Automatic basis selection for RBF networks using Stein's unbiased risk estimator

Ali Ghodsi
School of Computer Science
University of Waterloo
200 University Avenue West
N2L 3G1, Canada
Email: aghodsib@cs.uwaterloo.ca

Dale Schuurmans
School of Computer Science
University of Waterloo
200 University Avenue West
N2L 3G1,Canada
Email: dale@cs.uwaterloo.ca

***Abstract*— The problem of selecting the appropriate number of basis functions is a critical issue for radial basis function neural networks. An RBF network with an overly restricted basis gives poor predictions on new data, since the model has too little flexibility (yielding high bias and low variance). By contrast, an RBF network with too many basis functions also gives poor generalization performance since it is too flexible and fits too much of the noise on the training data (yielding low bias but high variance). Bias and variance are complementary quantities, and it is necessary to assign the number of basis function optimally in order to achieve the best compromise between them. In this paper we derive a theoretical criterion for assigning the appropriate number of basis functions. We use Stein's unbiased risk estimator (SURE) to derive a generic criterion that defines the optimum number of basis functions to use for a given problem. The efficacy of this criterion is illustrated experimentally.**

I. Introduction

Radial basis function (RBF) networks are a major class of neural network model, where the distance between the input vector and a prototype vector determines the activation of a hidden unit. RBF networks have attracted a lot of interest in the past. One reason is that they form a unifying link between function approximation, regularization, noisy interpolation, classification and density estimation. It is also the case that training radial basis function networks is usually faster than training multi-layer perceptron networks.

RBF network training usually proceeds in two steps: First, the basis function parameters (corresponding to hidden units) are determined by clustering. Second, the final-layer weights are determined by least squares which reduces to solving a simple linear system. Thus, the first stage is an unsupervised method which is relatively fast, and the second stage requires the solution of a linear problem, which is therefore also fast.

One of the advantages of radial basis function neural networks, compared to multi-layer perceptron networks, is the possibility of choosing suitable parameters for the units of hidden layer without having to perform a non-linear optimization of the network parameters. However, the problem of selecting the appropriate number of basis functions remains a critical issue for RBF networks. The number of basis functions controls the complexity, and hence the generalization ability of RBF networks. An RBF network with too few basis functions gives poor predictions on new data, i.e. poor generalization, since the model has limited flexibility. On the other hand, an RBF network with too many basis functions also yields poor generalization since it is too flexible and fits the noise in the training data. A small number of basis functions yields a high bias, low variance estimator, whereas a large number of basis functions yields a low bias but high variance estimator. The best generalization performance is obtained via a compromise between the conflicting requirements of reducing bias while simultaneously reducing variance. This tradeoff highlights the importance of optimizing the complexity of the model in order to achieve the best generalization.

In this paper, we propose a criterion for selecting the number of radial basis functions in an RBF network. To develop a theoretically well motivated criterion for choosing an appropriate number of basis functions , we derive a generalization of Stein's unbiased risk estimator (SURE) [5] that can be used to define a generic criterion which defines the optimum number of basis functions to use in a given problem.

In Section II of this paper we review RBF networks and their training algorithm. We then explain the under-fitting and over-fitting effects caused by using an inappropriate number of basis functions for RBF networks in Section III. In Section IV we derive a generalization of SURE, and in Section V show how it can be applied to RBF networks. Experimental results of the proposed criterion and its performance are presented in Section VI.

II. Radial Basis Function Networks

Radial basis function methods became a popular technique in the mid 80s for performing exact interpolation of a set of data points in a high-dimensional space [7]. The basic technique provides an interpolating function which passes through every data point: Consider a mapping from a d-dimensional input space to a one-dimensional target space y, where the data set consists of N input vectors X_i, with corresponding targets y_i, $i = 1...N$. An exact interpolation is achieved by introducing a set of N basis functions, one for each data point, and then setting the weights for the linear combination of basis functions. Here, basis functions are non-linear functions, $\Phi(||x - x_i||)$, of the input vectors x_i, and a linear combination of these basis functions can be written as

