ERROR-BOUND OF THE SVD-BASED NEURAL NETWORKS

Orsolya Takács*, István Nagy**

*Dept. of Measurement and Information Systems
Budapest University of Technology and Economics
Műegyetem rkp. 9., Budapest, H-1521 Hungary
Phone: +36 1 463 2057, Fax: +36 1 463 4112
takacs@mit.bme.hu
**Dept. of Telecommunications and Telematics
Budapest University of Technology and Economics
inagy@ttt.bme.hu

Abstract: Singular Value Decomposition -based complexity reduction was first proposed for various fuzzy inference systems. Recently, the method has been extended to generalized neural networks, which made possible the use of neural networks in time-critical systems. Beyond the elimination of redundancy, the SVD-based reduction could be used to make further reduction, if a certain amount of error could be tolerated. This paper gives an error-bound for this further complexity reduction of generalized neural networks.

Keywords: neural networks, neural-network-models, real-time AI, complexity reduction, real-time systems

1. INTRODUCTION

The main advantage of neural networks (NNs) is, that they are able to solve complicated problems, even if the exact mathematical model is not known. For example, if enough measured data is available from an industrial process, a neural model of the process could be constructed, even if the mathematical models, describing the process are unknown.

However, there are no universal methods for the approximation of the proper size of the neural networks. To achieve a good approximation one could be tempted to overestimate the size of the neural networks, with too many nodes, layers and links. The result is a large and redundant networks, with too high complexity and need of computational power and storage. All these ask for formal methods for the complexity reduction of neural networks.

The Singular Value Decomposition (SVD) based complexity reduction was first proposed for the reduction of fuzzy rule-bases (Yam, 1997; Baranyi and Yang, 1997a; Baranyi and Yam, 1997; Baranyi *et al.*, 1999; Yam *et al.*, 1999) and recently has been extended to generalized neural networks (Baranyi *et al.*, 2000a,b).

The generalized neural network type, for which the reduction method is proposed, is general in the sense that it has no transfer function in the neurons, but has weighting functions instead of constant values defined on the connections among the layers. The

proposed method is applicable not only to forward networks, but to nets consisting of feedback connections from all to all neurons contained in all layers. In the present work the weighting functions are approximated by fuzzy logic based techniques which has been recently introduced in the literature as neuro-fuzzy algorithm (Rojas, 1996; Hornik, 1989). The reason of its wide adoption is the good approximation property, simplicity, and its ability to training which has been exploited in applications (see e. g. (Baranyi *et al.* 1998, 2000c)).

Though, the SVD-based complexity reduction algorithm could be used to eliminate the redundancy of generalized neural networks, the need arises to make further, non-exact complexity reduction, allowing a certain amount of error. This could be carried out by discarding also non-zero singular values. Because the amount of the arising error must be estimable, the aim of this paper to give error-bounds for SVD-based complexity-reduction of generalized neural networks.

Section II. describes the generalized neural network algorithm. Section III. and IV. shows the reduction of two types of these neural networks (with singular and non-singular consequences) and gives an error-bound for the reduction, while Section V. summarizes the results and propose some possible application area.

2. GENERALIZED NEURAL NETWORK

The classical multilayer neural network could be generalised, if the non-linear transfer function is moved from the nodes into the links. It results in neurons, which apply only a sum operation to the input values, and links, which are characterised by possibly non-linear weighting functions, instead of a simple constant weights.

Let us focus on two neighbouring layers l and l+1 of a forward model. Let the neurons be denoted as $N_{l,i}$, $i=1..n_l$ in layer l, where n_l is the number of neurons. Further, let the input values of $N_{l,i}$ be $x_{l,i,k}$, $k=1..n_{l-1}$ and its output $y_{l,i}$. The connection between two layers (l and l+1) can be defined by matrix C_l, which consists of weighting functions $f_{l,j,i}(y_{l,i})$, where $j=1..n_{l+1}$. Thus

$$x_{l+1,j,i} = f_{l,j,i}(y_{l,i}) \qquad (1)$$

The neurons apply only sum operations to the input values. For instance, the output of $N_{l+1,j}$ is:

