A Novel Parameter Estimation Fuzzy Network for Target Approximation

Kourosh Neshatian

Iran Telecom Research Center
North Kargar St., Tehran, Iran
neshat@itrc.ac.ir

Mohammad Teshnehlab

K. N. Toosi University
Sharyatee St., Tehran, Iran
teshnehlab@eetd.kntu.ac.ir

Abstract. Here we introduce a novel Neuro-Fuzzy system for approximation and prediction of complex plants. The proposed system uses parameter estimation and scaling techniques; so called PENTA which stands for Parameter Estimation Network for Target Approximation. PENTA consists of two major components: a scaling component and a parameter estimation unit. The first component provides the gains required by the second component which is a set of primitive functions used for scaling. So the overall output of network is a synthetic function that is linear composition of these parameters and primitive functions. These two components have been expanded through five layers of a Neuro-Fuzzy network. Experimental results show that in comparison with current Neuro-Fuzzy systems like ANFIS, this novel system can have a better adaptation to complex plants. Also results show the unique capability of this system, in consecutive prediction of chaotic systems

Keywords: Fuzzy Systems, TSK Systems, ANFIS, Chaotic Time Series, Prediction, Error Back-Propagation

1 - Introduction

The complexity and the dynamics of some problems, such as prediction of chaotic systems and adapting to complex plant, require sophisticated methods and tools for building an intelligent system. Using fuzzy systems as approximators, identification and prediction methods, is a reliable approach for this purpose [13, 14]. Combination of fuzzy logic with architectural design of neural networks led to creation of Neuro-Fuzzy systems which benefit from feedforward calculation of output and back-propagation learning capability of neural networks, while keeping interpretability of a fuzzy system [9]. The TSK [18, 19] is a fuzzy system with crisp functions in consequent, which perceived proper for complex applications [1]. It has been proved that with convenient number of rules, a TSK system could approximate every plant [12]. TSK systems are widely used in the form of a neuro-fuzzy system called ANFIS (Adaptive Neuro-Fuzzy Inference System) [8]. Because of crisp consequent functions, ANFIS uses a simple form of scaling implicitly.

Input/Output scaling is a technique used to increase the flexibility of neural networks [16]. This scaling occurs by the mean of some scaling gains. Optimal input/output scaling gains in the fuzzy adaptive mechanism are computed off-line at the supervisor design phase. Generally scaling a vector means to add or subtract a constant and then multiply or divide by a constant, but here we extend this idea by providing a general form for scaling. This new form like before has some gains or parameters which must be determined. Against traditional methods in which the designer has to determine scaling coefficients, we propose a FIS(Fuzzy Inference System) for estimation of these parameters.

We construct this new architecture in the form of a nero-fuzzy network with five layers called PENTA (Parameter Estimation Network for Target Approximation). This network uses a FIS to estimate parameters required by scaling mechanism. We show how this estimation network and the scaling mechanism together could approximate and predict many complex systems. Numerical experiments also validate this novel architecture by showing that, in the same context, the PENTA acts more accurate than ANFIS.

2 - PENTA – A Novel Neuro-Fuzzy Architecture

As mentioned before, the primary theme in creating this new infrastructure is the architecture of Input-Output scaling systems [16]. A typical Input-Output scaling FIS has been illustrated in Figure (1). As shown in the figure, in an Input-Output scaling architecture, the input vector *x* is fed to a FIS or Neural Network as an input vector and again is incorporated in calculation of final output vector by the mean of a * operator (usually a multiplication).


Figure (1). General structure of an Input-Output scaling system.

In PENTA we extend this idea by assuming a general and flexible function for last layer of the system which is a linear combination of a set of *primitive functions* in the form of Equation (1).

$$Final\ Output = a_1(\mathbf{x})p_1(\mathbf{x}) + \ldots + a_{NP}(\mathbf{x})p_{NP}(\mathbf{x}) \tag{1}$$

in which *x* is the input vector, *NP* is the number of primitive functions, a_1 to a_{NP} are linear coefficients obtained form previous layer and p_1 to p_{NP} are primitive functions which determined by designer. We refer to this equation as *Scaling Cell* in PENTA architecture. The General block diagram of PENTA has been shown in figure (2).
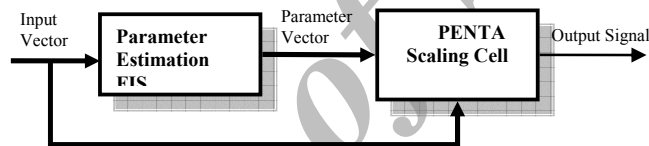

Figure (2). General block diagram of PENTA system. Bold arrows show vector transmission lines. Left block performs parameter estimation while right block uses these parameters for scaling.

