An Incremental Multivariate Regression Method for Function Approximation from Noisy Data

Menita Carozza & Salvatore Rampone
Facoltà di Scienze MM.FF.NN., Università del Sannio
Via Port' Arsa 11
I-82100 Benevento ITALY

Abstract

In this paper we consider the problem of approximating functions from noisy data. We propose an incremental supervised learning algorithm for RBF networks. Hidden gaussian nodes are added in an iterative manner during the training process. For each new node added, the activation function center and the output connection weight are settled according to an extended chained version of the Nadaraja-Watson estimator. Then the variances of the activation functions are determined by an empirical risk driven rule based on a genetic-like optimization technique.

Keywords: Function Approximation, Noisy Data, Network Size, Genetic Algorithm, Generalization.

1 Introduction

Neural networks are efficient non-parametric models for approximating bounded and continuous functions from empirical data. In particular Radial Basis Function (RBF) Networks form a unifying link within a number of disparate concepts. It motivates the use of radial basis functions from the point of view of function approximation, regularization, noisy interpolation, density estimation, optimal classification theory, and potential functions [1]. Further motivation for the use of radial basis functions in the estimating regression functions from noisy data comes from kernel regression theory [2].

However, existence theorems do not give methods to define a network architecture (mainly the number of neurons in the hidden layers) well-suited to the data, i.e. allowing a good approximation without overfitting [3]. By using a decomposition of the generalization error in estimation and approximation errors, Niyogi and Girosi [4] investigated the relationship between the number of network parameters and the training set size. They showed that simply fixing the number of data and increasing the number of parameters, on average, the estimation error grows. Namely they proved the number of network parameters should grow more slowly than the number of data points.

Furthermore RBF network training is usually divided in two stages. In the first stage, the parameters governing the basis functions (corresponding to hidden units) are determined using unsupervised methods on the input data. The second stage involves the determination of the final layer weights. In the first stage, the setting up of the basis functions, using density estimation on the input data, takes no account of the target labels associated with that data. In order to optimize the basis function parameters with respect to the required network outputs, target data should be included in the training procedure, i.e. we should perform supervised, rather than unsupervised, training [5]. Moreover the price we would have to pay is that the process of setting up the model using the data set is computationally intensive. This drawback many times makes other techniques more attractive [6].

Finally we have to consider regression problems where there is noise on the output, sometimes in an input-dependent manner [7].

In this paper we propose an incremental RBF network for function approximation from noisy data. Hidden gaussian nodes are added in an iterative manner during the training process. For each new added node, the activation function center and the output connection weight are settled according to an extended chained version of the Nadaraja-Watson estimator [1] [8] [9]. Then the variances of the activation functions are determined by an empirical risk driven rule based on a genetic-like optimization technique [10]. We also consider a theoretical upper bound to the number of hidden units [4].

Assuming the data affected by noise, an optimal network approximation must provide a residual error at least equal to the noise. Then we stop the adding procedure when the residual error is under a fixed threshold related to the noise level.

Our results extend to noisy data the strategy of Esposito et al. [11], and gives a precise theoretical insight. Moreover let us note that, with respect to other methods [3], such results appear better in terms of approximation and generalization capability, and computational cost.

This paper is organized as follows. In Section 2 we describe the used incremental estimator, and, in Section 3, the network training mechanism, bounding the network nodes and summarizing all the results in a single learning procedure. Finally, in Section 4, we describe the experimental results.

2 Regression Method

Let $\mathbf{X}$ and Y be two bounded subsets of $\mathcal{R}^n$ and $\mathcal{R}$ respectively. We consider the problem of approximating smooth functions

$$f : \mathbf{X} \rightarrow Y \tag{1}$$

given a set of training data

$$\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \ldots, (\mathbf{x}_p, t_p)\} \tag{2}$$

where

$$\mathbf{x}_k \in \mathbf{X}, \qquad t_k = f(\mathbf{x}_k) + rand[-a, a], \tag{3}$$

and $rand[-a, a]$ is a random additive noise from a zero mean uniform distribution[1].

Our goal is to find a smooth mapping which captures the underlying systematic aspects of the data without fitting the noise on the data.

