x_i w_{ij}$ is A-τ'_i". Hence, the bias term means a sheer translation. The A^i_{jk} fuzzy sets have to be redefined to account for both the weight w_{ij} and the bias τ_j. Their membership function is defined by

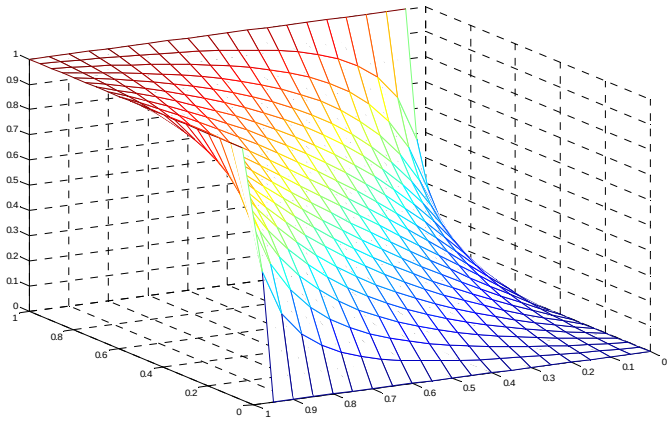
$$\mu_{A^i_{jk}}(x) = \mu_A[(x+\tau'_j)w_{ij}] \tag{7}$$

Based on that, the fuzzy rules extracted from the trained NN are:

$$R_{jk}: \textbf{If } x_1 \text{ is } A^1_{jk} * \ldots * x_n \text{ is } A^n_{jk} \textbf{ then } y_k = \beta_{jk} \tag{8}$$

Figure 3.

Interactive-or operator



3. ábra: Az i-or operátor

An interesting and useful application possibility is to initialize the NN on the basis of a priori knowledge. Initialization is a crucial question by NNs because there are often a huge number of parameters and the cost function has numerous local minima. The most often applied local (gradient based) search techniques may trap in a local minimum. To avoid that problem, a possible approach is multi-start method, i.e. to train the NN from several different (random) initial points. Other solution can be based on evolutionary algorithms, see e.g. *Castro* (1998). The flexibility of evolutionary algorithms makes possible the direct rule extraction from trained NNs (however, only crisp rules and by classification problems) as *Markowska-Kaczmar* (2003) shows. However, all of these

latter methods have high computational demand. The initialization using prior knowledge based if-then rules has other advantage as well: it combines the user's experience with the learning capability of NN.

Complexity reduction

In this subsection we focus on the combination of existing model reduction techniques with the previously presented rule based model extraction method. It is a wide research area, the interested reader can refer *Setnes* (2001) and the citations within. In general it can be stated that linear model reduction methods are preferred to nonlinear ones because they are exhaustively studied and effectively applied for several types of problems (e.g. in controller assessment recently in *Harris* (2007)). For that purpose the model should be linear in parameters.

A possible method family is orthogonal techniques. These methods can roughly be divided into two groups: the rank revealing ones like SVD-QR algorithm and those that evaluate the individual contribution of the rule or local models, like the orthogonal least-squares approach (OLS). This later technique requires more computations, but for system identification purposes it is preferable as it gives a better approximation result. In the remaining part of this paper OLS is applied for rule ranking and model reduction purposes. OLS works as follows (for a throughout discussion see *Nelles* (2001)). Consider a general linear in parameters model:

$$y=Z\theta+e \tag{9}$$

where $y=[y_1,\ldots,y_n]^T$ is the measured output, $Z=[z_1,\ldots,z_n]^T$ is the regressor matrix ($z_i=[z_{i1},\ldots,z_{iN}]^T$, $i=1,\ldots,h$ are the regressors), $\theta=[\theta_1,\ldots,\theta_h]$ is the parameter vector and $e=[e_1,\ldots,e_N]^T$ is the prediction error. OLS transforms the columns of the regressor matrix Z into a set of orthogonal basis vectors in order to inspect the individual contribution of each regressor. If they were not orthogonal, they could not been inspected individually. An orthogonalization method should be used to perform the orthogonal decomposition $Z=VR$ (often the simple Gram-Schmidt method is used), where V is an orthogonal matrix such that $V^TV=I$, and R is an upper triangular matrix with unity diagonal elements. Substituting Z= into (9), we get $y=VR\theta+e=Vg+e$ where $g=R\theta$. Since the columns v_i of V are orthogonal, the sum of squares of y_k can be written as

$$\mathbf{y}^T\mathbf{y}=\sum_{i=1}^{h} g_i^2\mathbf{v}_i^T\mathbf{v}_i+\mathbf{e}^T\mathbf{e}. \tag{10}$$