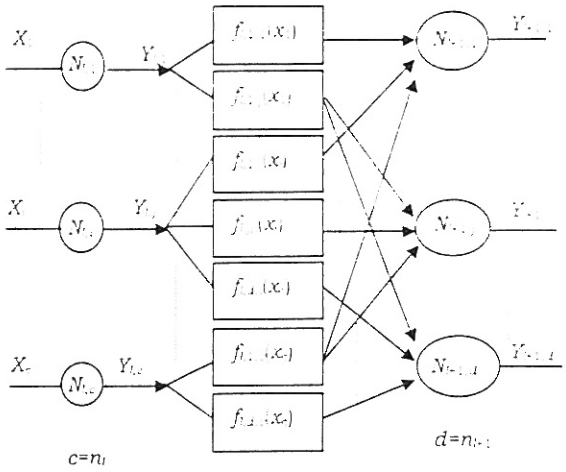

Fig. 1: General type neural network

$$y_{l+1,j} = \sum_{i=1}^{n_l} x_{l+1,j,i} .$$

Therefore, from (1) neuron $N_{l+1,j}$ yields

$$y_{l+1,j} = \sum_{i=1}^{n_l} f_{l,j,i}(y_{l,i}) . \qquad (2)$$

The general form is depicted on Figure 1.

Because not only the classical weights, but the weight functions could also be trained, generalised neural networks has a good approximation property. However, having various types of weighting functions, it may need considerable computational effort in contrast to the standard type neural networks and its training, namely, the searching of the unknown weighting functions may lead to a complicated or even unsolvable mathematical problem. One natural solution of the training is to replace the unknown weighting functions with linearly combined known functions, where only the factors of the linear combination is to be trained. It practically means that a parallel layer is added to the output. From (2):

$$y_{l+1,j} = \sum_{i=1}^{n_l} \sum_{t=1}^{m} f_{l,j,i,t}(y_{l,i}) b_{l,j,i,t} \qquad (3)$$

where m is the number of the known $f_{l,j,i,t}(y_{l,i})$ functions and values $b_{l,j,i,t}$ are trained. Obviously it results in the approximation of the original network. This approximation could be treated as the approximation of the unknown weighting functions with one-input, one-output fuzzy inference systems.

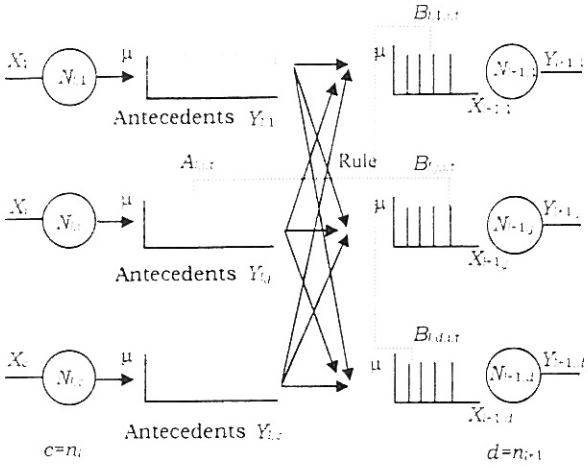

Figure 2: Approximation of the generalized network.

Definition 1.: Product-Sum-Gravity fuzzy inference (PSG)

The antecedent fuzzy sets $A_t : \mu_t(x)$, $x \in X$, $t = 1..m$ are defined on universe X in *Ruspini*-partition, where m is the number of antecedent sets. The fuzzification method is singleton, and during the inference product T-norm, sum S-norm and centre-of-gravity defuzzification is used.

If the rules are

$$\text{If } A_t \text{ then } B_t$$

and the consequent fuzzy sets are singleton, crisp sets (PSGS fuzzy inference), then the result of the inference is

$$y^* = \sum_{t=1}^{m} \mu_t(x^*) b_t \tag{4}$$

In order to approximate the contribution of each neuron to the neurons of the next layer by the PSG technique, let be Eq. (4) substituted into (2). Let be defined a more general form, where all antecedent universes may have different number of antecedent sets:

$$y_{l+1,j} = \sum_{i=1}^{n_l} f_{l,j,i}(y_{l,i}) = \sum_{i=1}^{n_l} \sum_{t=1}^{m_{l,i}} \mu_{l,i,t}(y_{l,i}) b_{l,j,i,t} \tag{5}$$

where $m_{l,i}$ is the number of antecedent sets in layer l. Figure 2 depicts a neuro-fuzzy network representing (5).