2.1 Estimator

Given the set (2) of training data, the complete description of the statistical properties of the data generator is given by the unknown probability density

$$p(\mathbf{x}, t) \tag{4}$$

[1] We assume the uniform noise for generality, while the results hold even in the gaussian case.

in the joint input-target space. Let us consider a subset of (2)

$$\{(\mathbf{x}_{k_1}, t_{k_1}), (\mathbf{x}_{k_2}, t_{k_2}), \ldots, (\mathbf{x}_{k_M}, t_{k_M})\} \tag{5}$$

where $k_j \in \{1, 2, \ldots, p\}$ and $M \leq p$, and the following form of radial basis function

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{||\mathbf{x} - \mu_j||^2}{2\sigma_j^2}\right) \tag{6}$$

where μ_j is the center, and σ_j^2 the variance. We use as an estimator of (4) an adaptive mixture model

$$\overline{p}(\mathbf{x}, t) = \frac{1}{M}\sum_{j=1}^{M} P(j)\frac{1}{(2\pi\sigma_j^2)^{\frac{n+1}{2}}}\exp\left(-\frac{||\mathbf{x} - \mathbf{x}_{k_j}||^2}{2\sigma_j^2} - \frac{||t - w_j||^2}{2\sigma_j^2}\right) \tag{7}$$

where we assume

$$P(j) = \frac{1}{c}\int_{\mathcal{R}^n}\exp\left(-\frac{||\mathbf{x}||^2}{\sigma_j^2}\right)d\mathbf{x} = \frac{(\pi\sigma_j^2)^{\frac{n}{2}}}{c}; \qquad c \geq \sum_{j=1}^{M}(\pi\sigma_j^2)^{\frac{n}{2}} \tag{8}$$

Then we have the following expression for the regression of noisy data

$$net_M(\mathbf{x}) = E(t|\mathbf{x}) = \frac{\sum_{j=1}^{M} w_j \exp(-||\mathbf{x} - \mathbf{x}_{k_j}||^2/2\sigma_j^2)}{\sum_{j=1}^{M}\exp(-||\mathbf{x} - \mathbf{x}_{k_j}||^2/2\sigma_j^2)} \tag{9}$$

where w_j will be specified in the following. By setting $w_j = t_{k_j}$ this can be viewed as a RBF network based on the Nadaraya-Watson estimator [1], extended to the multivariate case [12], and restricted to a subset of data.

The estimator performance is valued by means of the the empirical risk

$$E_M = \frac{1}{p}\sum_{k=1}^{p}(t_k - net_M(\mathbf{x}_k))^2 \tag{10}$$

2.2 Incremental Estimator

The estimator (9) can be easily chained to obtain an incremental approach

$$net_M(\mathbf{x}) = \frac{net_{M-1}(\mathbf{x}) den_{M-1}(\mathbf{x}) + w_M \exp(-||\mathbf{x} - \mathbf{x}_{k_M}||^2/2\sigma_M^2)}{den_M(\mathbf{x})} \tag{11}$$

where

$$den_M(\mathbf{x}) = \sum_{j=1}^{M} \exp(-||\mathbf{x} - \mathbf{x}_{k_j}||^2/2\sigma_j^2) \tag{12}$$

To explicitly reduce the residual approximation error given by (10), in (9) we assign to w_j the value

$$w_j = t_{k_j} - net_{j-1}(\mathbf{x}_{k_j}) \tag{13}$$

where k_j is such that

$$(t_{k_j} - net_{j-1}(\mathbf{x}_{k_j}))^2 = \max_{k=1,\ldots,p} (t_k - net_{j-1}(\mathbf{x}_k))^2 \tag{14}$$

i.e. the training point in which net_{j-1} has the worst approximation performance.

Moreover, since this quantity is affected by the normalization factor den_j, we multiply w_j for this factor, so having

$$net_M(\mathbf{x}) = \frac{\sum_{j=1}^{M} w_j den_j(\mathbf{x}) \exp(-||\mathbf{x} - \mathbf{x}_{k_j}||^2/2\sigma_j^2)}{\sum_{j=1}^{M} \exp(-||\mathbf{x} - \mathbf{x}_{k_j}||^2/2\sigma_j^2)} \tag{15}$$

or, in the chained version

$$net_M(\mathbf{x}) = \frac{net_{M-1}(\mathbf{x}) den_{M-1}(\mathbf{x})}{den_M(\mathbf{x})} + w_M \exp(-||\mathbf{x} - \mathbf{x}_{k_M}||^2/2\sigma_M^2) \tag{16}$$

3 Network Training

The training phase consists of an iterative process which adds hidden nodes until a target error level is reached. At each step M, the RBF center is selected by (14). Given (15), we have only to estimate the σ_j^2 values.

