Fast and Good Initialization of RBF Networks

D. Bauer[1] and J. Sjöberg[2]

1- Austrian Institute of Technology, Department Mobility, Giefingg.2,
A1210 Vienna, Austria. dietmar.bauer@ait.ac.at

2- Chalmers University, Signals & Systems Department,
Gothenburg, Sweden. jonas.sjoberg@chalmers.se

Abstract. In this paper a new method for fast initialization of radial basis function (RBF) networks is proposed. A grid of possible positions and widths for the basis functions is defined and new nodes to the RBF network are introduced one at the time. The definition of the grid points is done in a specific way which leads to algorithms which are computationally inexpensive due to the fact that intermediate results can be reused and do not need to be re-computed. If the grid is dense one obtains estimators close to estimators resulting from an exhaustive search for the initial parameters which leads to a lower risk to be caught in local minima in the minimization which follows. The usefulness of the approach is demonstrated in a simulation example.

1 Introduction

Typically the parameters in the neural nets are estimated using criterion optimization using a gradient based iterative algorithm. Such a minimization requires 1) the definition of the model structure of the neural network, 2) the supply of an initial model structure and initial parameter values.

For RBF networks there exists a large number of suggestions of algorithms deciding positions and widths of the basis functions so that only the amplitude parameters need to be estimated, which can be done with least squares. Many suggested initialization algorithms, like the one proposed in this paper, are incremental where the size of the network is stepwise increased with new basis functions. Some of the first of such algorithms were suggested in [1] and [2]. These algorithms normally need to be combined with some kind of stopping algorithm, see e.g. [3]. A more recent contribution is [4] where the authors suggest an algorithm deciding both positions of the basis functions and their radii by including knowledge of the output data. Many other ideas exist and an overview can be found in, e.g., [5].

The suggested algorithm adds one important idea compared to the algorithms mentioned above. It starts with a re-sampling of the data so that an equidistant sampled data set is obtained. Alternatively, in some applications the data might be regular from the beginning. The equidistant data is used in the initialization where basis functions are stepwise introduced. The position and the width of each introduced basis function can only take values on a pre specified regular grid. With this setting it is possible to test all combinations given by the grid within a reasonable computation time. With a dense grid one obtains estimators which are close to estimators obtain using an exhaustive search, which means

that local minima become less likely. After the initialization all parameters are adapted using the original data.

The method applies in general multi-input multi-output settings. However, due to space limitations, a simplified version of the algorithm in one dimension is explained, and an example on a two dimensional problem with non-regularly sampled data is given. For a complete description of the algorithm see [6].

The paper is organized as follows: In the next section the considered problem is presented. The proposed algorithm is detailed in Section 3 for the case of uniformly spaced input data. Extensions to non-uniformly spaced data sets as well as for bivariate data sets are discussed in Section 4. The performance is illustrated in an example in Section 5. Finally Section 6 concludes the paper.

2 Problem Statement

The data set is assumed to be generated according to

$$y_i = f(x_i) + e_i \tag{1}$$

where $\{e_i\}_{i=1,\ldots,N}$ are i.i.d. with mean zero and variance σ^2 and $\{x_i\},\ i = 1,\ldots,N$ is the input signal. In the first step it is assumed that (for some x_1)

$$x_{i+1} = x_i + \Delta x,\ i = 1,\ldots,N-1,$$

i.e. that the input signal is equidistantly spaced.

The goal is to use $\{y_i, x_i\},\ i = 1,\ldots,N$ to estimate the unknown function $f : \mathbb{R} \to \mathbb{R}$ using the RBF network of the
following form:

$$f_K(x,\theta) = c + \sum_{k=1}^{K} \alpha_k \varphi(|x-\beta_k|/\gamma_k) = c + \sum_{k=1}^{K} \alpha_k \exp(-|x-\beta_k|^2/\gamma_k^2) \tag{2}$$

where K is the number of basis functions. The function is parameterized with c and $\{\alpha_k, \beta_k, \gamma_k\}_{k=1}^{K}$, where the latter ones are the amplitude, position and width. All parameters are stored in a common parameter vector θ.

The estimation problem consists in finding the parameter value θ such that the squared sum of estimation errors, $\varepsilon_i(\theta) = y_i - f_K(x_i;\theta)$,

$$Q(\theta) = \sum_{i=1}^{N} \varepsilon_i(\theta)^2 = \sum_{i=1}^{N} (y_i - c - \sum_{k=1}^{K} \alpha_k \varphi(|x_i-\beta_k|/\gamma_k))^2 \tag{3}$$

is minimized. For given K the usual approach to minimize $Q(\theta)$ is to first find an initial estimate $\hat{\theta}^{(0)}$, which is the starting point for an iterative search for the minimum. In each iteration, j, one finds a new estimate $\hat{\theta}^{(j+1)}$ with a lower value of $Q(\theta)$. Typically, for this numerical search Levenberg-Marquardt, or some other gradient based method, is used, see, eg, [5].

