Function Approximation from Noisy Data by an Incremental RBF Network

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

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

In this paper we propose an incremental RBF network for function approximation from noisy data. 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 [2]. 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. [2].

2 Incremental Estimator

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. Our goal is to find a smooth mapping which captures the underlying systematic aspects of the data without fitting the noise on the data. We consider the following form of radial basis function

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{||\mathbf{x} - \mu_j||^2}{2\sigma_j^2}\right) \tag{4}$$

where μ_j is the center, and σ_j^2 the variance. In the context of kernel regression the following expression for the regression of noisy data is known (Nadaraya-Watson estimator) [1]

$$net_p(\mathbf{x}) = \frac{\sum_{k=1}^{p} t_k \exp(-||\mathbf{x} - \mathbf{x}_k||^2/2\sigma^2)}{\sum_{k=1}^{p} \exp(-||\mathbf{x} - \mathbf{x}_k||^2/2\sigma^2)} \tag{5}$$

It can be extended to the multivariate case by means of the Ghahramani and Jordan result [3]. So we have

$$net_p(\mathbf{x}) = \frac{\sum_{k=1}^{p} t_k \exp(-||\mathbf{x} - \mathbf{x}_k||^2/2\sigma_k^2)}{\sum_{k=1}^{p} \exp(-||\mathbf{x} - \mathbf{x}_k||^2/2\sigma_k^2)} \tag{6}$$

The average error of net_p estimator, the expected risk I_p [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. By the 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 the following

$$M = p^r; \quad r < 1 \tag{7}$$

So we restrict the estimator to

$$net_M(\mathbf{x}) = \frac{\sum_{j=1}^{M} t_{k_j} \exp(-||\mathbf{x} - \mathbf{x}_{k_j}||^2/2\sigma_j^2)}{\sum_{j=1}^{M} \exp(-||\mathbf{x} - \mathbf{x}_{k_j}||^2/2\sigma_j^2)} \tag{8}$$

where $k_j \in \{1, 2, \ldots, p\}$, and $M \leq p^r$. The empirical risk becomes

$$E_M = \frac{1}{p}\sum_{k=1}^{p}(t_k - net_M(\mathbf{x}_k))^2 \tag{9}$$

The estimator (8) can be easily chained to obtain an incremental approach

$$net_M(\mathbf{x}) = \frac{net_{M-1}(\mathbf{x})den_{M-1}(\mathbf{x}) + t_{k_M} \exp(-||\mathbf{x} - \mathbf{x}_{k_M}||^2/2\sigma_M^2)}{den_M(\mathbf{x})} \tag{10}$$

where

$$den_M(\mathbf{x}) = \sum_{j=1}^{M} \exp(-||\mathbf{x} - \mathbf{x}_{k_j}||^2/2\sigma_j^2) \tag{11}$$

In (8) we substitute to t_{k_j} the value

$$w_j = t_{k_j} - net_{j-1}(\mathbf{x}_{k_j}) \tag{12}$$

In this way we take into account the residual approximation error. Since this quantity is affected by the normalization factor den_j, we multiply w_j for this factor, so having

$$net_M(\mathbf{x}) = \frac{\sum_{j=1}^{M} w_j den_j(\mathbf{x}) \exp(-||\mathbf{x} - \mathbf{x}_{k_j}||^2/2\sigma_j^2)}{\sum_{j=1}^{M} \exp(-||\mathbf{x} - \mathbf{x}_{k_j}||^2/2\sigma_j^2)} \tag{13}$$

or, in the chained version

$$net_M(\mathbf{x}) = \frac{net_{M-1}(\mathbf{x})den_{M-1}(\mathbf{x})}{den_M(\mathbf{x})} + w_M \exp(-||\mathbf{x} - \mathbf{x}_{k_M}||^2/2\sigma_M^2) \tag{14}$$

3 Learning Algorithm

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 the point $\mathbf{x}_{k_M}$ such that

$$(t_{k_M} - net_{M-1}(\mathbf{x}_{k_M}))^2 = \max_{k=1,\dots,p}(t_k - net_{M-1}(\mathbf{x}_k))^2 \tag{15}$$

i.e. the training point in which the net has the worst approximation performance. Given (13), we have only to estimate the σ_j^2 values. 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

$$\sigma_{j,new}^2 \leftarrow \sigma_j^2(1+\psi) \tag{16}$$

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 $\sigma_j^2 \leftarrow \sigma_{j,new}^2$. This step is iterated a fixed number of times, and it is motivated by the following

Theorem 1 *Let $I_1, I_2, \dots, I_\gamma$ be disjoint intervals of $\mathcal{R}$ such that $\cup_{k=1}^{\gamma} I_k = I$, and $|I_k| = |I_q|, \quad \forall k, q \in \{1, \dots \gamma\}$. Let I_m be the interval in which $E_M \leq E'_M$, where E_M is the empirical risk of net_M with $\sigma_j^2 \in I_m$, and E'_M the empirical risk of net_M with $\sigma_j^2 \in I_k$, $k \neq m$. Then the expected number of drawings in I necessary to acquire at least a value in I_m is*

$$\gamma \log \gamma \tag{17}$$

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 [5])

The learning algorithm can be summarized as follows:

Learning Algorithm

Step 1: *Input:* T the target training approximation error, tr the number of search iteration of each σ_j^2, r the exponent that maximizes M, α the σ_j^2 mutation parameter, $G = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \dots (\mathbf{x}_p, t_p)\}$ the set of training instances.
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 some functions used by several authors [2] to show the function approximation capabilities of their systems. Table 1 shows the results for $y = \sin(x) + \sin(3x) + \sin(6x)$ in the range $[-.35, .35]$, 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 $[0, 1]$. Both the training and test sets are of $p = 100$ random noisy samples. The results show the net behavior 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. 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.

[2] 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.

[3] 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.

[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] 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.

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

Target	Noise	RBF	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

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

Target	Noise	RBF	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	Noise	RBF	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