3.1 Variance Set Up

The training of the variances consists in two stages: a "local training", where only the last added node variance is updated, and a "fine tuning" of all the net variances. These parameters are optimized by means of a genetic-like algorithm. A population of individuals is generated by applying the following mutation

$$\sigma_{j,new}^2 \leftarrow \sigma_j^2(1+\psi) \tag{17}$$

where ψ is a random number in a fixed range $[-\alpha, \alpha]$. If the empirical risk associated to $\sigma_{j,new}^2$ is less than the same quantity associated to σ_j^2, then $\sigma_j^2 \leftarrow \sigma_{j,new}^2$. Each mutation step, corresponding to the selection of $\sigma_{j,new}^2$ in a range

$$I = [\sigma_j^2(1-\alpha), \sigma_j^2(1+\alpha)] \tag{18}$$

is iterated a fixed number of times (tr) since

Theorem 1 *Let $I_1, I_2, \ldots, I_\gamma$ be disjoint subsets of $\mathcal{R}$ such that $\cup_{k=1}^{\gamma} I_k = I$, and $|I_k| = |I_q|, \quad \forall k, q \in \{1, \ldots \gamma\}$. Let I_m be a subset in which $E'_M < E_M$, where E'_M is the empirical risk of net_M with $\sigma_{j,new}^2 \in I_m$, and E_M the empirical risk of net_M with σ_j^2. Then the expected number of drawings in I (ψ selections) necessary to acquire at least a value in I_m is*

$$\gamma \log \gamma \tag{19}$$

Proof: The result is easily proved by considering that to draw a value in I corresponds to sampling with replacement a population of at most γ classes uniformly distributed (see [13] example IX.3.d, [14])

[]

3.2 Node Number

The average error of net_M estimator, the expected risk [4], is unknown because our only source of information is the data set (2). So we approximate the expected risk by the empirical one, i.e. by the sample mean square error (10).

By a Niyogi and Girosi result [4], in order to guarantee the empirical risk converges to the expected one, the number M of RBF units must be

$$M = p^r, \quad r < 1 \tag{20}$$

In this section we bound the network node number M, and so the size of the subset (5), by viewing it as a function of the given number of examples (2). Let

$$f_{M,p} = arg \min_{\sigma_j^2} \sum_{k=1}^{p} (t_k - net_M(\mathbf{x}_k))^2 = arg \min_{\sigma_j^2} E_M \tag{21}$$

Theorem 2 *Let H_M be the class of RBF network with n input nodes and M hidden nodes as defined in (15) and f be the target function (1), element of the Bessel potential space $\mathcal{L}_1^m(\mathcal{R}^n)$ of order m, with $m > n/2$. Given the data set (2) is obtained by randomly sampling f in presence of noise, and that the noise distribution has compact support, then for any $0 < \delta < 1$, with probability greater than $1 - \delta$, the following bound for the generalization error holds*

$$||f - f_{M,p}||^2_{L^2(P)} \leq O\left(\frac{1}{M}\right) + O\left(\left[\frac{Mn\ln(Mp) - \ln\delta}{p}\right]^{1/2}\right) \tag{22}$$

Proof: We show our approximation scheme (15) matches the requirements of the Niyogi and Girosi result [15] [4]. Then the result is straightforward. Namely we have to prove that

net_M belongs to the class $\{f | f = \sum_{j=1}^M \beta_j \exp(-||\mathbf{x} - \mu_j||^2 / 2\sigma_j^2)$ and $\sum_{j=1}^M \beta_j$ is bounded$\}$.

