Computer Assisted Mechanics and Engineering Sciences, 14: 651–658, 2007.

Multivariate data approximation with preprocessing of data

Witold Kosiński[1,2], Dorota Kowalczyk[2], Martyna Weigl[3]

[1] *Polish-Japanese Institute of Information Technology, Chair of Intelligent Systems*
ul. Koszykowa 86, 02-008 Warszawa, Poland

[2] *Kazimierz Wielki University in Bydgoszcz,*
Department of Mathematics, Physics and Technology
ul. Chodkiewicza 30, 85-064 Bydgoszcz

[3] *Polska Telefonia Cyfrowa, ERA GSM, 02-222 Warszawa*

(Received in the final form September 18, 2007)

An adaptive information system is constructed in order to approximate a set of multidimensional data. To get better approximation properties a pre-processing stage of data is proposed in which the set of points, forming the multidimensional data base and called a training set TRE, undergoes a clustering analysis. In the analysis two independent clustering algorithms are used; on each cluster a feed-forward neural network is trained and a membership function of a fuzzy set is constructed. The constructed system contains a module of two-conditional fuzzy rules consequent parts of which are of the functional type. Each rule is designed on a pair of clusters.

1. Introduction

Construction of universal approximators for multivariate functions (i.e. real-valued functions of many variables) is often based either on artificial neural networks or on fuzzy inference systems. Some of them are grounded with theoretical results concerning the existence problem, others are equipped with adaptation schemes. In the previous century, general existence results have been obtained within fuzzy systems (cf. [2, 8]) or within neural networks (cf. [3, 4, 7, 22, 25]).

The present authors have approached that problems using the both tools: one based on an adaptive fuzzy inference system, the other — on feed-forward neural networks [17, 27–30]. It has been also shown there that a non-homogeneous M-Delta neural networks as well as an adaptive fuzzy inference system equipped with a generalised Takagi–Sugeno–Kang fuzzy rules are universal approximators [17, 29].

While constructing and working with neural networks and fuzzy inference systems our experience shows that their real approximation capabilities can be improved by an appropriate pre-processing of sample data introduced for such systems. In fact our observations made for an adaptive fuzzy inference system show that they are very sensitive to the definition of the membership functions of the fuzzy sets and their support used in the covering of input domain. It is difficult to adjust the overall output of the system to the sample data by adapting the generalised weight vector and parameters of the membership functions if the data are of a particular type. For example, if on a part of the domain the function to be approximated is constant, while on another part the function has very high gradients.

In the present paper we are changing the order of construction. Knowing a particular multidimensional data base TRE which can be regarded as points from a graph of a multivariate real-valued (continuous) function (that is known in a discrete number of points), we start our construction procedure with a process of data mining. In that initial process the knowledge about the type and the behaviour of the function to be approximated is extracted. This knowledge will be used in the actual

designing process. The process of knowledge extraction from numerical multidimensional data is just a kind of clustering analysis during which partitioning of the data base into two unions of subsets is performed. Since each entry to the data base can be treated as an order pair in which the first component represents a value of a multidimensional input vector, while the second is a real-valued output to that input value, the set TRE can be regarded as a subset of the Cartesian product of multidimensional input space, say $\mathcal{X}$ and 1D output one, say $\mathcal{Y}$. Hence together with that partitioning of the data base one can perform the projection on the space $\mathcal{X}$ to get a splitting of the input domain into sub-domains or clusters.

After the first stage of the construction procedure the next one appears in which on each cluster a single mapping neural network SIMNN is designed and trained. Moreover, with each cluster a fuzzy set is attached with the corresponding membership function in the form of a generalised Gaussian function. The function possesses in its definition two characteristic features of the cluster: its covariance matrix $\boldsymbol{S}$ and its centroid $\boldsymbol{a}$. Then a module of two-conditional rules *if-then* for a fuzzy inference system is constructed, consequent parts of which are weighted sum of outputs of artificial feed-forward neural networks already constructed. In each premise part of the rules a pair of fuzzy sets attached to the pair of clusters from both unions (cluster coverings) appear. The number of rules is equal to the product of the numbers of members (clusters) of the both unions.