$$\sum_{i=1}^{N} w_i \Phi(||x - x_i||)$$

The interpolation problem can then be written in a matrix

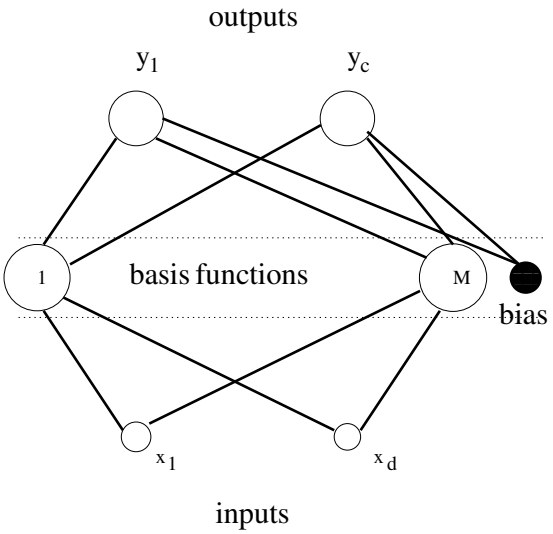

Fig. 1. A graphical form of a radial bassis function neural network architecture. Basis functions act like hidden units. The corresponding elements μ_{ji} of the vector μ_j is represented by the line connecting basis function Φ_j to the inputs. The weights w_{kj} are shown as lines from the basis functions to the output units. An extra basis function whose output is fixed at 1 serves as the bias for each output unit.

form as

$$\Phi W = y$$

where the square matrix Φ has elements $\Phi_{ii'} = \Phi(||x_i - x_{i'}||)$. This linear system of equations can be solved to yield

$$W = \Phi^{-1} y$$

For the case of Gaussian basis functions we have

$$\Phi(x) = \exp\left(-\frac{x^2}{2v^2}\right)$$

where v is a parameter that controls the smoothness of the interpolating function.

A radial basis function neural network model [2], [6] can be obtained by a number of modifications to the exact interpolation procedure as follows: First, the number, M, of basis functions is usually much less than the number, N, of data points. Second, the centers of the basis functions no longer need to be given by input data vectors, and appropriate centers can alternatively be determined during the training process. Third, unlike the exact interpolation procedure, each basis function can have its own width parameter, v_j, whose value is also determined in the training process. Finally, by applying these changes to the original (exact) interpolation formula we obtain the following form for the radial basis function neural network mapping.

$$y_k(X) \quad = \quad \sum_{j=1}^{M} w_{kj}\Phi_j(X) + w_{k0} \qquad (1)$$

By including an extra basis function Φ_0 whose activation is set to 1, the biases w_{k0} can be absorbed into the final summation. Another useful variation is the normalized RBF representation:

$$y_k(X) = \frac{\sum_{j=1}^{M} w_{kj}\Phi_j(X)}{\sum_{r=1}^{M} \Phi_r(X)}$$

This normalized representation is closely related to TSK fuzzy inference systems [9][10].

Several forms of basis function have been considered in previous research on RBF models, the most common being the Gaussian:

$$\Phi_j(X) = \exp\left(-\frac{||X - \mu_j||^2}{2v_j^2}\right)$$

where X is the d-dimensional input vector with elements x_i, and μ_j is the center of basis function Φ_j.

In practice, training RBF networks proceeds through two steps. The first step determines the first layer of weights, in which the basis function parameters μ_j and σ_j are selected based on the X-values of the training samples (via unsupervised learning techniques). The basis functions are then kept fixed while the second-layer weights w_i, are estimated via linear least squares.

Typical approaches for the first phase include using the generalized Lloyd algorithm (GLA) and Konhonen's self-organized maps (SOM). Another common approach is to model the input distribution as a Gaussian mixture model and then estimate the center and width parameters of the Gaussian mixture components via the EM algorithm [1]. For the second phase one can consider the radial basis function network mapping in (1). If we absorb the bias parameters into the weights, this can be written in matrix notation as

$$Y \quad = \quad W\Phi \qquad (2)$$

where Y is a matrix of output values, and $W = (w_{kj})$ is a matrix of second-layer weights to be estimated.

This is a classical least squares estimation problem. A necessary condition for $||Y - W\Phi||^2$ to be minimized is that W must satisfy [1]

$$W = (\Phi^T\Phi)^{-1}\Phi^T Y$$

III. Model validation

Learning a radial basis function network from data is a parameter estimation problem. One difficulty with this problem is selecting parameters that show good performance both on training and testing data. In principle, a model is selected to have parameters associated with the best observed performance on training data, although our goal really is to achieve good performance on unseen testing data. Not surprisingly, a model selected on the basis of training data does

[1] A unique solution exists as long as the columns of Φ are linearly independent. This will be true in most cases when the number of basis is smaller than the number of data points. Under this condition W can be estimated via $W = (\Phi^T\Phi)^{-1}\Phi^T Y$. For the case where the columns of Φ are not linearly independent, equation (2) are solved using singular value decomposition, to avoid problems due to ill-conditioning of the matrix Φ.

not necessarily exhibit comparable performance on the testing data. When squared error is used as the performance index, a zero-error model on the training data can always be achieved by using a sufficient number of basis functions. However, training error, err, and testing error, Err, do not demonstrate a linear relationship. In particular, a smaller training error does do not necessarily result in a smaller testing error. In practice, one often observes that, up to a certain point, the model error on testing data tends to decrease as the training error decreases. However, if one attempts to decrease the training error too far by increasing model complexity, the testing error often can take a dramatic increase.