The estimator (15) can be viewed as

$$net_M(\mathbf{x}) = \sum_{j=1}^{M} \frac{w_j den_j(\mathbf{x})}{den_M(\mathbf{x})} \exp(-||\mathbf{x} - \mathbf{x}_{k_j}||^2 / 2\sigma_j^2) \tag{23}$$

We set

$$\frac{w_j den_j(\mathbf{x})}{den_M(\mathbf{x})} = \beta_j(\mathbf{x}) \tag{24}$$

so having

$$net_M(\mathbf{x}) = \sum_{j=1}^{M} \beta_j(\mathbf{x}) \exp(-||\mathbf{x} - \mathbf{x}_{k_j}||^2 / 2\sigma_j^2) \tag{25}$$

At each step M, the RBF center is the point $\mathbf{x}_{k_M}$ such that

$$(t_{k_M} - net_{M-1}(\mathbf{x}_{k_M}))^2 = \max_{k=1,\ldots,p} (t_k - net_{M-1}(\mathbf{x}_k))^2 \tag{26}$$

By (13), this implies

$$|w_j| \leq 2|t_{k_1}| \quad \forall j \tag{27}$$

Since

$$\frac{den_j(\mathbf{x})}{den_M(\mathbf{x})} \leq 1; \quad j = 1, 2, \ldots, M \tag{28}$$

we have

$$\sum_{j=1}^{M} \beta_j(\mathbf{x}) \leq 2M|t_{k_1}| = \overline{M} \tag{29}$$

[]

By this result we are in the hypotheses of the Niyogi and Girosi theorem, and so (22) holds. In this way, as in [4], we obtain an upper bound for the node number M, namely, given p examples, we have $M \leq p^r$.

3.3 Learning Algorithm

The learning algorithm can be summarized as follows

Learning Algorithm

Step 1: *Input:* $G = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \ldots (\mathbf{x}_p, t_p)\}$ the set of training instances, T the target training approximation error, r the exponent that maximizes M, α the σ_j^2 mutation parameter, tr the number of search iteration of each σ_j^2.

Initialization: Set $M = 0$.

Step 2: While the empirical risk $E_M > T$ and $M < p^r$ do

2.1 *Choose the center:* Select k such that $|t_k - net_M(\mathbf{x}_k)|$ is maximum.

2.2 *Add a new node:* Set $M \leftarrow M + 1$, initialize σ_M^2, $w_M = t_k - net_{M-1}(\mathbf{x}_k)$, $\mu_M = \mathbf{x}_k$.

2.3 *Adapt σ_j^2 parameters:* For all nodes $j, j \leq M$, starting from the last do

2.3.1 While E_M decreases do

2.3.1.1 For tr times do

- *Mutate σ_j^2:* $\sigma_{j,new}^2 \leftarrow \sigma_j^2(1 + rand[-\alpha, \alpha]) + \varepsilon$. Compute the corresponding risk $E_{M,new}$.

- *Update σ_j^2:* if $E_{M,new} < E_M$ then $\sigma_j^2 \leftarrow \sigma_{j,new}^2$.

Step 3: Output net_M.

4 Experiments

The algorithm has been tested on several functions. The choice of the initial σ_j^2 value seems not to influence the algorithm behavior; we will fix it to 0.95. Furthermore tr will be fixed to 10, and α to 0.5.

4.1 Comparison to no-noisy data results

Here we compare our experimental results to the ones obtained by other authors [11] [16] on the same function set with no-noisy data.

Table 1 shows our results for $y = \sin(x) + \sin(3x) + \sin(6x)$ in the range $[-0.35, 3.50]$, Table 2 shows the results for $y = 0.5\exp(-x) + \sin(6x)$ in the range $[-1, 1]$, and Table 3 shows the results for $y = x$ in the range $[0, 1]$.

Each Table reports the resulting training error, test error, and RBF units on three different error targets and increasing the noise level. In each experiment, the training and test sets are both of $p = 100$ random noisy samples. The results show the net behavior

Table 1: $y = \sin(x) + \sin(3x) + \sin(6x)$ results.

Target Error (T)	Noise	RBF (M)	Training Error	Test Error
0.100	0.001	2	0.0114	0.0167
	0.010	2	0.0225	0.0230
	0.100	2	0.0155	0.0153
0.010	0.001	3	0.0096	0.0154
	0.010	3	0.0099	0.0094
0.001	0.001	8	0.0010	0.0016

does not degrade when the noise grows, and the overall results, in terms of basis number, computational cost, and test error, are comparable to the function approximation from no-noisy data literature [11] [16].