In this architecture, the FIS block (left) estimates parameters according to the input vector and submits them to the scaling cell in which primitive functions are evaluated according to the input vector again and a linear combination is generated as final one dimensional output of the system. It's obvious this architecture satisfies the requirements of an input/output scaling system, in a more general form. This generality lets designer to select different strategies for scaling cell, estimation block. It's also obvious that in the case of a single primitive function which is an identity function, the PENTA plays the role of a traditional Input-Output scaling system. All places across this paper we study a MISO (Multiple Input, Single Output) infrastructure of PENTA. In MIMO (Multiple Input, Multiple Output) applications we can simply use a combination of multiple PENTA architecture simultaneously. In the next section we will consider this novel architecture as five layer neuro-fuzzy system (as its name inspires) an describe details of each layer.

3 - Functionality of PENTA Layers

Now we expand the previously mentioned block across five layers of a complete neuro-fuzzy system. This new composition has been shown in figure (3). The first four layers play the role of a FIS with singleton fuzzification, product implication and aggregation and center-average defuzzification. The single cell of fifth layer is performs required scaling mechanism as discussed before. The input vector is **x** which has *n* elements. So our inputs space is *n*-dimensional. The output of PENTA is a scalar value. As a common convention, we use bold letters for vectors and capital bold letters for matrices in our notation.
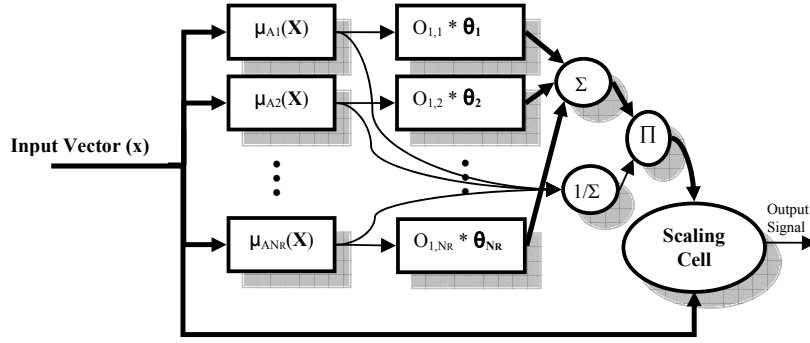
Figure (3). Five layers of PENTA. The first four layers constitute a FIS which estimates the parameters required by fifth layer. The fifth layer is a scaling operator which scales the input vector with parameters received from fourth layer by a linear combination of some primitive functions.

The singleton fuzzification in PENTA causes that the first layer simply contains the antecedent membership functions. These functions take a point in a *n*-dimensional hyperspace and return the membership grade of that point to the corresponding fuzzy set. Accordingly the number of cells in this layer determines the number of rules of FIS. We note this number with *NR*. FCM (Fuzzy C-Means) clustering [9,15] method is used to partition the input space of PENTA, so antecedent membership functions are like those used in FCM clustering as below:

$$O_{1,i} = \mu_{Ai}(\mathbf{x}) = \frac{1}{\|\mathbf{c}_i - \mathbf{x}\|_2^{Expo}}$$

$$where \qquad i = 1,2,..., \ N_R$$

$$and \qquad \|\mathbf{c}_i - \mathbf{x}\|_2 = \sqrt{(\mathbf{c}_{i,1} - \mathbf{x}_1)^2 + ... + (\mathbf{c}_{i,n} - \mathbf{x}_n)^2}$$

(2)

where $\mathbf{c_i}$ is the center of *i*-th region (partition) and *Expo* determines the steepness of membership function which usually is 2. In fact the above expression evaluates the inverse of Euclidean distance between point *x* and center *c_i* as membership grade of *x* to region *i*. Output of first layer cells are scalar and determine the membership grade of input vector *x* to the corresponding antecedent fuzzy set.