In the author's opinion proposed type of procedure can reduce greatly the discrepancy (or noise) contained in the numerical data. Then the overall output of the approximating system is defined as a convex combination of all outputs given by consequent parts of all rules, where the coefficient of the combination aggregates values of their membership functions. The aggregation can be made with the use of some t-norm. However, if the fuzzy sets and their membership functions involved will be regarded in most general sense, namely ordered fuzzy sets proposed by the first author in [14], as some generalisation of ordered fuzzy numbers [11, 15], then all algebraic operations are for our disposal, and t-norm are no more necessary. For example in his Ph.D. Thesis [24] Prokopowicz has invented different aggregation procedures which can be adapted for the case of ordered fuzzy sets, as well.

At the end an adaptation (or so-called learning) process is performed on the whole set TRE during which some free parameters of the membership functions involved in the fuzzy sets of the fuzzy rules can be tuned.

Notice that in this paper we make the next step beyond our previous approach [13, 18, 21] with generalised Takagi–Sugeno–Kang fuzzy rules [26], since each consequent part of the rule is a weighted sum of outputs of two single mapping neural network SIMNN constructed for data from the corresponding clusters. Each network is trained on the cluster as on its input domain.

The constructed information system has grounded rather well the term a *fuzzy-neural system.* In the author's opinion proposed type of procedure is essential for the efficiency and accuracy of the constructed system: an adaptive fuzzy-neural inference system. Moreover, it can reduce greatly the noise contained in the numerical data.

For knowledge extraction from data one can use different methods. We can suggest two of them proposed recently in our previous publications. One is based on a seed-growing methods [9, 16], another uses an evolutionary programming, that makes use of a problem-oriented genetic algorithms [19].

The organisation of the paper is as follows. In Section 2 we make assumptions about the covering of data space. Single mapping neural networks with different activation functions defined for their neurons and trained on individual clusters are described in Section 3. The adaptive fuzzy inference system equipped with two-conditional fuzzy *if-then* rules is shortly sketched in Section 4.

2. Knowledge extraction from data

In the standard approach when dealing with an approximation problem the knowledge about the function to be approximated is contained in a set of the so-called training pairs

$$\mathsf{TRE} = \{\boldsymbol{p}^q = (\boldsymbol{x}^q, y^q) \in \mathbb{R}^{n+1} : \ q = 1, 2, \ldots, P\} \tag{1}$$

that represents a discrete number of points from the graph of unknown (i.e. to be looked for) function relationship between values of $\boldsymbol{x}$ and desired y since in each pair $\boldsymbol{p}^q = (\boldsymbol{x}^q, y^q) \in \mathsf{TRE}$ the value y^q is the so-called desired value corresponding to data represented by given input vector $\boldsymbol{x}^q$.

We describe two methods of extracting knowledge from the set TRE: a seed growing approach for clustering problem of large numerical multidimensional data set [9] and the evolutionary approach [10, 19] for inference systems.

Clustering via seed growing algorithm

The approach used in [9] is based on image segmentation known in the medical, digital picture segmentation to extract the significant information from images and to improve the interpretation process realised by the end-user, e.g. a physician. Notice that in digital processing of large numerical multidimensional data the clustering analysis corresponds some how to the image segmentation in the picture processing . However, in medical images the number of possible classes is given by the physician explicitly together with the seed pixel for each class. Here one should choose in some sense automatically by the algorithm. Moreover, in the set TRE i.e. in the subset of a graph of a scalar-valued multivariate function, there is a lack of a natural neighbouring topology of the images that is based on the concept of the 4- or 8-connectedness.

The seed growing algorithm is composed of two parts:
Rough clustering according to y in which a fuzzy histogram of the variability of y is used, and is divided into the following steps:

1. create a histogram of the variability of y;
2. fuzzify it by calculating its convolution with a Gauss-like function;
3. use the minima of the fuzzy histogram as boundaries of subintervals in y;
4. divide all the elements (pairs $(\boldsymbol{x}, y)$) of TRE into clusters; pairs with y's belonging to the same subinterval form the same cluster.

Exact clustering according to $\boldsymbol{x}$ and y in which the seed growing approach is developed after the rough clustering [9].

At the end of this clustering analysis we get a covering $\{\mathcal{K}_{1h}, h = 1, 2, \ldots, M_1\}$, i.e. a union of clusters that covers the whole set TRE.

Clustering via evolutionary algorithm

In [19] an evolutionary algorithm for extracting the knowledge from the database by splitting it into clusters was proposed and implemented in [10]. Here we sketch only its main features.