The part of the output variance $\mathbf{y}^T\mathbf{y}/N$ explained by regressors is $\sum_{i=1}^{h} g_i^2\mathbf{v}_i^T\mathbf{v}_i/N$, and an error reduction ratio due to an individual regressor *i* can be defined as

$$err_i=\frac{g_i^2\mathbf{v}_i^T\mathbf{v}_i}{\mathbf{y}^T\mathbf{y}},\ i=1,\ldots,h \tag{11}$$

This ratio offers a simple means of ordering the regressors. As *Nelles* (2001) shows, "there are only two restrictions to the application of this subset selection technique. First, the model has to be linear in parameters. Second, the set of regressors from which the significant ones will be chosen must be precomputed." This later one is an important restriction because it means that all regressors are fixed during this procedure. By normalized RBF networks and Takagi-Sugeno fuzzy models (see Interpretation of NNs subsection) this requirement is not met, therefore the original version of OLS cannot be applied. It is because the normalization denominator changes as the number of selected rules changes, thus the fuzzy basis functions (here: regressors) change. To overcome this

problem the value of the denominator can be fixed, but in this case interpretability issues are discarded completely. However, OLS can be very useful for various purposes; modified versions of OLS can also be applied to determine the centers of radial basis functions (*Huang*, 2005), or to generate Takagi-Sugeno-Kang fuzzy models (*Mastorocostas*, 2001, 2003).

In case of MLP networks and FAS systems, this problem does not occur because of the special output computing mechanism (4). Thus classical OLS can be applied on FAS systems to rank the rules since the parameters of the trained NN are fixed. However, OLS is formulated as a MISO technique. If the NN has more than one output, then the outputs can be evaluated individually one by one. In this case (using the notation of OLS (9-11)), **y** is the k th network output, the regressors $\mathbf{z}_i$ are the outputs of the hidden neurons, and the parameters θ_j corresponds to the weights from the j th hidden neuron to the k th output neuron β_{jk} (see also (1)). This approach was directly applied on NNs in *Henrique* (2000), and it was shown that analog method can be applied to the subset selection of the original network inputs. In this case in (9-11), y is the output of the k th hidden neuron, the regressors $\mathbf{z}_i$ are the inputs of the network, and the parameters θ_j corresponds to the weights from the j th input neuron to the k th hidden neuron w_{jk} (see also (2)). Other NN pruning can also be considered, e.g. optimal brain damage (*Cun*, 1990) or optimal brain surgeon (*Hassibi*, 1992), and it should be emphasized that these methods can directly be applied on FAS systems as well. The application examples in the next section show that it can be very effective if a model reduction technique (in this paper OLS for ranking the rules) and rule base extraction from NN are applied together, and validate the identified models by human experts.

It should be noted that the applied i-or operator in the extracted fuzzy rules does not belong to the commonly applied fuzzy t-norms or t-conorms. However, it would be interesting to test the extracted fuzzy rules with common fuzzy logic operators, and maybe recompute the output weights (which can easily be done because the model is linear with respect to these parameters). Our presumption is that the crisper the activation functions are (f_A), the less the difference is between the modeling performances of the original and the modified FAS's that uses classical fuzzy logic operators. For that purpose, numerous tests have to be completed in the future. If this guess proves true, then the cost function of the NN during training can be modified to get 'crisper' activation functions.

APPLICATION EXAMPLES AND DISCUSSION

Benchmark datasets are extremely helpful to consider the possibilities of the presented reduction and transformation techniques. We used a dataset of a pH process, where the concentration of hydrogen ions in a continuous stirred tank reactor is modeled (CSTR). This well-known modeling problem presents difficulties due to the nonlinearity of the process dynamics. This process can be correctly modeled as a first-order input-output(NARX) system, where the actual output (the pH) $y(k+1)$, depends on pH of the reactor $y(k)$ and the NaOH feed $u(k)$ at the kth sample time (sample time is t_s=0.2 min):

$$y(k+1) = f(y(k), u(k)) . \tag{12}$$

For detailed information of the process see *Abonyi* (2003).

Parameters of the neural network were identified by the back-propagation algorithm based on a uniformly distributed training data where F_{NaOH} is in the range of 515-525 l/min. Experiences pointed on, that for good model performance 7 neurons are

sufficient in the hidden layer of the neural network. The results in *Table 1* shows, that the neural network models give very good prediction performance for this process.