Table 2: $y = 0.5\exp(-x) + \sin(6x)$ results.

Target Error (T)	Noise	RBF (M)	Training Error	Test Error
0.100	0.001	1	0.0745	0.0840
	0.010	1	0.0736	0.0838
	0.100	2	0.0229	0.0305
0.010	0.001	3	0.0098	0.0107
	0.010	3	0.0090	0.0093
0.001	0.001	6	0.0009	0.0013

Table 3: $y = x$ results.

Target Error (T)	Noise	RBF (M)	Training Error	Test Error
0.100	0.001	1	0.0025	0.0030
	0.010	1	0.0029	0.0029
	0.100	1	0.0029	0.0026
0.010	0.001	1	0.0025	0.0031
	0.010	1	0.0022	0.0026
0.001	0.001	8	0.0008	0.0010

4.2 Generalization capabilities

Local approximation methods, as the present one, are believed to have poor generalization with respect to global techniques as Polynomials and Multilayer Perceptron (MLP) [17]. Here we compare the model generalization capabilities to such global techniques, by using two benchmarks known in the literature [3].

A training set is considered in the range $[0, 20]$ with 21 noisy points, according to the function $y = sin(x/3)$, and noise parameter (3) $a = 0.25$.

FIGURE 1 HERE

This data set is used to fit polynomial models [3] of order from 2 to 20. As shown in Figure 1, only for the order 10 the approximation is reasonably good, and, as the order increases, significant overfitting is evident. At order 20, the approximating function fits the training data very well, while the interpolation between training points is very poor.

The results of Multilayer Perceptron [3] (single hidden layer, back propagation, 100,000 stochastic training updates, and learning rate schedule with an initial learning rate of 0.5) on the same training set shows a reasonably good approximation with two hidden units (7 weights), but requires 50 hidden units to show a test error of 0.0201.

Our results appear far better with just 3 RBF ($T = 0.01$), as shown in Table 4.

Table 4: $y = \sin(x/3)$ in the range [0, 20] results.

System (T)	Training Error	Test Error
MLP (1 hidden)	0.3730	0.2750
MLP (2 hidden)	0.0358	0.0343
MLP (10 hidden)	0.0204	0.0222
MLP (50 hidden)	0.0204	0.0201
net_2 ($T = 0.100$)	0.0142	0.0099
net_3 ($T = 0.010$)	0.0019	0.0027

The same function is evaluated for $x \in \{0, 1, \ldots, 5\}$ so having 6 training points [3]. The test set is made by evaluating the function without noise at intervals of length 0.1. The MLP error remains quite high and increases with the number of hidden nodes due to overfitting.

Our results are better with just one basis, and obviously no overfitting problem appears. The results are reported in Table 5.

Table 5: $y = \sin(x/3)$ in the range [0, 5] results.

System (T)	Training Error	Test Error
MLP (1 hidden)	0.0876	0.0943
MLP (2 hidden)	0.0347	0.0761
MLP (3 hidden)	$4.08e^{-5}$	0.1030
MLP (4 hidden)	$7.29e^{-5}$	0.1030
net_1 ($T = 0.010$)	0.0012	0.0017

4.3 Time series analysis

Here we test our procedure on the Sunspots time series [18]. This series has been largely analyzed in literature; the first application of autoregressive process modeling was performed in 1927 on the yearly sunspot number as introduced in 1848 by the Swiss astronomer Johann Rudolph Wolf [19] [20] [21].

The used sequence is constituted by 280 yearly values, ranging from 1700 to 1979. The training and test sets are constituted by 200 and 80 randomly selected points, respectively. The results are reported in Table 6.

Table 6: Results on Sunspots time series.

Target Error (T)	RBF (M)	Training Error	Test Error
0.5000	1	0.0789	0.0997
0.2500	1	0.0980	0.0781
0.1250	1	0.0855	0.0830
0.0625	5	0.0571	0.0761
0.0313	12	0.0300	0.0345
0.0156	19	0.0149	0.0205
0.0078	41	0.0075	0.0133
0.0039	55	0.0039	0.0068
0.0020	66	0.0019	0.0054

This series evidences the smoothing (low pass filtering) property of the model.