As a matter of fact, if $\forall i : m_{l,i} = m$ then this algorithm leads to the same as summing up the outputs of m generalized networks connected parallel:

$$y_{l+1,j} = \sum_{i=1}^{n} \sum_{t=1}^{m} \mu_{l,i,t}(y_{l,i}) b_{l,j,i,t} = \\ = \sum_{t=1}^{m} \sum_{i=1}^{n_l} \mu_{l,i,t}(y_{l,i}) b_{l,j,i,t} = \sum_{t=1}^{m} \sum_{i=1}^{n_l} f_{l,i,t}(y_{l,i}) \tag{6}$$

The benefit is that the arbitrary type weighting functions which are unknown before the training, are replaced usually with rather simple known functions and only their linear weighting must be trained.

3. Complexity Reduction and Error-Bound of the Singleton-Based Generalized Algorithm

3.1. Basic algorithm

The heart of the complexity reduction algorithm is the reduction procedure based on the Singular Value Decomposition (RSVD). Any $\underline{\underline{F}}$ matrix could be decomposed as:

$$\underset{=}{F}_{(n_1 \times n_2)} = \underset{=}{A}_{1,(n_1 \times n_1)} \underset{=}{B}_{(n_1 \times n_2)} \underset{=}{A}^{T}_{2,(n_1 \times n_2)}, \tag{7}$$

where $\underline{\underline{A}}_k$ are orthogonal matrixes ($\underline{\underline{A}}_k \underline{\underline{A}}_k^T = \underline{\underline{E}}$), and $\underline{\underline{B}}$ contains the λ_i singular values of $\underline{\underline{F}}$ in decreasing order. The maximum number of the nonzero singular values is $n_{SVD} = \min(n_1, n_2)$. The singular values indicate the significance of the corresponding columns of $\underline{\underline{A}}_k$. Let the matrixes be partitioned in the following way (Yam, 1997):

$$\underset{=}{A}_k = \left| \underset{=}{A}^{r}_{k,(n_k \times n_r)} \quad \underset{=}{A}^{d}_{k,(n_k \times n_d)} \right| \text{ and}$$

$$\underset{=}{B}_k = \begin{vmatrix} \underset{=}{B}^{r}_{(n_r \times n_r)} & 0 \\ 0 & \underset{=}{B}^{d}_{((n_1 - n_r) \times (n_2 - n_r))} \end{vmatrix}.$$

If $\underline{\underline{B}}^d$ contains only zero singular values, then $\underline{\underline{B}}^d$ and $\underline{\underline{A}}_k^d$ could be dropped: $\underline{\underline{F}} = \underline{\underline{A}}_1^r \underline{\underline{B}}^r \underline{\underline{A}}_2^{rT}$.

If $\underline{\underline{B}}^d$ contains nonzero singular values, as well, then the $\underline{\underline{F}}' = \underline{\underline{A}}_1^r \underline{\underline{B}}^r \underline{\underline{A}}_2^{rT}$ matrix will be only an approximation of $\underline{\underline{F}}$, and the maximal difference between the values of $\underline{\underline{F}}$ and $\underline{\underline{F}}'$ (Yam *et al.*, 1999):

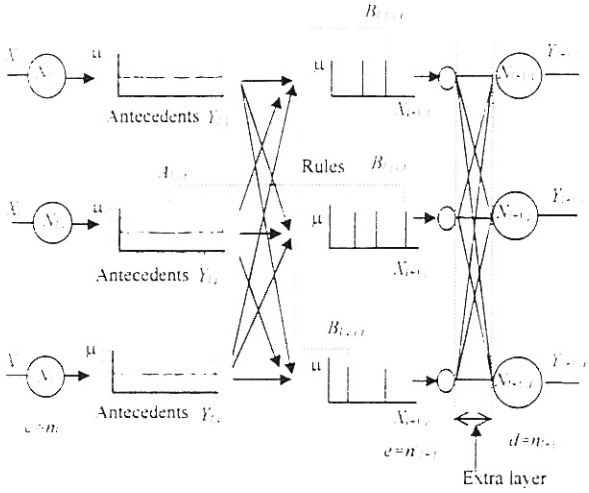

Figure 3: Reduced neural network

$$E_{PSTD} = \left| \underline{\underline{F}} - \underline{\underline{F}}' \right| \le \left(\sum_{i=n_r+1}^{n_{STD}} \lambda_i \right) \underline{\underline{1}}_{=(n_1 \times n_2)} . \qquad (8)$$

Further transformations could be made to ensure certain properties of the matrices, if needed (sum-normalization (SN): the sum of each row equals one; non-negativeness (NN); normalization (NO): there is an element equal to one in every column).