Table 1.

Modeling and testing errors (one-step ahead prediction)

Testcase (number of neurons in hidden layer/number of removed neurons) (1)	Training errors (MSE) (2)	Testing Errors (MSE) (3)
Neural Network (7) (4)	3.088e-005	3.267e-005
Using i-or (7) (5)	3.053e-005	3.259e-005
Network reduction (8/1 neuron) (6)	4.434e-005	4.285e-005
Network reduction (7/1 neuron) (7)	3.060e-005	3.247e-005
Network reduction (6/2 neuron) (8)	2.884e-004	3.690e-004
Network reduction (6/1 neuron) (9)	1.086e-004	1.316e-004

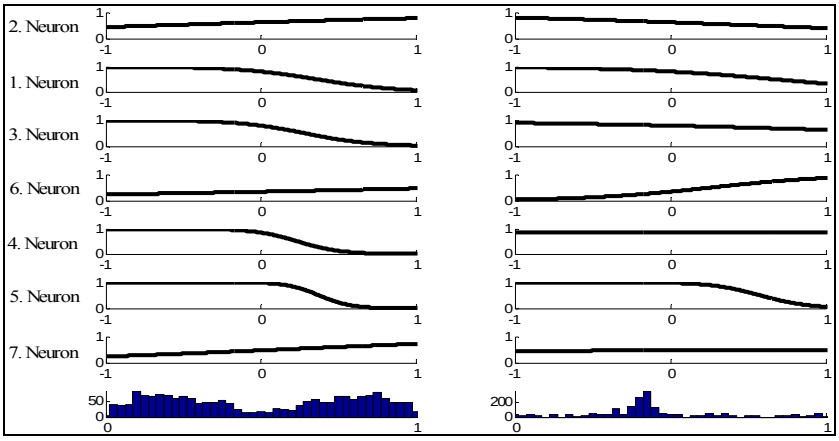
1. táblázat: Modellezési és validálása hibák egylépéses predikcióra

Teszteset (neuronok száma a rejtett rétegben/eltávolított neuronok száma)(1), Hiba a tanítási eseteken (MSE - Átlagos Négyzetes Hiba)(2), Hiba a tesztelési eseteken (MSE - Átlagos Négyzetes Hiba)(3) Neurális hálózat(4), I-or operátor használata(5), Neurális hálózat redukció(8/1 neuron)(6), Neurális hálózat redukció(7/1 neuron)(7), Neurális hálózat redukció(6/2 neuron)(8), Neurális hálózat redukció(6/1 neuron)(9)

However the deviation between training and testing prediction errors of the original neural network model reveals that the model is a bit overparameterized (there are too many neurons in the hidden layer of the network) and the test error of the models could be lowered by model reduction.

Figure 4

Decomposed univariate membership functions



4. ábra: Egyváltozós dekomponált tagsági függvények

Applying the described transformation technique, *Figure 4* shows the decomposed univariate membership functions. Since the neural network model of the pH process contains seven neurons in the hidden layer of the model the transformed FAS consists of seven fuzzy rules according to equation (6). The neurons are listed according to the OLS ranking (see equation (13)), starting with the rules decomposed from the most important neuron in the hidden layer of the network. For better interpretability the histogram of the corresponding model inputs are illustrated on the last two subplots. Results in the first two rows of *Table 1* show that the transformed fuzzy rulebase using i-or is identical with the original neural network as expected, because the modeling errors are almost the same both for testing and training results. However a small difference between the training and testing errors exists, which is due to the re-identification of parameters for the output layer after applying *i*-or. There is a small decrease in modeling errors with the re-identification. In the last four rows of *Table 1*, the numbers in parenthesis are the number of the neurons used in the hidden layer of the neural network model In parenthesis the second number means the number of the reeducated (removed) neurons if there is any.

As it was mentioned, ordering the neurons by OLS estimated error reduction ratios reveals the unnecessary neurons (the importance of the extracted rules) in the hidden layer, because neurons with low error reduction ratio are insignificant for the appropriate model. In the meaining of the FAS equivalent of the neural network the OLS ranking means a reduction based on the consequent of the fuzzy rule. It is also possible to reduce the FAS rule base by analyzing the antecedent part of the rules using a similarity measure for the membership functions. Utilizing the equality of FAS and NN, the following classical interclass separability measure (originally for fuzzy systems) could be used to compare the univariate functions decomposed from hidden neurons:

$$S^{i}{}_{j,l,k} = \frac{\int \min(A^{i}_{j,k}(x_i), A^{i}_{l,k}(x_i))dx_i}{\int \max(A^{i}_{j,k}(x_i), A^{i}_{l,k}(x_i))dx_i},\ i = 1,...,n;\ j,l = 1,...,h \tag{13}$$

According to (13) the similaritiy measure to compare the hidden neurons with multivariate activation functions can be defined:

$$S_{j,l,k} = \prod_i (S^{i}{}_{j,l,k}),\ i = 1,...,n;\ j,l = 1,...,h \tag{14}$$

The similarity results of the neurons are illustrated in *Table 2*. The achieved similarity measures correspond to the membership functions depicted on *Figure 4*, consequently the 7th and the 2th neurons are the most similar.