The basic reason behind this phenomenon is that in the process of minimizing training error, after a certain point, the model begins to over-fit the training set. Over-fitting in this context means fitting the model to training data at the expense of losing generality. In the extreme form, as we mentioned in the previous section, a set of N training data points can be modeled exactly with N radial basis functions. Such a model follows the training data perfectly. However, the model is not representative features of the true underlying data source, and this is why it fails to correctly model new data points.

In general, the training error rate err will be less than the testing error on the new data, Err. A model typically adapts to the training data, and hence the training error err will be an overly optimistic estimate of the generalization error Err. An obvious way to estimate generalization error is to estimate the degree of optimism OP inherent in a particular estimate, and then add a penalty term to the training error to compensate, i.e., such that $Err = err + OP$. The method described in the next section works in this way.

IV. Estimating the optimism

Let:

- $MSE(\widehat{f}) = E(\widehat{f} - f)^2$.
- $\widehat{f}(X)$ denote the *prediction model*, which is estimated from a training sample by the RBF neural network model.
- $f(X)$ denote the *real model*.
- err denote the *training error*, which is the average loss over the training sample.
- Err denote the *generalization error*, which is the expected prediction error on an independent test sample.

Recall that the training error, $err = \sum_{i=1}^{N}(\widehat{y} - y)^2$, is an estimate of the expectation of the squared error on the training data, $E(\widehat{y} - y)^2$, while the generalization error (test error) Err is an estimate of mean squared error, $MSE = (\widehat{f} - f)^2$, where $\widehat{f}(X)$ is the estimated model and $f(X)$ is the true model.

Now suppose $y_i = f(x_i) + \varepsilon_i$, where ε is additive Gaussian noise $N(0, \sigma^2)$. We need to estimate $\widehat{f}$ from training data $D = \{(x_i, y_i)\}_i^n$. Consider

$$\begin{aligned} E[(\widehat{y_0} - y_0)^2] &= E[(\widehat{f} - f - \varepsilon)^2] \\ &= E[(\widehat{f} - f)^2] + E[\varepsilon^2] - 2E[\varepsilon(\widehat{f} - f)] \quad (3) \\ &= E[(\widehat{f} - f)^2] + \sigma^2 - 2E[\varepsilon(\widehat{f} - f)] \quad (4) \end{aligned}$$

Here, the last term can be written as:

$$E[2\varepsilon(\widehat{f} - f)] = 2E[(y_0 - f)(\widehat{f} - f)] \equiv cov(y_0, \widehat{f})$$

We consider two different cases.

a) Case 1.: Consider the case in which a new data point has been introduced to the estimated model, $i.e.(x_0, y_0) \notin D$. Since y_0 is a new point, $\widehat{f}$ and y_0 are independent. Therefore $cov(y_0, \widehat{f}) = 0$ and (4) in this case can be written as:

$$E[(\widehat{f} - f)^2] = \sigma^2 - E(\widehat{y_0} - y_0)^2 \quad (5)$$

This is the justification behind the technique of cross validation. In cross validation, to avoid overfitting or underfitting, a validation data set is used which is independent from the estimated model. The optimal model parameters should be selected to have the best performance index associated with this data set. Since this data set is independent from the estimated model, it is a fair estimate of $E(\widehat{f} - f)^2$ and consequently of generalization error Err as indicated in (5).

b) Case 2.: A more interesting case is the case in which we do not use new data points to assess the performance of the estimated model, and the traing data is used for both estimating and assessing a model $\widehat{f}$. In this case the cross term in (4) cannot be ignored because $\widehat{f}$ and y_0 are not independent. Therefore the cross term, which is $cov(y_0, \widehat{f})$, is not zero. However the cross term can be estimated by Stein's lemma [5] [8], which was originally proposed to estimate the mean of a Guassian distribution [8].