Because FIS part of PENTA uses product engine and center-average defuzzification, the second layer simply multiplies the outputs of first layer by the center of corresponding consequent fuzzy set. This is a scalar by vector multiplication so that output of first layer cells are scalar membership grade and center of consequent fuzzy sets are points in FIS output hyperspace. The number of dimensions of FIS output hyperspace is equal to the number of parameters which scaling cell requires which is *NP*. Equation (3) shows the second layer activity. In this equation θ_i is the center of *i*-th consequent fuzzy set and *NR* shows the number of FIS rules.

$$\mathbf{o}_{2,i} = o_{1,i} * \mathbf{\theta}_i \quad (i = 1,2,... , NR)$$

(3)

These two layers play the role of a center-average defuzzification unit in the FIS. As shown in figure (3), number of cells is constant in these two layers. The output of fourth layer is a vector consisting of *NP* elements which are the parameters required by scaling cell. These parameters can be calculated as follow:

$$a(\mathbf{x}) = \frac{\sum_{i=1}^{N_R} \mu_{Ai}(\mathbf{x}) * \mathbf{\theta}_i}{\sum_{i=1}^{N_R} \mu_{Ai}(\mathbf{x})}$$

(4)

Where θ_i is the center of *i*-th consequent fuzzy set and μ_{Ai} is the *i*-th antecedent membership function and *a(x)* is the parameter vector estimated by the FIS and will be fed to scaling cell. Third and fourth layer are formulated as follow:

$$\mathbf{o}_{3,1} = \sum_{i=1}^{N_R} \mathbf{o}_{2,i}$$

$$o_{3,2} = (\sum_{i=1}^{N_R} o_{1,i})^{-1}$$

(5)

$$a(\mathbf{x}) = \mathbf{o}_{4,1} = \mathbf{o}_{3,1} * o_{3,2}$$

The four layers described till now were components of FIS block. The result of this block which is an estimated parameter vector will be fed to single cell of fifth layer which is scaling cell. Scaling cell evaluate the final scalar output of the system by equation (6). This could be expressed with respect to output of previous layers as follow:

$$o_{5,1} = \sum_{i=1}^{N_P} (\mathbf{o}_{4,1})_i \, \mathbf{p}(\mathbf{x})_i = \mathbf{o}_{4,1}^T * \mathbf{p}(\mathbf{x})$$

(6)

Like many other FISs and NNs it will be very useful and compendious to study the total behavior of PENTA as a function of the input vector. We can rewrite PENTA with respect to *x* as follow which includes all the functionalities provided by all five layers:

$$PENTA(\mathbf{x}) = \sum_{j=1}^{N_P} \left(\frac{\sum_{i=1}^{N_R} \mu_{Ai}(\mathbf{x}) * \mathbf{\Theta}_{i,j}}{\sum_{i=1}^{N_R} \mu_{Ai}(\mathbf{x})} \right) * p_j(\mathbf{x})$$

(7)

Where $\mathbf{\Theta}$ is a matrix yielded by ordering transposed θ vectors vertically as illustrated in equation (8). $\mathbf{\Theta}$ is a *NR×NP* matrix as below:

$$\mathbf{\Theta} = [\theta_1, \theta_2, \ldots, \theta_{NR}]^T$$

(8)

Equation (7) expresses the response of PENTA system to a single input vector **x** which is a point in input hyperspace. Sometimes during implementation of neuro-fuzzy systems it's preferred to have a response function for matrix inputs. For example when studying offline learning methods for a particular system, we are interested to know the response of the system for a batch of input vector (in matrix form) with a brief expression. Bellow we address this matrix form:

$$\mathbf{PENTA(X)} = \sum_{i=1}^{NP} [\mathbf{M(X)} * \mathbf{\Theta}] \otimes \mathbf{P(X)}_{*,i}$$

(9)

Where **X** is matrix yielded by stacking up *s* input vectors. So **X** is a *s×n* matrix. **M(X)** is a matrix of membership grades in the way that $\mathbf{M(X)_{i,j}}$ is a scalar value indicating the membership grade of *i*-th input vector to the *j*-th cluster (or antecedent membership function). The output of above function is a vector with *s* elements which contains the output (response) of PENTA for each *s* input vector. It's important to know that **M(X)** is normalized which means:

$$\sum_{i=1}^{N_R} \mathbf{M(X)}_{*,i} = 1$$

(10)

So there is no need for a division like fraction used in Equation (7) and defuzzification is performed implicitly. In above equation, sigma sign and '*,I' indices indicates summation along rows.