Figure 5 illustrates the neurons ordering by their effect on the modeling error. This ordering is also indicated on *Figure 4*, so the most important rules are on the top of the picture. The consequence of synthesizing the results is that it is possible to remove one neuron out of 7 (in FAS the corresponding rules) from the model without a significant increase in modeling performance, because of the low error reduction rate of the last, 7th neuron. This achievement harmonizes with the issues of *Table 2*, because the interclass separability identifies the 2nd and the 7th neuron as similar, but OLS indicates the 2nd one as more important.

As *Table 1* shows that model reduction techniques like OLS makes it possible to overcome the problem of overfitting and the performance of the reduced model is almost the same as the original one. A rigorous test of NARX models is free run simulation

because the errors can be cumulated. The following result indicates the goodness of the reduced model even by free run simulation:

- 3.508e-003 for neural network with 7 neurons;
- 3.828e-003 using *i*-or for FAS with 7 rules;
- 3.824e-003 after removing 1 neuron from the hidden layer containing 7 neurons.

Table 2.

Interclass separabilities for the neurons in the hidden layer

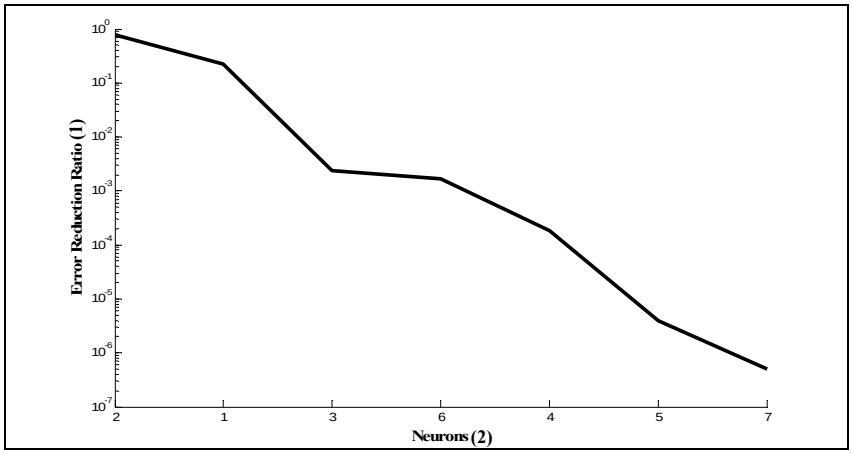
Neurons (1)	1	2	3	4	5	6	7
1	1.00	0. 44	0. 67	0. 53	0. 35	0.44	0.39
2		1.00	0. 32	0. 26	0. 40	0.20	**0.75**
3			1.00	0. 47	0.36	0.63	0.30
4				1.00	0.20	0.43	0.21
5					1.00	0.26	0.49
6						1.00	0.19
7							1.00

2. Táblázat: Hasonlósági értékek az egyes neuronok között

Neuronok száma a rejtett rétegben (1)

Figure 5

Error reduction ratios



5. ábra: Hibaredukciós arány

Az egyes neuronokhoz rendelhető hibaredukciós arányok(1), Rejtett rétegbeli neuronok(2)

After transforming the network into a FAS, it is also possible to use similarity measures which can be used to reduce further the rule base. It can be done in an automatic way if a

threshold value is defined previously. If the measured similarity is greater than the threshold, the corresponding two neurons in the original neural network can be considered as identical; therefore further reduction of the FAS rule base is possible. This technique can be used even in the learning process of the neural network.

SUMMARY

This article gave brief summary of visualization of the neural networks, and discussed that neural network with logistic function $f_A(x)=1/(1+e^{-x})$ is identical to fuzzy additive systems. The neural networks are often too complex (overtrained) and not interpretable, therefore it is very difficult to utilize these networks correctly. A possible solution is to visualize and/or reduce these models.