We are distinguishing two types of evolutions: external, at the level of clusters, and internal, at the level of training pairs. We introduce three types of *selection operators* for the population of clusters: roulette, tournament selection, proximity selection, as a combination of two others, and four types of genetic operators (cf. [19]) for the population of clusters: unification operator that acts on a pair of clusters and produces a new cluster as a union of both parents, crossover operator that exchanges parts of two clusters, separation operator that produces two other clusters by splitting a cluster into two, and global mutation operator that acts on an individual covering, regarded as a family of clusters producing a new covering.

To evaluate each off-spring cluster in the population we are using different local fitness functions: it could be the maximal distance of the elements of the cluster from the cluster centroid, or a mean variation of all elements in the cluster. Basing on the local fitness function of cluster one can build a global fitness function for the whole covering as a sum of local fitness function of its clusters.

In the first stage m independent evolutionary processes of m coverings by creating m initial coverings are performed. It is done with the help of the histogram as in the previous approach. During the evolutionary process for each generation one of the genetic operators that acts on

clusters (or pairs of clusters) is applied. Then one of the selection operators to the whole population is used. In this way new covering is constructed that forms the population for the next generation. After a fixed number of generations a global fitness (evaluation) function to each covering is applied and then a population of all coverings is formed.

The present authors belief that in real time applications constructing approximating systems the use of two different tools: one based on an adaptive fuzzy inference system, the other — on feed-forward neural networks, may be more promising.

2.1. Characterisation of clusters

At the end two coverings of TRE by clusters are ready, i.e. two families of clusters $\{\mathcal{K}_{\alpha 1}, \mathcal{K}_{\alpha 2}, \ldots, \mathcal{K}_{\alpha M_\alpha}\}$, $\alpha = 1, 2$, such that

$$\mathsf{TRE} = \bigcup_{h=1}^{M_1} \mathcal{K}_{1h} \qquad \text{and} \qquad \mathsf{TRE} = \bigcup_{k=1}^{M_2} \mathcal{K}_{2k}\,. \tag{2}$$

For each cluster $\mathcal{K}_{1h} \subset \mathbf{R}^{n+1}$ and $\mathcal{K}_{2k} \subset \mathbf{R}^{n+1}$ its centroid $\boldsymbol{p}^{1h} = (\boldsymbol{a}^{1h}, d^{1h}) \in \mathcal{X}$ and $\boldsymbol{p}^{2k} = (\boldsymbol{a}^{2k}, d^{2k}) \in \mathcal{X}$, respectively, is defined in similar way to (3)

$$\boldsymbol{p}^{1h} = \frac{1}{N_{1h}} \sum_{j=1}^{N_{1h}} \boldsymbol{p}_j^{1h}\,, \qquad \boldsymbol{p}^{2k} = \frac{1}{N_{2k}} \sum_{i=1}^{N_{2k}} \boldsymbol{p}_j^{2k}\,. \tag{3}$$

To each cluster $\mathcal{K}_{1h}$ or $\mathcal{K}_{2k}$ we relate its scatter (variance-covariance) matrix $\boldsymbol{W}^{1h}$ and $\boldsymbol{W}^{2k}$, respectively, calculated according to the formula [1]

$$\boldsymbol{W}^{1h} = \frac{1}{N_{1h}} \sum_{j=1}^{N_{1h}} (\boldsymbol{p}_j^{1h} - \boldsymbol{p}^{1h}) \otimes (\boldsymbol{p}_j^{1h} - \boldsymbol{p}^{1h}), \qquad \boldsymbol{W}^{2k} = \frac{1}{N_{2k}} \sum_{i=1}^{N_{2k}} (\boldsymbol{p}_i^{2k} - \boldsymbol{p}^{2k}) \otimes (\boldsymbol{p}_i^{2k} - \boldsymbol{p}^{2k}), \tag{4}$$

where $\otimes$ denotes the tensor product of two vectors. The scatter matrix can be used to measure the efficiency of the clustering analysis in the definition of the fitness function [19].

Now the projection of each $\mathcal{K}_{ih}, \mathcal{K}_{2k} \subset \mathbf{R}^{n+1}$ on the input space $\mathcal{X} \subset \mathbf{R}^n$ forms two families $\{X_1, X_2, \ldots, X_{M_\alpha}\}$, $\alpha = 1, 2$, of sub-domains (input clusters) that forms two coverings of the input data $\boldsymbol{x}'s$ from $\mathcal{X}$.