4 - Learning Strategies

As a convention in drawing block diagram of neuro-fuzzy systems, parametric cells have been drawn within a rectangle in Figure (3). So as shown in that figure the cells of first and second layer contain some parameters which could be adjusted. Though the primitive functions in scaling cell may themselves contain parameters, the only layers which we focus on them for parameter tuning will be first and second layer. Again we recall that these parameters are cell-parameters of layers and are different from the estimated scaling parameters which are output of fourth layer but adjusting these cell-parameters would lead to a better estimation of scaling parameters.

Parameters of first layer are about antecedent membership functions and define the center and wideness of them. Because we don't use grid meshing and instead use cluster method to partition the input space, the

adjustment of first layer parameters after clustering is not a critical problem and has no a major impact on the precision of the system [6] so we ignore this layer during training phase. The second layer parameters are center of consequent fuzzy sets which are stored in Θ matrix. So the learning process is responsible for finding optimum values of Θ elements.

As mentioned before, PENTA uses clustering to partition its input space. Against traditional meshing methods, clustering prevents creation of some unused partitions where the probability of occurrence of an input vector, is very low. Because we are going to compare the capabilities of PENTA with a well known neuro-fuzzy system called ANFIS which the number of its rules must be qualified by designer, we use a clustering method which the number of clusters could be specified before, like FCM [3].

4.1 Gradient-based Learning

Although in some situations, the output of the system may be linear related to some parameters and so regression techniques [4,7] could be used for learning, for saving the generality of our learning strategy, we consider the gradient-based optimization. Gradient-based optimization is a big family in neuro-fuzzy learning methods [2 pages: 199-206]. Like other kinds of optimization methods we have to find a proper Objective function (also Energy, Error or Cost function) which could be minimized during learning phase. Here we don't focus on details of minimization method because designer is free to select any method among many alternatives, available. In experiments, the Unconstraint Nonlinear Optimization method has been used for tuning parameters.

The key step for applying an iterative gradient-based approach is determination of objective function. We are interested about offline functionality of PENTA and intended to test its approximation and prediction capabilities so we have to find an objective for a set (batch) of data. Using same notation as above, X stands for a set of s training input vectors and $\mathbf{f(X)}$ for a set of s target (desired) value. The offline objective function will be:

$$E = (\mathbf{f(X)} - \mathbf{PENTA(X)})^T (\mathbf{f(X)} - \mathbf{PENTA(X)}) \tag{11}$$

Where PENTA($\mathbf{X}$) show the real output of the system. Usage of transpose operator and a matrix multiplication is equivalent of dot product. The result of this multiplication is a scalar value which is sum of square error. The optimization goal is to minimize the function E. so we have to provide the gradient of this function for a gradient-based optimization. Regarding the variable of the function E is Θ:

$$\frac{\partial E}{\partial \Theta} = \left[-(\mathbf{f(X)} - \mathbf{PENTA}(\mathbf{X})) \otimes \mathbf{M(X)} \right]^T \mathbf{P(X)} \tag{12}$$

The $\otimes$ symbol is an array multiplication (element by element). Expression in the bracket results a $NR \times s$ matrix after transposition. After multiplying this matrix by $P(\mathbf{X})$ we obtain a $NR \times NP$ matrix which is in the same dimension of Θ

5 - Experimental Results

Two experiments have been done to study different aspects of PENTA. Both experiments have been also performed for ANFIS with exactly same context and conditions. ANFIS is a well-known neuro-fuzzy system which its resources and programs is publicly available. So we could evaluate both PENTA and ANFIS in the same manner. In the first experiment the flexibility of PENTA for having different kinds of functions is tested. The experiment shows how this flexibility would lead to a better capture of target system, in identification and approximation applications. In the second experiment we study how scaling capability of PENTA results better fitness in prediction of a chaotic systems like Mackey Glass time series.