According to Stein's lemma if $X \sim N(\theta, \sigma^2)$ and $g(x)$ is differentiable function then $E(g(x)(x-\theta)) = \sigma^2 E(g'(x))$. So we let

$$g(\varepsilon) = \widehat{f} - f = \widehat{f} - y - \varepsilon$$

and $x = \varepsilon$. Then by applying Stein's lemma we obtain

$$E(\varepsilon(\widehat{f} - f)) = \sigma^2 E(g'(\varepsilon)) = \sigma^2 E(\frac{d\widehat{f}}{dy})$$

Summing over all y we get

$$\begin{aligned} Err &= \sum_{i=1}^{N}(\widehat{y} - y)^2 - N\sigma^2 + 2\sigma^2 \sum_{i=1}^{N} \frac{d\widehat{f}(x_i)}{dy_i} \\ &= err - N\sigma^2 + 2\sigma^2 \sum_{i=1}^{N} \frac{d\widehat{f}(x_i)}{dy_i} \quad (6) \end{aligned}$$

This is known as Stein's Unbiased Risk Estimator (SURE).

V. Determining the optimum number of basis functions

Based on this criterion, the optimum number of clusters should be assigned to have the minimum generalization error Err in (6). From the least squared solution of (2) we have:

$$\begin{aligned} W &= (\Phi^T\Phi)^{-1}\Phi^T Y \\ \widehat{f} &= \Phi W = \Phi(\Phi^T\Phi)^{-1}\Phi^T Y = HY \quad (7) \end{aligned}$$

where H depends on the input vector x_i but not on y_i. Note that in practice, the equation (2) is solved using singular

value decomposition to avoid problems due to possible ill-conditioning of the matrix Φ.

From (7) we can easily obtain the required derivative of $\widehat{f}(x_i)$ with respect to y_i.

$$\sum_{i=1}^{N} \frac{d\widehat{f}(x_i)}{dy_i} = \sum_{i=1}^{N} H_{ii}$$

Now, substituting this into (6) we obtain

$$Err \quad = \quad err - N\sigma^2 + 2\sigma^2 \sum_{i=1}^{N} H_{ii}$$

Here we observe that $\sum_{i=1}^{N} H_{ii} = Trace(H)$, the sum of the diagonal elements of H. Thus, we can obtain the further simplification that $Trace(H) = Trace(\Phi(\Phi^T\Phi)^{-1}\Phi^T) = Trace(\Phi^T\Phi(\Phi^T\Phi)^{-1}) = Trace(I) = P$, where P is the dimension of Φ. Since Φ is a projection of input matrix X onto a basis set spanned by M, the number of basis functions, one can show generally $P = M + 1$.

To use this method to find the optimum number of clusters, we simply choose the model that obtains the smallest Err over the set of models considered. Given a set of models $\widehat{f}_M(x)$ indexed by the number of basis functions, M, denote the training error for each model by $err(M)$. We then obtain

$$Err(M) \quad = \quad err(M) - N\sigma^2 + 2\sigma^2(M+1) \qquad (8)$$

where N is the number of training samples and the noise, σ^2, can be estimated from the mean squared error of the model.

VI. Experimental results

To explore the effectiveness of our complexity control method, we considered the problem of fitting a RBF network model to a set of points (Figure 2). The goal is to minimize the squared generalization error Err. To determine the efficacy of the method we compared its performance to the well studied standard cross validation [4].

We first conducted a simple series of experiments by fixing a uniform distribution on the unit interval [0,1], and then fixing various target functions $f : [0,1] \rightarrow R$. To generate training samples, a sequence of values $x_1, ..., x_t$ is drawn from $[0,1]$, the target function values $f(x_1), .., f(x_t)$ are computed, and independent Gaussian noise is added to each value. In the first step of RBF network training, centers μ_j and width parameter v_j are estimated using subtractive clustering [3], an unsupervised training technique.

For a given training sample, the series of best fit functions corresponding to a number of basis functions M= 1, 2, .. ,*etc*. are computed. Given this sequence, the cross validation strategy will choose some particular model $\widehat{f}^*_M$ on the basis of the observed empirical errors on the validation data set (generated the same way as training data). Our technique will alternatively chose the model corresponding to minimum Err in (8). To determine the effectiveness of these two strategies, the ratio of the test error of the model selected by them to the best test error on a new test data set among the models in sequence $M = 1, 2, ..$ is measured.