In a similar way to Eq. (4), we can define two corresponding scatter matrices $\boldsymbol{S}_{1h}$ and $\boldsymbol{S}_{2k}$ of each input cluster X_{1h} and X_{2k},

$$\boldsymbol{S}_{1h} = \frac{1}{N_{1h}} \sum_{j=1}^{N_{1h}} (\boldsymbol{x}_j^{1h} - \boldsymbol{a}^{1h}) \otimes (\boldsymbol{x}_j^{1h} - \boldsymbol{a}^{1h}), \tag{5}$$

and in a similar way $\boldsymbol{S}_{1k}$, where $h = 1, 2, \ldots, M_1$, $k = 1, 2, \ldots, M_2$. Let us notice that the matrices $\boldsymbol{S}_{1h}$, $\boldsymbol{S}_{2k}$, as well $\boldsymbol{W}^h, \boldsymbol{W}^k$ are symmetric and positive semi-definite. The eigenvectors corresponding to the vanishing eigenvalues determine the directions of the vanishing "thickness" of the cluster. These observations can be used in reducing the data.

We assume here for simplicity that matrices $\boldsymbol{S}_h$, $\boldsymbol{S}_k$, $h = 1, 2, \ldots, M_1$, $k = 1, 2, \ldots, M_2$, is nonsingular.

One can go beyond the present case of the extraction of knowledge from the data base. Such a case has been described by the author of thesis [21]. In [21] Kowalczyk assumed that on the input domain two families of fuzzy rules have been constructed: the first family contained multi-conditional (generalised) fuzzy rules of Takagi–Sugeno–Kang and the other — one-conditional rules, similar to SIMNN as their consequent parts.

3. SINGLE MAPPING NEURAL NETWORKS

On each cluster from the both families $\{X_1, X_2, \ldots, X_{M_\alpha}\}$, $\alpha = 1, 2$, of sub-domains (input clusters) that forms two coverings of the input data $\boldsymbol{x}'$s from $\mathcal{X}$ we construct a single mapping neural network which is a feed-forward neural network.

A standard back–propagation perceptron–type neural network cannot well approximate given set of training pairs and leads to rather high values of an error function. In the present paper, after the extracting knowledge from the set TRE we build a family of mapping neural networks. Each h-th element of the first family $\{X_1, X_2, \ldots, X_{M_1}\}$ and each k-th element of the second family $\{X_1, X_2, \ldots, X_{M_2}\}$ leads to a single network SIMNN, composed of one hidden layer. The number of neurons in the input layer is n, while the number of neurons in the hidden layer is l_h and l_k, respectively. We restrict ourselves to a perceptron type neural network, which is a universal approximator [22, 23]. Hence in the h-th network and k-th network each neuron of hidden layer is equipped with an activation function, say σ_h or σ_k, respectively, which is not a polynomial. The activation function can be taken from the family of two–parameter generalised sigmoidal functions, implemented by the present authors in a number of publications (cf. [17, 18, 21, 30]),

$$\sigma_h(z) = \frac{m_h}{1 + \exp(-\delta_h z)}, \tag{6}$$

and the same for the index k. Both parameters m_h and δ_h give more flexibility in the adaptation process. Moreover, their appearance has given a possibility to design a corrected adaptation algorithm for neural network weight vector [6].

More complex neuron networks are also possible. However, in the present paper output layer nodes (neurons) have the identity activation function, and hence neurons can be characterised by constants, only. Hence the output from the h-th network, denoted by y_h, or from k-th network, denoted by y_k, can be written as

$$y_h = f_h(\boldsymbol{x}, \Omega_h) = \sum_{j=1}^{l} \omega_{hj}^{II} \sigma_h \left(\sum_{i=0}^{n} \omega_{hji}^{I} x_i \right), \qquad y_k = f_k(\boldsymbol{x}, \Omega_k) = \sum_{j=1}^{l} \omega_{kj}^{II} \sigma_k \left(\sum_{i=0}^{n} \omega_{kji}^{I} x_i \right) \tag{7}$$