3.2 Complexity reduction of the generalized neural-network

Equation (5) could always be transformed into the following form:

$$y_{l+1,j} = \sum_{z=1}^{n^r_{l+1}} a_{l,j,z} \sum_{i=1}^{n_l} \sum_{t=1}^{m^r_{l,i}} \mu^r_{l,i,t}(y_{l,i}) b^r_{l,z,i,t} \qquad (9)$$

where "r" denotes "reduced", furthermore $n^r_{l+1} \le n_{l+1}$ and $\forall i : m^r_{l,i} \le m_{l,i}$.

The reduced form can be represented as a neural network with an extra inner layer between layers l and $l+1$ (Fig. 3.). Between the original layer l and the new layer the weighting functions are approximated from the reduced PSGS fuzzy systems, and layer $l+1$. simply computes the weighted sum of the outputs of the new layer, with weighting factor $a_{l,j,z}$. The reduction could be made in two steps: in the first, the $a_{l,j,z}$ values are determined, and in the second step the new membership functions are determined.

Step 1.: Determination of the $a_{l,j,z}$ values

Let $\underline{\mu}_0$ and $\underline{\underline{S}}_l$ be defined as:

$$\underline{\mu}_0 = \left[\mu_{1,1}(y_{1,1}) \cdots \mu_{1,m_1}(y_{1,1}) \cdots \mu_{n_1,m_{n_1}}(y_{1,n_1}) \right]^T$$

$$\underline{\underline{S}}_l = \begin{bmatrix} b_{l,1,1,1} & \cdots & b_{l,1,1,m_1} & & b_{l,1,n_1,1} & \cdots & b_{l,1,n_1,m} \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ b_{l,n_{l-1},1,1} & \cdots & b_{l,n_{l-1},1,m} & & b_{l,n_{l-1},n,1} & \cdots & b_{l,n_{l-1},n,m} \end{bmatrix}$$

where, $\underline{\mu}_0$ (length: $\sum_{i=1}^{n_r} m_{l,i}$) contains the values of all of the membership functions in a given $(y_{1,1}, \ldots, y_{1,n_1})$ point, and $\underline{\underline{S}}_l$ contains all $b_{l,i,j,k}$ values. Its size is $n_{l+1} \times \sum_{i=1}^{n_r} m_{l,i}$.

With this notation:

$$\underline{y}_{l+1} = \underline{\underline{S}}_l \underline{\mu}_0 \qquad (10)$$

Applying the above described singular value-based reduction to the $\underline{\underline{S}}_l$ matrix one will get:

$$\underline{\underline{S}}_l \approx \underline{\underline{A}}_l \underline{\underline{D}}_l \underline{\underline{V}}_l = \underline{\underline{A}}_l \underline{\underline{S}}'_l . \qquad (11)$$

The error of the reduction will be the sum of the discarded singular values, E_1:

$$\left| b_{l,j,i,t} - \sum_{z=1}^{n^r_{l+1}} a_{l,j,z} b'_{l,z,i,t} \right| \le E_1 , \qquad (12)$$

where $b_{l,j,i,t}$ are the elements of $\underline{\underline{S}}'_l$. The outputs of the reduced network after the first step are:

$$\underline{y}'_{l+1} = \underline{\underline{A}}_l \underline{\underline{S}}'_1 \underline{\mu}_0$$

$$y'_{l+1,j} = \sum_{z=1}^{n^r_{l+1}} a_{l,j,z} \sum_{i=1}^{n_l} \sum_{t=1}^{m_l} \mu_{l,i,t}(y_{l,i}) b'_{l,z,i,t} \qquad (13)$$

The difference between the outputs of the original and the reduced network from Eq (10), (13) and (12):

$$\left| y_{l+1,j} - y'_{l+1,j} \right| =$$

$$= \left| \sum_{i=1}^{n_l} \sum_{t=1}^{m_l} \mu_{l,i,t}(y_{l,i}) \left(b_{l,j,i,t} - \sum_{z=1}^{n^r_{l+1}} a_{l,j,z} b'_{l,z,i,t} \right) \right| \le \qquad (14)$$

$$\le \left| \sum_{i=1}^{n_l} \sum_{t=1}^{m_l} \mu_{l,i,t}(y_{l,i}) E_1 \right| = \left| \sum_{i=1}^{n_r} E_1 \right| = n_l E_1$$