5.1 Grabbing Sinc Function

The Sinc Function is a two variable function which makes a surface like figure 4.(a). The Sinc is defined as bellow:

$$\text{sinc}(x, y) = \frac{\sin(x)}{x} + \frac{\sin(y)}{y} \tag{13}$$

For generating training data set and the figure 4.(a), *40* points have been generated with equal distance from each other (linear spacing) in range *[-10, 10]* in each input dimension. So there are *1600* data pairs.

Since the Sinc is not defined in $x= 0$ or $y= 0$, these pairs have been excluded. In both PENTA and ANFIS, we use nine rules. ANFIS uses meshing method and TSK model according to its nature. PENTA incorporates FCM Clustering to partition the input space and we use a scaling operator like bellow in fifth layer, with two nonlinear primitive functions in the first and second clauses.

$$PENTA(x, y) = a_1(x, y)x^2 + a_2(x, y)y^2 + a_3(x, y)x + a_4(x, y)y + a_5(x, y) \qquad (14)$$

Where parameters a_1 to a_5 must be estimated by first four layers. Figure 4.(b) shows how this equation approximates a portion of Sinc function inside a cluster (partition). Because we used some nonlinear primitive functions the surface is curved. PENTA approximates Sinc function as Figure 4.(c) with normalized relative error of *0.0037* and ANFIS performs this as illustrated in Figure 4.(d) with normalized relative error of *0.0429*. Evidently, Increase in number of rules, results the decrease in error value in both systems.

(a)

(b)

(c)

(d)
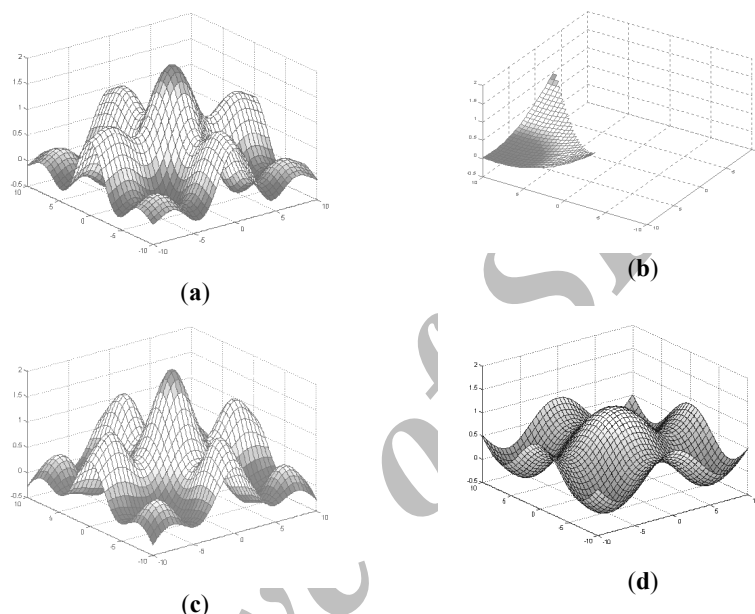
Figure 4. (**a**) The Sinc function in the domain $x,y \in [-10,10]$. (**b**) Equivalent surface for a rule of PENTA (spanning a cluster). (**c**) PENTA approximation of Sinc with five rules. (**d**) ANFIS approximation of Sinc with five rules.

5.2 Prediction of Mackey Glass Time Series

The Mackey Glass time series is a chaotic time series which has recently been a benchmark for testing the power of prediction systems [17]. Chaotical behavior of this series means that the values of the series are erratic and unpredictable during the time. This series can not be expressed as function of time. The Mackey Glass time series is define by following differential equation:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1 + x(t-\tau)^{10}} - 0.1x(t) \qquad (15)$$

Where τ is constant. Values greater than *17* for this constant makes the series, chaotic. The bigness of this constant determines the complexity of time series. This equation can not be solved analytically (symbolic) and so we generate time series numerically. Respect to the fact that $x(t+1) \approx x(t) + \frac{dx(t)}{dt}$ and with initial value $x(t_0)=1.2$ and $\tau = 30$, the time series have generated for $t=0$ to $t=599$. This has been shown in Figure 5.(a). a data set from $t=0$ to *349* is allocated for training purpose. *15%* percent of these training data selected for checking (overfit control). So about *300* data remains for training. After training of PENTA and ANFIS they will be tested by data from $t=350$ to *599*.