where $\boldsymbol{\omega}_{hj}^{II}$ and $\boldsymbol{\omega}_{hji}^{I}$, $j = 1, \ldots, l_h$, and $\boldsymbol{\omega}_{kj}^{II}$ and $\boldsymbol{\omega}_{hkji}^{I}$, $j = 1, \ldots, l_k$, $i = 0, \ldots, n$, are constant components of the weight vectors $\boldsymbol{\omega}_h^{II}$, $\boldsymbol{\omega}_{hj}^{I}$, and $\boldsymbol{\omega}_k^{II}$, $\boldsymbol{\omega}_{kj}^{I}$, respectively. Here the zero component x_0 of the input variable $\boldsymbol{x}$ is equal to 1 and was introduced to incorporate the bias $\boldsymbol{\omega}_{hj0}^{I}$ (or $\boldsymbol{\omega}_{kj0}^{I}$) under one summation sign. The vector Ω_h incorporate all above components of weight vector together with parameters r_h and δ_h of the activation function. In the similar way we define the vector Ω_k, for each $k = 1, 2, \ldots, M_2$.

Each SIMNN is trained on the data from TRE belonging to the corresponding cluster, i.e. each f_h is trained on the cluster $\mathcal{K}_h$, with $h = 1, 2, \ldots, M_1$, and each f_k is trained on the cluster $\mathcal{K}_k$, with $k = 1, 2, \ldots, M_2$.

4. ADAPTIVE FUZZY-NEURAL INFERENCE SYSTEM

The next stage of construction of approximation system is to define two families of fuzzy sets A_h and B_k corresponding to the family of clusters $\mathcal{X}_h$ and $\mathcal{X}_k$, respectively, in the input domain. In the paper we assume the membership functions as generalized Gaussian functions

$$\mu_{A_h}(\boldsymbol{x}) = d^h \exp\left(-0.5 \left((\boldsymbol{x} - \boldsymbol{a}^h) \cdot \boldsymbol{S}_{1h}^{-1} (\boldsymbol{x} - \boldsymbol{a}^h) \right)^{b^h} \right), \tag{8}$$

and similar form for B_k. Here $\boldsymbol{S}_h^{-1}$ denotes the inverse of the matrix $\boldsymbol{S}^h$ defined in the previous section on the corresponding cluster. Fuzzy sets A_h and B_k related to the pair of clusters X_{1h} and X_{2k} for each pair (h, k) through the matrices $\boldsymbol{S}_{1h}$, $\boldsymbol{S}_{2k}$ and the centroids $\boldsymbol{p}^{1h} = (\boldsymbol{a}^{1h}, d^{1h})$,

$\boldsymbol{p}^{2k} = (\boldsymbol{a}^{2k}, d^{2k})$ from Eq. (3), will be involved in premise parts of fuzzy rules of the constructed fuzzy inference system as our approximation information system. In the learning process the adaptation will undergo the parameters d^h and b^h.

In our publications ([17]) it was shown that by introducing two additional adaptable parameters d^h and b^h one makes the system more flexible. The parameter d^h has to be non-negative and such that the maximum value of the membership function does not exceed 1. A crucial adaptive features is contained in exponent b^h. Depending on its value (i.e. whether it is smaller or bigger than 1, or even non-negative) we can reach for a particular membership function practically a constant value or a singleton. The negative exponent b^h is also possible.

Constructing the fuzzy inference system for our problem we consider a family $\{\mathcal{R}_m : m = 1, 2, \ldots, Q\}$ of two-conditional rules of the form

$$\textbf{if} \quad \boldsymbol{x} \text{ is } A_h \quad \textbf{and} \quad \boldsymbol{x} \text{ is } B_k \quad \textbf{then} \quad y \text{ is } C_{hk}(\boldsymbol{x}), \tag{9}$$

with $Q = M_1 \cdot M_2$, since all possible pairs (h, k) are admitted, in which the consequent part C_{hk} is not a fuzzy set but a weighted combination of two functions f_h and f_k, namely

$$C_{hk}(\boldsymbol{x}, \Omega_h, \Omega_k) = \mu_{A_h}(\boldsymbol{x})\, f_h(\boldsymbol{x}, \Omega_h) + \mu_{B_k}(\boldsymbol{x})\, f_k(\boldsymbol{x}, \Omega_k). \tag{10}$$

This type of generalised Takagi–Sugeno–Kang's fuzzy rule will lead in the final construction stage.