Step 2.: Determination of the new membership functions

Let the following vectors and matrices be defined:

$$\underline{\underline{M}}_{l,j} = \begin{bmatrix} b'_{l,j,1,1} & \cdots & b'_{l,j,n^r_{l-1},1} \\ \vdots & \ddots & \vdots \\ b'_{l,j,1,m_l} & \cdots & b_{l,j,n^r_{l-1},m_l} \end{bmatrix},$$

$$\underline{\mu}_l(y_{l,j}) = \left[\mu_{l,j,1}(y_{l,1}) \cdots \mu_{l,j,m_l}(y_{l,j})\right]$$

where the size of $\underline{\underline{M}}_{l,j}$ is $m_{l,j} \times n^r_{l-1}$. With this notification:

$$\underline{y}'^T_{l-1} = \sum_{i=1}^{n_l} \underline{\mu}_l(y_{l,i}) \underline{\underline{M}}_{l,i} \underline{\underline{A}}^T_l \qquad (15)$$

Applying the above described singular value-based reduction to the $\underline{\underline{M}}_{l,j}$ matrices one will get:

$$\underline{\underline{M}}_{l,j} \approx \underline{\underline{T}}_{l,j} \underline{\underline{D}}_{l,j} \underline{\underline{V}}_{l,j} = \underline{\underline{T}}_{l,j} \underline{\underline{M}}'_{l,j}. \qquad (16)$$

The error of this reduction ($\underline{\underline{M}}_{l,i} - \underline{\underline{T}}_{l,j} \underline{\underline{M}}'_{l,j}$) will be the sum of the discarded singular values, $E_{2,j}$.

The output of the reduced neural network will be

$$\underline{y}''^T_{l+1} = \sum_{i=1}^{n_l} \underline{\mu}^r_l(y_{l,i}) \underline{\underline{M}}'_{l,i} \underline{\underline{A}}^T_l, \qquad (17)$$

where

$$\underline{\mu}^r_l(y_{l,j}) = \left[\mu^r_{l,j,1}(y_{l,1}) \cdots \mu^r_{l,j,m_l}(y_{l,j})\right] = \underline{\mu}_l(y_{l,j}) \underline{\underline{T}}_{l,j}$$

The error in the second step from Eq. (15) and (17):

$$\left|\underline{y}'^T_{l-1} - \underline{y}''^T_{l+1}\right| = \left|\sum_{i=1}^{n_l} \underline{\mu}_l(y_{l,i})(\underline{\underline{M}}_{l,i} - \underline{\underline{T}}_{l,i} \underline{\underline{M}}'_{l,i}) \underline{\underline{A}}^T_l\right| \le$$

$$\le \left|\sum_{i=1}^{n_l} \underline{\mu}_l(y_{l,i}) 1_{=(m_{l,i} \times n^r_{l-1})} E_{2,i} \underline{\underline{A}}^T_l\right| \le$$

$$\left|\sum_{i=1}^{n_l} \sum_{t=1}^{m_{l,i}} \sum_{z=1}^{n^r_{l-1}} \mu_{l,i,t}(y_{l,i}) a_{l,j,z} E_{2,i}\right| \le \left|\sum_{i=1}^{n_l} E_{2,i}\right| \qquad (18)$$

For the last step, the sum of the rows of $\underline{\underline{A}}_l$ must be less or equal than one. This could be ensured by a further transformation in the first step:

$$\underline{\underline{S}}_l \approx \underline{\underline{A}}'_l \underline{\underline{D}}'_l \underline{\underline{V}}'_l = (\underline{\underline{A}}'_l \underline{\underline{K}}_l) \underline{\underline{K}}^{-1}_l \underline{\underline{D}}'_l \underline{\underline{V}}'_l = \underline{\underline{A}}_l \underline{\underline{S}}'_l,$$

$$\underline{\underline{K}}_l = \left\langle 1 / \max_j \{\sum_z a_{l,j,z}\} \right\rangle$$

So the error in the reduction altogether is the sum of the error originating from the first and second step (from Eq. (14) and (18)):

$$E \le \sum_{i=1}^{n_l} E_{2,i} + n_l E_1. \qquad (19)$$

5. Conclusion

While neural networks are able to provide solutions in cases, when exact mathematical models not, and enough sample data are available, their high complexity and the lack of an universal method for the determination of the minimal needed size of the network limits their use in time-critical systems.