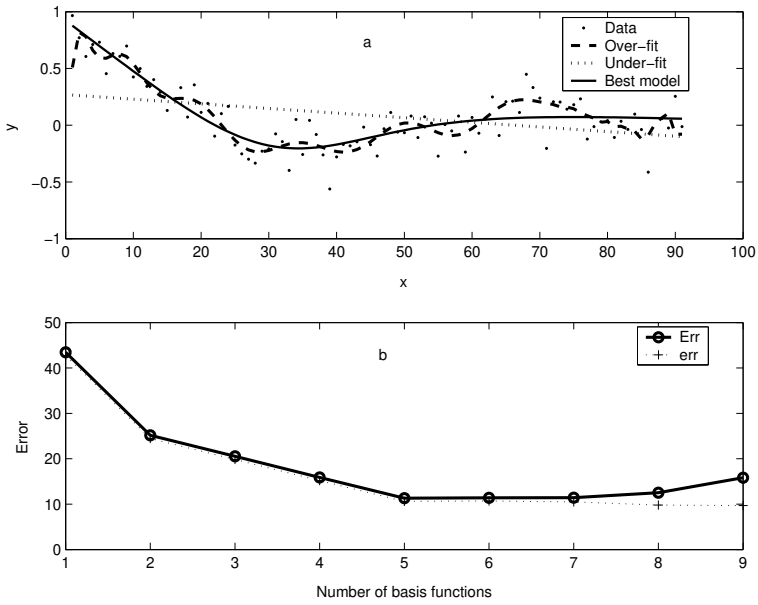

Fig. 2. (a)Overfitting, underfitting, and the best estimated model for $y = \frac{sin(x)}{x}$. (b) Err obtained in (8) used to find the optimum number of clusters for model $y = \frac{sin(x)}{x}$.

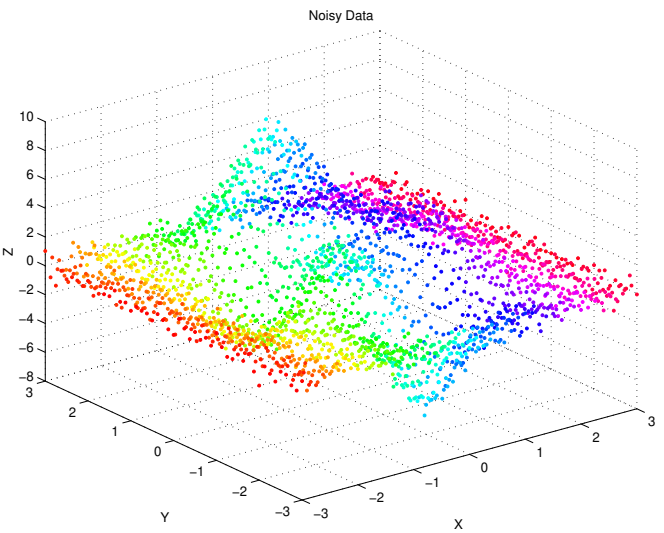

Fig. 3. Data generated in $3D$ dimensional space by nonlinear function 'Peaks' and distorted by additive Gaussian noise $N(0, 0.25)$.

Table 1 shows the results obtained for fitting various functions. These results are obtained by repeatedly generating training samples of a fixed size, and recording the ratio of test error achieved relative to the best possible test error, for each technique (CV and SURE).

In another experiment we considered a highly nonlinear function distorted by Gaussian noise.

		CV		SURE	
Target function	N	Test error ratio	RBF diff.	Test error ratio	RBF diff.
step($x \geq 0.5$)	100	1.013	0.15	1.011	-0.06
$sin(\frac{1}{x})$	100	1.005	0.19	1.006	0.13
$sin^2(2\pi x)$	100	1.02	0.05	1.02	-0.05
step($x \geq 0.5$)	200	1.008	0.2	1.008	0.00
$sin(\frac{1}{x})$	200	1.005	0.29	1.005	0.31
$sin^2(2\pi x)$	200	1.017	0.2	1.015	-0.01

TABLE I

FITTING DIFFERENT TARGET FUNCTIONS WITH $\sigma = 0.25$. TABLE REPORTS RATIO OF TEST ERRORS RELATIVE TO BEST POSSIBLE TEST ERROR ACHIEVED BY DIFFERENT METHODS. A SMALLER RATIO IS BETTER. RESULTS ARE REPORTED AT TRAINING SAMPLE SIZES $N = 100$ AND $N = 200$ AND AVERAGED OVER 100 REPEATED TRIALS IN EACH CASE. COLUMNS 4 AND 6 SHOW THE DIFFERENCE BETWEEN THE OPTIMUM NUMBER OF RBF FUNCTIONS AND THE NUMBER OF RBF FUNCTIONS CHOSEN BY CV AND SURE RESPECTIVELY.