Now we define the overall output of the fuzzy-neural inference system as

$$z = f(\boldsymbol{x}, \Theta, \Omega) = \sum_{h=1}^{M_1} \sum_{k=1}^{M_2} v(\boldsymbol{x})\, C_{hk}(\boldsymbol{x}, \Omega_h, \Omega_k), \tag{11}$$

where

$$v(\boldsymbol{x}) = \frac{1}{M_2 \sum_{h'=1}^{M_1} \mu_{A_{h'}}(\boldsymbol{x}) + M_1 \sum_{k'=1}^{M_2} \mu_{B_{k'}}(\boldsymbol{x})}. \tag{12}$$

Then at each $\boldsymbol{x}$ the overall activity of all two-conditional rules is normalised to one. Here Ω is a collection of all individual vectors Ω_h, Ω_k, $h = 1, 2, \ldots, M_1$, $k = 1, 2, \ldots, M_2$, and $\Theta = \{(d^h, b^h, (d^k, b^k)) : h = 1, 2, \ldots, M_1,\ k = 1, 2, \ldots, M_2\}$ forms a vector of parameters, that will be adapted in the final stage of the learning process. It is worthwhile to mention that when more different fuzzy domain coverings are constructed, one can assume multi-conditional rules in the module [16, 21].

Notice, that if the fuzzy sets and their membership functions involved will be regarded as ordered fuzzy sets proposed by the first author in [11, 15], then all algebraic operations are for our disposal on the right hand side of Eq. (12)[1].

5. Final stage of learning, implementation and conclusions

Constructed in the last sections the approximation information system (APIS) presented in the form of Eq. (11) needs the last stage of adaptation of the coefficient v of the convex combination. Since only free parameters now are (d^h, b_h) we have to define a new error function.

The form a new error function will be

$$E(\Theta, \boldsymbol{x}, y) := \frac{1}{M} |f(\boldsymbol{x}, \Theta, \Omega) - y|^2 = \frac{1}{Q} \left| \sum_{h=1}^{M_1} \sum_{k=1}^{M_2} v(\boldsymbol{x})\, C_{hk}(\boldsymbol{x}, \Omega_h, \Omega_k) - y \right|^2, \tag{13}$$

where $\Theta = (\Omega_h, \Omega_k)$ is a vector parameter to be adapted.

[1] In his Ph.D. Thesis [24] Prokopowicz has invented different aggregation procedures and methods of determination of the level of activation of two-conditional fuzzy rules with ordered fuzzy numbers, which can be adapted for the case of ordered fuzzy sets, as well.

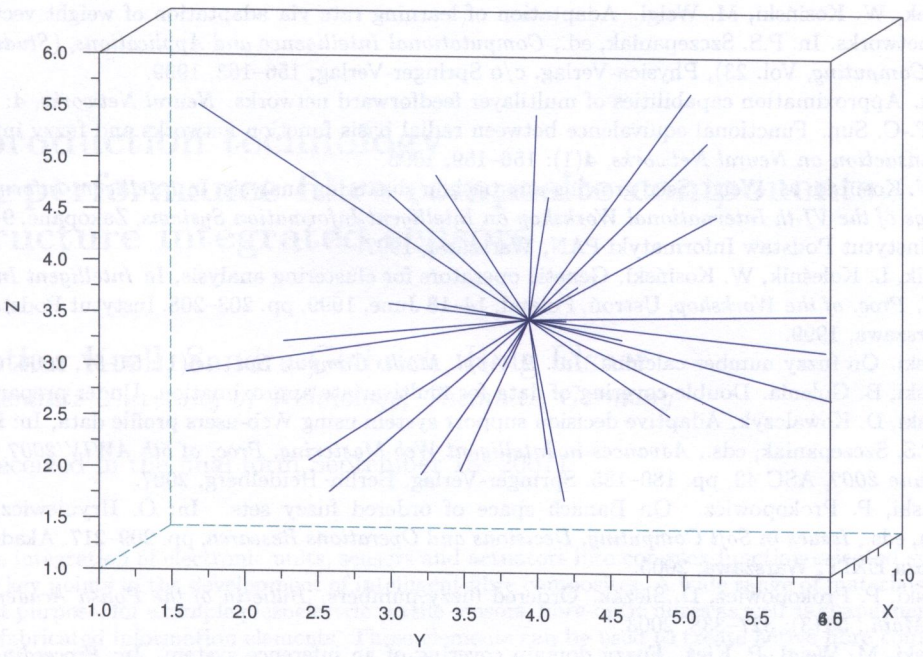

Fig. 1. Cluster of training data for the function (14)

Now the terminal stage of the construction follows: the adaptation of the weight vector Θ. To this end the error function (13) will be minimise over all points $(\boldsymbol{x}, y)$ taken from TRE. The gradient descent method or a genetic algorithm can be implemented for this purpose, since the error function is non-quadratic in the variables Θ. As initial values for each d^h one can take the y-component of the centroid (cf. Eq. (3)) of the cluster $\mathcal{K}_{1h}$ while initial value for b_h could be taken 1.