The SVD-based reduction offers a method to eliminate the redundancy of generalized neuro-fuzzy type networks. Furthermore, it makes possible further reduction, if a certain amount of error could be tolerated. This paper gives an error-bound for this further reduction, so the error of the reduction could be estimated from the discarded singular values. However, this error is valid only in case of the reduction of one layer of the neural network. If more layers are reduced, then the effect of the reduction of the layers must be summarized.

With this further reduction, it could be possible to construct several neural networks for the same task with different complexity and accuracy. This gives a way to use these generalized neural networks in anytime systems, as well, where an expert system can choose one of them, considering the temporarily available time and resources (Takács and Várkonyi-Kóczy, 2000a). By using the original SVD-based reduction method for fuzzy rule-bases, fuzzy inference systems could also be used in such architecture (Takács and Várkonyi-Kóczy, 2000b).

References

Baranyi P. and Y. Yang (1997). Singular value-based approximation with non-singleton support, *Seventh Int. IFSA World Congress*, Prague, June 25-29, 1997., pp.127-132.

Baranyi P., Y. Yam (1997). Singular Value-Based Approximation with Takagi-Sugeno Type Fuzzy Rule Base, *IEEE Int. Conf. on Fuzzy Systems*, 1997., pp. 265-270.

Baranyi P., L.T.Kóczy and T.D.Gedeon (1998). Improved Fuzzy and Neural Network Algorithms for word frequency Prediction in Document Filtering *Journal of Advanced Computational Intelligence* Vol. **2.** No. 3, 1998, pp 88-95.

Baranyi P., Y. Yam, C. T. Yang, A. R. Várkonyi-Kóczy (1999). Complexity Reduction of a Rational General Form, *IEEE Int. Fuzzy Systems Conf.*, Aug. 22-25, 1999, Seoul, Korea, pp.366-371.

Baranyi P., Y. Yam, H. Hashimoto, P. Korondi, P. Michelberger (2000a). Approximation and Complexity Reduction of the Generalized Neural Network, accepted to *IEEE Trans. on Fuzzy Systems*.

Baranyi P., K.F. Lei, Y. Yam (2000b). Complexity Reduction of Singleton Based Neuro-fuzzy Algorithm, accepted to *IEEE Conf on System, Man and Cybernetics 2000*

Baranyi P., I.Nagy, P.Korondi and H.Hashimoto (2000c). General Guiding Model for Mobile Robots and its Complexity Reduced Neuro-fuzzy Approximation *9th IEEE Int. Conf. on Fuzzy Systems* (FUZZ-IEEE 2000), San Antonio, Texas, 2000, pp. 1029-1032.

Hornik, K., M. Stinchcombe, H. White (1989). Multilayer Feedforward Networks are Universal Approximators, *Neural Networks 2*, pp. 359-366.

Rojas R. (1996). *Neural Networks, A Systematic Introduction*. Springer-Verlag, Berlin,

Takács, O., A. R. Várkonyi-Kóczy. (2000a). Anytime soft computing methods for intelligent diagnosis and control, *IFAC Symp. on Artificial Intelligence in Real Time Control*, 2-4 Oct., 2000., Budapest, Hungary

Takács, O., A. R. Várkonyi-Kóczy (2000b). Fuzzy tools in anytime systems, *IEEE Conf. on Intelligent Engineering Systems*, IEEE-INES 2000, 15-1.7 Sept., 2000, Portoroz, Slovenia

Yam Y. (1997). Fuzzy approximation via grid point sampling and singular value decomposition, *IEEE Trans. Syst., Man, Cybern.*, vol. **27**, pp. 933-951, Dec. 1997.

Yam Y., P. Baranyi, C. T. Yang (1999). Reduction of Fuzzy Rule Base Via Singular Value Decomposition, *IEEE Trans. on Fuzzy Systems*, Vol. 7., No. 2., Apr., 1999, pp.120-132.