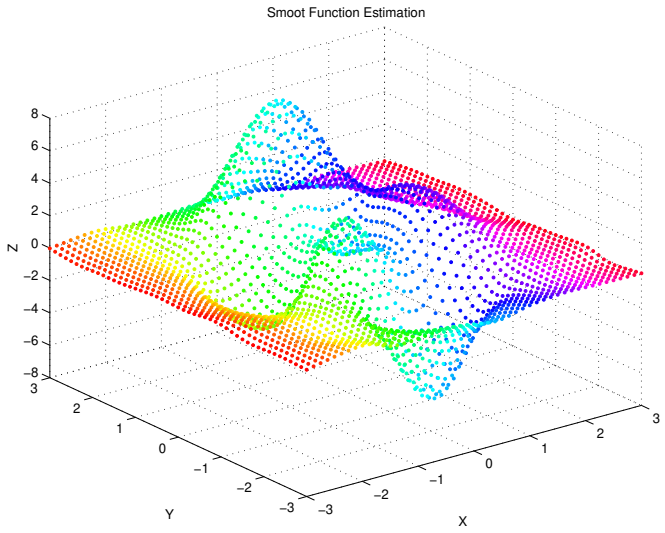

Fig. 4. The smooth true function 'Peaks' has been recovered from noisy observation. Err obtained in (8) used to find the optimum number of radial basis functions for the model.

$$z = 3(1-x)^2 e^{(-(x^2)-(y+1)^2)} - 10(\frac{x}{5} - x^3 - y^5)e^{(-x^2-y^2)} - \frac{1}{3}e^{(-(x+1)^2-y^2)} + \varepsilon$$

$$\varepsilon \sim N(0, 0.25)$$

We call this function Peaks.

Here the goal is to estimate the true smooth function based on noisy observations (Figure 3). In this experiment, our proposed criterion SURE and cross validation CV chose 35 basis functions , and therefore they achieved the same generalization error on the test data. However, the SURE technique does not require an extra validation set to choose the number of basis functions, and in fact used half the data available to CV in this case. Therefore, we claim that it achieved more effective performance on this example.

VII. CONCLUSION

We have proposed a new approach to choosing the optimum number of basis functions for RBF networks. Our approach minimizes a theoretically unbiased estimate of generalization error of the model. Our experimental results validate the effectiveness of this approach. A comparison cross validation illustrates that the generalization error of the models selected by our approach can be less than models selected by cross validation. Importantly, this is achieved while requiring much less computation than cross validation. The utility of our method is greatest when there is insufficient data to hold out a validation set for cross validation.

REFERENCES

[1] Bishop, C.M., *Neural Networks for Pattern Recognition*, Oxford, Oxford University Press, (1995)

[2] Broomhead, D.S. and Lowe D. *Multivariable functional interpolation and adaptive networks.* Complex Systems **2** (1988), 321-355.

[3] Chiu S.L,, *Fuzzy model identification based on cluster estimation.* Journal of Intelligent and Fuzzy System. **2** (1994) 267–278

[4] Craven P., Wahba G, *Smoothing noisy data with spline functions.* Number. Math **31** (1979) 377–403

[5] Ker-Chau L, *From Stein's Unbiased Risk Estimates to the Method of Generalized Cross Validation.* Annals of Statistics. **13(4)** (1985) 1352–1377

[6] Moody, J. and Darken C. J.,*Fast learning in networks of locally-tuned processing units.* Neural Computation **1** (2), 1989, 281–294

[7] Powell, M.J.D., *Radial basis functions for multivariable interpolation:* a review. In J.C. Mason and M.G. Cox (Eds.), Algorithms for Approximation, pp. 143-167. Oxford: Clarendon Press.

[8] Stein M. C, *Estimation of the Mean of a Multivariate Normal Distribution.* Annals of Statistics.**9(6)** (1981) 1135–1151

[9] Sugeno M. and Yasukawa T, *A fuzzy-logic-based approach to qualitative modeling.* IEEE Trans. Fuzzy Syst. **l** (1993) 7–31

[10] Takagi T. , Sugeno M, *Fuzzy identification of systems and its application to modeling and control.* IEEE Trans. Syst. Man & Cybern. **SMC-15(3)** (1985) 116–132