FIGURE 2 HERE

As we see from Figure 2, at first glance, the results obtained on high target errors $(2^{-4}, 2^{-5})$, corresponding to low cut off frequencies, suggest a main period of about 5/6 decades, and reveal the cyclic pattern of gradual increase and decrease in the average number of sunspots visible on the solar disk. Intermediate results evidences also the low and high amplitude sunspot cycles, known to be related to geomagnetic activity such as aurorae (e.g., 1800-1822, 1940-1960). By lowering the target, the resulting function exhibits the well know 11y period (the number of sunspots reaches a maximum about every 11 years).

4.4 Computational comparison to SVM

The Support Vector Machine (SVM) is a nonlinear generalization of the Generalized Portrait algorithm, developed at At&T Bell Labs [22]. In the general case the SVM solution of a problem is expressed in terms of a small number of support vectors. It is to point out that our approach reverse the SVM's one: SVM tends to *prune* a set of support vectors during the parameter optimization, while our approach *grows* a small subset of such support vectors related to the generalization ability. A more detailed comparison of the two models will be reported in a forthcoming paper [23].

Here we limit ourselves to compare the execution times of our procedure against Smola and Scholkopf's SVM SMO algorithm for regression [24] and Keerthi et al. modifications [25] on a toy problem terminated at the same training set accuracy. As in the Keerthi et al. experimental environment, we use a 200 MHz Pentium machine. The function to be approximated is a cubic polynomial, $0.02x^3 + 0.05x^2 - x$ in the range $[-10; 10]$. A zero mean, unit variance gaussian noise is added to the $p = 100$ training points (details in [25]). As evidenced by Table 7, in this case our procedure works 6 times faster than the mean of the remaining ones.

Table 7: $y = 0.02x^3 + 0.05x^2 - x$ in the range [-10, 10] results.

System (details)	Time (s)
SMO ($C = 1$, 6 non-bound support vectors, 41 bound support vectors)	2.27
SMO-Modification 1 (C=1, 6 non-bound support vectors, 39 bound support vectors)	0.86
SMO-Modification 2 (C=1, 7 non-bound support vectors, 39 bound support vectors)	2.52
net_4 (4 RBF)	0.32

5 Concluding remarks

In this paper an incremental RBF network for function approximation from noisy data has been proposed. Hidden gaussian nodes are iteratively added in the training process. For each new added node, the activation function center and the output connection weight are settled according to an extended chained version of the Nadaraja-Watson estimator [1]. Then the variances of the activation functions are determined by an empirical risk driven rule based on a genetic-like optimization technique [10]. Theoretical upper bounds to the number of hidden units have been derived.

Assuming the data affected by noise, the results show the net behavior does not degrade when the noise grows. The overall results, in terms of basis number, computational cost, and test error, are comparable to the ones of function approximation from no-noisy data methods known in the literature.

As theoretically expected, no overfitting problem appears in the used benchmarks, and the generalization performances in the considered intervals are better than the global techniques.

Furthermore our system appears to be less sensible to the local-minima problem since it avoids the use of gradient descendent in the variance updating.

References

[1] C.M. Bishop, Neural Networks for Pattern Recognition, Clarendon Press, Oxford, 1995; (and the references therein).

[2] D.W. Scott, Multivariate Density Estimation: Theory, Practice, and Visualization, New York, John Wiley, 1992.

[3] S. Lawrence, C. Lee Giles, A. Chung Tsoi, What Size Neural Network Gives Optimal Generalization?, University of Maryland Technical Report UMIACS-TR-96-22 and CS-TR-3617, 1996.

[4] P. Niyogi, F. Girosi, On the Relationship Between Generalization Error, Hypothesis Complexity, and Sample Complexity for Radial Basis Functions, Neural Computation, vol. 8, 819-842, 1996.

[5] R. Chentouf, M. Maignan, C. Jutten, M. Kanevky, Incremental Neural Networks for Function Approximations, AIHENP'96, EPFL-UNIL Lausanne Sept. 2-6 1996.

[6] D. Cubanski, D. Cyganski, Multivariate Classification through Adaptive Delaunay-based $\mathcal{C}^0$ Spline Approximation, IEEE Transactions on PAMI, vol. 17 n.4, 403-417, 1995.