The article showed how model reduction techniques can be used by neural networks. For that purpose an orthogonal technique, the orthogonal least-squares approach (OLS) was used. It was shown that after transforming the network into an equivalent fuzzy additive system it is possible to reduce the network by analyzing the antecedent part of the fuzzy rules.

A possible future research area is to develop a new learning procedure for neural networks using prior knowledge based if-then rules, which combines the user's experience and/or constraints with the learning capability of NN. The extracted fuzzy rules are planned to be tested with common fuzzy logic operators to confirm our presumption that the crisper the activation functions are (f_A), the less the difference is between the modeling performances of the original and the modified FAS's that uses classical fuzzy logic operators.

ACKNOWLEDGMENT

The authors would like to acknowledge the support of the Cooperative Research Centre (VIKKK, project III/1) and Hungarian Research Found (OTKA T049534). János Abonyi is grateful for the support of the Bolyai Research Fellowship of the Hungarian Academy of Sciences and the Öveges Fellowship.

REFERENCES

J. Abonyi, B. Feil (2007). Aggregation and Visualization of Fuzzy Clusters based on Fuzzy Similarity Measures. Advances in Fuzzy Clustering and its Applications, John Wiley & Sons, 95-123.

J. Abonyi (2003). Fuzzy Models of Dynamical Systems. Fuzzy Model Idetiftcation for Control, Birkhauser, 60-61.

J.M. Benitez, J.L. Castro, I. Requena (1997). Are artifical neural networks black boxes? IEEE Transactions on Neural Networks, 8. 5. 1156-1164.

L.N. de Castro, E.M. Iyoda, F.J.V. Zuben, R. Gudwin (1998). Feedforward Neural Network Initialization: an Evolutionary Approach. Proceedings of the 5th Brazilian Symposium on Neural Networks, 43-49.

Y.L. Cun, J. Denker, S. Solla (1990). Optimal brain damage. Advances in neural information processing systems, 2. 598-605.

W. Duch (2003). Coloring black boxes: visualization of neural network decisions. Int. Joint Conf. on Neural Networks, Portland, Oregon, IEEE Press, 1. 1735-1740.

W. Duch (2004). Visualization of Hidden Node Activity in Neural Networks: I. Visualization Methods. Lecture Notes in Artificial Intelligence, 3070.38-43

W. Duch (2004). Visualization of hidden node activity in neural networks: II. Application to RBF networks. Lecture Notes in Artificial Intelligence, 3070. 44-49.

T.J. Harris, W. Yu (2007). Controller assessment for a class of non-linear systems. Journal of Process Control, in press.

B. Hassibi, D. Stork, G. Wolff (1992). Optimal brain surgeon and general network pruning. Technical Report 9235, RICOH California Research Center, Menlo Park, CA.

H.M. Henrique, E.L. Lima, D.E. Seborg (2000). Model structure determination in neural network models. Chemical Engineering Science, 55. 5457-5469.

D.S. Huang, W.B. Zhao (2005). Determining the centers of radial basis probabilistic neural networks by recursive orthogonal least square algorithms, Applied Mathematics and Computation, 162. 461–473.

U. Markowska-Kaczmar, M. Chumieja (2003). Opening neural network black box by evolutionary approach. Design and application of hybrid intelligent systems, 147-156.

P.A. Mastorocostas, J.B. Theocharis, V.S. Petridis (2001). A constrained orthogonal least-squares method for generating TSK fuzzy models: Application to short-term load forecasting. Fuzzy Sets and Systems, 118. 215-233.

P.A. Mastorocostas, J.B. Theocharis (2003). An orthogonal least-squares method for recurrent fuzzy-neural modeling. Fuzzy Sets and Systems, 140. 285-300.

O. Nelles. (2001) Nonlinear system identification. Springer-Verlag.

R. Setiono, W.K. Leow, J.Y.L. Thong (2000). Opening the neural network black box: an algorithm for extracting rules from function approximating artificial neural networks. Proceedings of the 21st International Conference on Information systems, Queensland, Australia, 176-186.

M. Setnes (2001). Complexity reduction in fuzzy systems. PhD Thesis, TU Delft.

Corresponding author (*Levelezési cím*):

János Abonyi
University of Pannonia, Department of Process Engineering
H-8201 Veszprém, P.O.Box 158
Pannon Egyetem, Folyamatmérnöki Intézeti Tanszék
8201 Veszprém, Pf. 158
Tel.: 36-88-624-209, Fax: 36-88-624-171
e-mail: abonyij@fmt.uni-pannon.hu