The presented algorithm has been implemented in C++ and is in a testing stage for 4-D input data. We have adapted the network using 216 elements of training data in TRE and other 125 elements as testing data TES. The training and testing pairs were chosen randomly from the graph of the real-valued function of 3 variables,

$$y = F(x_1\,,\, x_2\,,\, x_3) = \left(1 + x_1^{0.5} + x_2^{-1} + x_3^{-1.5}\right)^2, \tag{14}$$

where $x_1\,,\, x_2\,,\, x_3$ were randomly taken from the interval $[1, 6]$. Output values y were in the interval $[5.101, 22.049]$. The result of this implementation will be presented in the next paper [12]. Here only one of clusters constructed in the covering analysis from 28 elements, is presented in Fig. 1.

Acknowledgments

The research work on the paper was partially done by W.K. in the framework of the KBN Project (Ministry of Science and Higher Education) No. 3 T11C 007 28.

References

[1] M.R. Anderberg. *Cluster Analysis for Applications.* Probability and Mathematical Statistics, Academic Press, New York, 1973.

[2] J.J. Buckley, Y. Hayashi. Numerical relationships between neural networks, continuous functions, and fuzzy systems. *Fuzzy Sets and Systems*, **60**: 1–8, 1993.

[3] G. Cybenko. Approximation by superpositions of sigmoidal function. *Mathematics of Control, Signals, and Systems*, **2**: 303–314, 1989.

[4] K. Funahashi. On the approximate realization of continuous mapping by neural networks. *Neural Networks*, **2**: 183–192, 1989.

[5] D.J. Hand. *Discrimination and Classification.* Wiley, Chichester, 1981.

[6] B. Gołąbek, W. Kosiński, M. Weigl. Adaptation of learning rate via adaptation of weight vector in modified M-Delta networks. In: P.S. Szczepaniak, ed., *Computational Intelligence and Applications*, (*Studies in Fuzziness and Soft Computing*, Vol. 23), Physica-Verlag, c/o Springer-Verlag, 156–163, 1999.

[7] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, **4**: 251–257, 1991.

[8] J. Jang, T.-C. Sun. Functional equivalence between radial basis function networks and fuzzy inference system. *IEEE Transaction on Neural Networks*, **4**(1): 156–159, 1993.

[9] P. Kieś, W. Kosiński, M. Weigl. Seed growing approach in clustering analysis. In *Intelligent Information Systems. Proceedings of the VI-th International Workshop on Intelligent Information Systems*, Zakopane, 9–13 June, 1997, pp 7–15. Instytut Podstaw Informatyki PAN, Warszawa, 1997.

[10] R. Koleśnik, L. Koleśnik, W. Kosiński. Genetic operators for clustering analysis, In *Intelligent Information Systems VIII, Proc. of the Workshop*, Ustroń, Poland, 14–18 June, 1999, pp. 203–208. Instytut Podstaw Informatyki PAN, Warszawa, 1999.

[11] W. Kosiński. On fuzzy number calculus. *Int. J. Appl. Math. Comput. Sci.*, **16**(1): 51–57, 2006.

[12] W. Kosiński, B. Golenia. Double covering of data for multivariate approximation. Under preparation

[13] W. Kosiński, D. Kowalczyk, Adaptive decision support system using Web-users profile data, In: K.M. Węgrzyn-Wolska, P.S. Szczepaniak, eds., *Advances in Intelligent Web Mastering, Proc. of 5th AWIV'2007, Fontainebleau, France, June 2007*, ASC 43, pp. 180–185. Springer-Verlag, Berlin–Heidelberg, 2007.

[14] W. Kosiński, P. Prokopowicz. On Banach space of ordered fuzzy sets. In: O. Hryniewicz, J. Kacprzyk, D. Kuchta, eds., *Issues in Soft Computing, Decisions and Operations Research*, pp. 209–217. Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2005.