[7] P.W. Goldberg, C.K.I. Williams, C.M. Bishop, Regression with Input-dependent Noise: A Gaussian Process. Treatment, Advances in Neural Information Processing Systems 10, M.I.Jordan, M.J.Kearns, and S.A.Solla Ed.s, Laurence Erlbaum, 1998.

[8] H. Schioler, U. Hartmann, Mapping Neural Network Derived from the Parzen Window Estimator, Neural Networks vol. 5 n.6, 903-909, 1992.

[9] D.F. Specht, Probabilistic Neural Networks, Neural Networks. vol.3 n.1, 109-118,1990.

[10] J.R. Koza, Genetic Programming: On the programming of computers by means of Natural selection, MA, MIT Press, 1992.

[11] A. Esposito, M. Marinaro, D. Oricchio, S. Scarpetta, A New Incremental Strategy for Function Approximation by Radial Basis Function Neural Networks, Neural Nets WIRN 98, M.Marinaro and R.Tagliaferri ed.s, London: Springer Verlag, 1998; (and the references therein).

[12] Z. Ghahramani, M.I. Jordan, Supervised Learning from Incomplete Data via an EM approach, Advances in Neural Information Processing Systems, J.D.Cowan, G.T.Tesauro, and J.Alspector Ed.s, vol. 6, 120-127, San Mateo, Ca, Morgan Kaufmann, 1994.

[13] W. Feller, An Introduction to Probability Theory and its Applications, vol. 1, Third edition, J.Wiley and Sons, New York, 1970.

[14] S. Rampone, Linear Codes Interpolation from Noisy Patterns by Means of a Vector Quantization Process, Computers Math. Applic., vol. 30 n.11, 91-106, 1995.

[15] P. Niyogi, F. Girosi, On the Relationship Between Generalization Error, Hypothesis Complexity, and Sample Complexity for Radial Basis Functions, MIT AI LAB memo n.1467 and CBCL memo n.88, 1994.

[16] V. Vinod, S. Ghose, Growing non-uniform feed-forward networks for continuous mapping, Neurocomputing vol.10 n.1, 55-69, 1996.

[17] S. Lawrence, A. Chung Tsoi, A.D. Back, Function Approximation with Neural Networks and Local Methods: Bias, Variance and Smoothness, ACNN 96, P.Bartlett, A.Burkitt, and R.Williamson ed.s, Australian National University, 1996.

[18] A.S. Weigend, B.A. Huberman, D.E. Rumelhart, Predicting the future: a connectionist approach, International Journal of Neural Systems, vol.1 n.3, 193-209, 1990.

[19] A.J. Izenman, J. R. Wolf, H. A. Wolfer, An historical note on the Zurich sunspot relative numbers, J. R. Statist. Soc. A, vol. 136, 311-318, 1983.

[20] C.W.J. Granger, A statistical model for sunspot activity, Astrophys. J., vol. 126, 152-158, 1957.

[21] M.J. Morris, Forecasting the sunspot cycle, J. R. Statist. Soc. A, vol. 140, 437-447, 1977.

[22] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery, vol. 3 n.2, 1998.

[23] M. Carozza, S. Rampone, Towards an Incremental SVM for Regression, Proc. IJCNN'2000, IEEE Press, 2000.

[24] J.C. Platt, Fast training of support vector machines using sequential minimal optimization, in B. Scholkopf, C. Burges, A. Smola. Advances in Kernel Methods: Support vector Machines, MIT Press, Cambridge, MA, 1998.

[25] S.K, Shevade, S.S. Keerthi, C. Bhattacharyya and K.R.K. Murthy, Improvements to SMO Algorithm for SVM Regression, Technical Report CD-99-16, Control Division, Dept. of Mechanical and Production Engineering, National University of Singapore, Singapore, 1999 (see: $http://guppy.mpe.nus.edu.sg/\tilde{}mpessk$).

Figure 1: Polynomial interpolation (solid line) of the function $y = sin(x/3)$ (dotted line) from noisy data ($\times$) in the range 0 to 20 as the order of the model is increased from 2 to 20. The additive noise is in the range between -0.25 and 0.25. Significant overfitting can be seen for orders 16 and 20 (Lawrence et al. 1996).

Figure 2: Plots of the Sunspots data (dots) and of the resulting functions (solid lines), ranging the target error from $T = 2^{-4}$ to $T = 2^{-9}$.