Both PENTA and ANFIS are not recurrent networks. They have no memory and so are stateless. This is an obstacle in prediction. To satisfy this requirement we increase the number of system inputs and feed previous inputs of the system as extra inputs to the system. So an input vector in the time *t-1* will be like

$\mathbf{x} = \{ x(t-1), x(t-2), x(t-3), x(t-4) \}$. This causes the system to act as a state-based system which is sensitive to the previous inputs (and so states). The number of this recall and the distance between them affects the precision of prediction. Table 1 show error values in training and testing phase of ANFIS and PENTA. Less error in testing phase of PENTA is because of the smallness of testing set relative to the training set and good approximation o PENTA.

Table (1). Absolute (sum of square) errors in PENTA and ANFS in prediction of Mackey Glass time series.

	Training Error	Checking Error	Testing Error
ANFIS	0.02595825	0.00301605	0.0370125
PENTA	0.01724000	0.00108330	0.0008401

Finally an experiment has been done to reveal the unique feature of PENTA. All the experiments performed to predict a chaotic time series since now use a step by step approach and so the predict one step forward [11]. In this manner, when the system is going to predict the value of *x(t)* all the previous values fed to the system are real values. So having values of series since *t* we can't predict the value of series in *t+2*, because the system requires the value of series in *t+1*. In *Consecutive Prediction* given the values of series since *t*, the system could predict the series till any desired time. This is because, in consecutive prediction, the output of the system is fed to itself as an input. Naturally in this method, the error is accumulated step by step, because we incorporated an approximated value as an input again consecutively.

Figure 5(b) illustrates a result of such consecutive prediction by PENTA. For this experiment a training data set containing data from *t=100* to *700* and *t=951* to *1550* has been used and then the system predicts the value from *701* to *950*. The absolute error for this prediction is *8.53* which is very greater than previous errors an as mentioned before this because of cumulative nature of error in this experiments. The figure shows, as the time elapses, the prediction goes far away actual value of the series. This experiment could not be preformed by ANFIS, because after 4 to 6 time step, the system awfully diverges and the error goes to the infinity.

(a) **(b)**
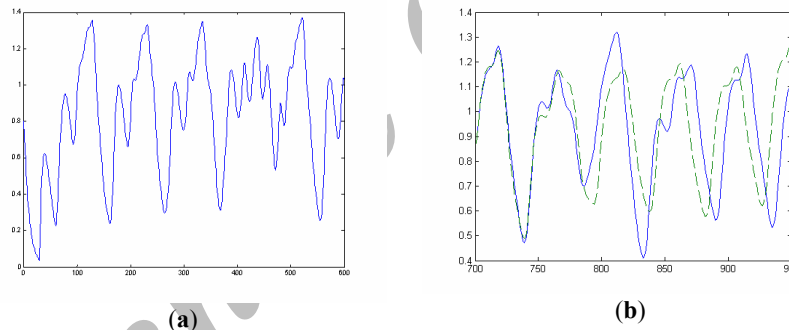
Figure (5). Prediction of Mackey Glass time series. (**a**) Mackey Glass time series from *t=0* to *599* with *x(t₀)=0.8* and *τ=30*. (**b**) Consecutive prediction of Mackey Glass time series by PENTA from *t=700* to *949* (dotted curve) in contrast with actual series value (continuous curve).

5.1 Improvement Evaluation

Although the analytical studies show that PENTA architecture is grate solution in input-output scaling systems, here we take a comparative approach to validate our design. Table 2 shows performance of ANFIS and PENTA in different applications and the improvement of PENTA relative to the ANFIS. Averaging through the third row of table, results a *96.6%* of overall improvement in PENTA.

Table 2. ANFIS and PENTA errors have shown in different applications. Third row shows PENTA improvement in comparison with ANFIS.