[15] W. Kosiński, P. Prokopowicz, D. Ślęzak. Ordered fuzzy numbers. *Bulletin of the Polish Academy of Sciences, Sér. Sci. Math.*, **51**(3): 327–338, 2003.

[16] W. Kosiński, M. Weigl, P. Kieś. Fuzzy domain covering of an inference system. In: *Proceedings of the Third National Conference on Neural Networks and Their Applications*, Kule n. Częstochowa, October 14–18, 1997, pp. 310–321. PNNS, Częstochowa, 1997.

[17] W. Kosiński, M. Weigl. General mapping approximation problems solving by neural networks and fuzzy inference systems. *Systems Analysis Modelling Simulation*, **30**(1): 11–28, 1998.

[18] W. Kosiński, M. Weigl. Adaptive information systems for data approximation problems. In R. Hampel, M. Wagenknecht, N. Chaker, eds., *Fuzzy Control Theory and Practice*, pp. 109–120. Physica-Verlag, Springer-Verlag, Heidelberg–New York, 2000.

[19] W. Kosiński, M. Weigl, Z. Michalewicz. Evolutionary domain covering of an inference system for function approximation. In: V.W. Port, N. Saravanam, D. Waagen, A.E. Eiben, eds., *Evolutionary Programming VII. Proceedings of the 7th International Conference, EP'98*, San Diego, California, USA, March 25–27, 1998. NCS, vol. 1447, pp. 167–180, New York, 1998.

[20] W. Kosiński, M. Weigl, Z. Michalewicz, R. Koleśnik. Genetic algorithms for preprocessing of data for universal approximators. In: *Intelligent Information Systems VII. Proceedings of the Workshop*, Malbork, Poland, 15–19 June, 1998, pp. 320–331, Instytut Podstaw Informatyki PAN, Warszawa, 1998.

[21] D. Kowalczyk. *An Adaptive Fuzzy Inference System as a Fuzzy Controller* (in Polish: Rozmyty adaptacyjny system wnioskujący jako sterownik rozmyty). Master Thesis, WSP w Bydgoszczy, Wydział Matematyki, Techniki i Nauk Przyrodniczych, Specjalność: Technika Komputerowa, Bydgoszcz, czerwiec 1999.

[22] H. Mhaskar, C. Micchelli. Approximation by superposition of a sigmoidal function and radial basis functions. *Advances Applied Mathematics*, **13**: 350–373, 1992.

[23] H. Mhaskar, C. Micchelli. Dimension-independent bounds on the degree of approximation by neural network. *IBM Journal of Research and Development*, **38**(3): 277–284, 1994.

[24] P. Prokopowicz. *Algorithmization of Operations on Fuzzy Numbers and its Applications* (in Polish: Algorytmizacja działań na liczbach rozmytych i jej zastosowania). Ph.D. Thesis, IPPT PAN, kwiecień 2005.

[25] F. Scarselli, A.Ch. Tsoi. Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results. *Neural Networks*, **11**(1): 15–37, 1998.

[26] H. Takagi, M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. on Systems, Man and Cybernetics*, **15**: 116–132, 1985.

[27] M. Weigl, W. Kosiński. Fuzzy inference system and modified back–propagation network in approximation problems. *Proceedings of the III-rd International Symposium on Intelligent Information Systems*, Wigry n. Suwałki, June 1994, pp. 427–442. IPI PAN, Warszawa, 1994.

[28] M. Weigl. *Neural Networks and Fuzzy Inference Systems in Approximation Problems* (in Polish: Sieci neuronowe i rozmyte systemy wnioskujące w problemach aproksymacji). Ph.D. Thesis, IPPT PAN, Warszawa, 1995.

[29] M. Weigl, W. Kosiński. Fuzzy reasoning in adaptive expert systems for approximation problems. In: *Proceedings of the 3-th Zittau Fuzzy-Colloquy*, Zittau, September 5–6, 1995. *Wissenschaftliche Berichte*, Heft 41, 163–174, 1995.

[30] M. Weigl, W. Kosiński. Approximation of multivariate functions by generalized adaptive fuzzy inference network. In: *Proceedings of the 9-th International Symposium on Methodologies for Intelligent Systems ISMIS'96*, Zakopane, June 1996, pp. 120–133. IPI PAN, Warszawa, 1996.