	Approximation (Sinc function)	Prediction (Mc. Glass series)	Consecutive Prediction
ANFIS Error	0.0429	0.0370125	∞
PENTA Error	0.0037	0.0008401	8.53
Improvement	91.3%	97.7%	100%

6 - Concluding Remarks and Future Works

In this paper we proposed a novel architecture for input-output scaling systems called PENTA. The PENTA is a neuro-fuzzy system consisting of two major components: Parameter Estimator and Scaling Cell. PENTA uses a FIS for parameter estimation and a general flexible function for scaling. So PENTA brings a more precise approach in identification and prediction. Results show that PENTA improves the capability of ANFIS by *96.6* percent.

In our design, the selection of primitive functions is manual task. It means that the designer have to chose proper functions in scaling cell. This may seem a drawback but could be faced as feature selection problem. A suitable feature selection method can be used for: 1. selecting proper inputs for the system and 2. Determining what function of input is better for scaling cell. For example, in Sinc approximation the square of input has great effect in results (because of curved surface of Sinc). A few works has been done on feature selection in ANFIS, but they usually use a try and error approach [10].

Here, because we intended to compare our design with ANFIS, we had to use a clustering method in which the number of clusters could be defined before. So we used the FCM (Fuzzy C-Means) Clustering. In real application, it will be more appropriate that this number could be determined by the system, itself. So we suggest a different cluster method in real applications. For instance in *Subtractive Clustering* [15] the number of clusters (partitions) will be calculated by the system.

References

[1]. Alcala R., Casillas J., Cordon O. and Herrera F. Learning TSK fuzzy rule-based system from approzimate ones by mean of MOGUL Methodlogy. Granada university of spain, October 2000.

[2]. Burden R.L. and Faires J.D. Numerical analysis. PWS-Kent Pub. Co., Boston, 5th edition, 1993.

[3]. Chiu S.L. Fuzzy model identification based on cluster estimation. Journal of intelligent and fuzzy systems, 2(3), 1994.

[4]. Draper N.R. and Smith H. Applied regression analysis. John Wiley & Sons, New York, 2nd edition, 1981.

[5]. Feuring T. Learning in Fuzzy neural Networks. Wesfalische Wilhelms-University, 1996.

[6]. Hoppner F. and Klawonn F. A new approach to fuzzy partitioning. Emden University of Applied Sciences, 2001.

[7]. Hsia T. C.. System identification: least square methods. D. C. Heath and Company, 1997.

[8]. Jang J.R. ANFIS: Adaptive-Network-Based Fuzzy Inference System. IEEE Trans. On Systems, MAN and Cybernetics. Vol 23, No. 3: 665-685, May/June 1993.

[9]. Jang J.R. Sun C. and Mizutani E. Neru-Fuzzy and Soft Computing. Prentice-Hall, 1997.

[10]. Roger Jang J. S. Input selection for ANFIS Learning. National Tsing Hua University of Taiwan, 1998.

[11]. Jones R. D., Y. C. Lee, C. W. Barnes, G. W. Flake, K. Lee and P. S. Lewis. Function approximation and time series prediction with neural networks. In proceding of IEEE internation joint conference on neural networks, pages 649-665 (Vlolume I), 1990.

[12]. Mannle M. FTSM : Fast Takagi-Sugeno fuzzy modeling. University of Karsruhe (informatik.uni-karsruhe.de), 2000.

[13]. Mannle M. Identifying rule-based TSK fuzzy models. University of Karsruhe (informatik.uni-karsruhe.de), 1999.

[14]. Mannle M., Richard A. and Dorsam T. A rule-based fuzy model for nonlinear system identification. University of Karsruhe (informatik.uni-karsruhe.de), 1996.

[15]. Mathworks Inc. Fuzzy Logic Toolbox User's Guide. Mathworks, 2001.

[16]. Mckinney T.M. and Kehtarnavas N. Fuzzy rule genereation via multi-scale clustering. IEEE Transactions on System, Man, Cybernetics. 1997.

[17]. Sincak P., Holecy M.and Ducai M. Computational Intelligent in financial cybernetics. Tehcnical university of Kosice. 1997.

[18]. Sugeno M. and Kang G.T. Structure identification of fuzzy model. Fuzzy Sets and Systems, 28:15-33, 1998.

[19]. Takagi T. and Sugeno M. Fuzzy identification of systems and its application to modeling and control. IEEE Transaction on Systems, Man and Cybernetics, 15:116-132, 1985.