The minimization of $Q(\theta)$ can be decomposed into two parts using the fact that given $\beta_k, \gamma_k,\ k = 1\ldots,K$ the problem of finding the optimal α_k in (3)

reduces to a linear regression which can be solved explicitly. Define the following symbols $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$, $\tilde{y}_i = y_i - \bar{y}$ and

$$\bar{\varphi}_k \quad = \tfrac{1}{N}\textstyle\sum_{i=1}^{N} \varphi(|x_i - \beta_k|/\gamma_k), \qquad \tilde{\varphi}_{i,k} \quad = \varphi(|x_i - \beta_k|/\gamma_k) - \bar{\varphi}_k$$

as the mean and the deviations from the mean of the output data and the output $\tilde{\varphi}_{i,k}$ of the basis function evaluated at data point i, respectively.

For $K = 1$ and given β_1 and γ_1 it is straight forward to show that

$$\hat{c} = \bar{y} - \bar{\varphi}_1\hat{\alpha}_1, \qquad \hat{\alpha}_1 = \frac{\sum_{i=1}^{N} \tilde{y}_i\tilde{\varphi}_{i,1}}{\sum_{i=1}^{N} \tilde{\varphi}_{i,1}^2}. \tag{4}$$

Combining (4) and (3) one obtains

$$Q(\theta) = \sum_{i=1}^{N} \tilde{y}_i^2 - \frac{(\sum_{i=1}^{N} \tilde{y}_i\tilde{\varphi}_{i,1})^2}{\sum_{i=1}^{N} \tilde{\varphi}_{i,1}^2}. \tag{5}$$

Hence, if we have several possible positions and widths $\{\beta_m, \gamma_j\}, m = 1, \ldots, M$, $j = 1, \ldots, J$ to choose from we should select that pair maximizing the second part of (5), ie,

$$V(m,j) = \frac{(\sum_{i=1}^{N} \tilde{y}_i\tilde{\varphi}_{i,(m,j)})^2}{\sum_{i=1}^{N} \tilde{\varphi}_{i,(m,j)}^2} = \frac{D(m,j)^2}{N(m,j)}. \tag{6}$$

where the index (m,j) of $\tilde{\varphi}_{i,(m,j)}$ indicates the particular choice of β_m and γ_j for basis function k, and $D(m,j)$ and $N(m,j)$ are defined as $D(m,j) = \sum_{i=1}^{N} \tilde{y}_i\tilde{\varphi}_{i,(m,j)}$ and $N(m,j) = \sum_{i=1}^{N} \tilde{\varphi}_{i,(m,j)}^2$.

Maximizing $V(m,j)$ provides the exact solution to a problem approximating the choice of the optimal location $(\beta, \gamma) \in \mathbb{R}^2$ of the basis function. Once the basis functions are evaluated on the grid points, the calculation of $V(m,j)$ mainly involves the formation of inner products which can be done numerically efficient. It is the key insight used in this paper that, thanks to the regular grid, the value of the basis functions needed in $V(m,j)$ and $N(m,j)$ are the same for different m and j. Hence, these values do not need to be re-computed, and this saves an enormous amount of time. The detailed expression of this is shown in [6] but the effect is illustrated in the example in Section 5.

3 The Proposed Algorithm

The grid is defined as

$$\begin{aligned} \gamma_j &= \gamma_1/2^{j-1}, \quad j = 2, \ldots, J, \\ \beta_m &= x_1 + p(m-1)\Delta x, \quad m = 1, \ldots, M \end{aligned} \tag{7}$$

where γ_1 denotes the largest scaling factor for the width and p is a positive integer.

The computations to obtain $D(m,j)$ for all values of m and j can compactly be written using matrix multiplications. Put all the output data in a vector $\tilde{Y} = [\tilde{y}_1,\ \tilde{y}_2,\ \ldots,\ \tilde{y}_N]^T$. Then, for γ_j,

$$A_{M,j}\,\tilde{Y} = [D(1,j)\ D(2,j)\ \ldots D(M,j)]^T \tag{8}$$

where

$$A_{M,j} = \begin{pmatrix} \tilde{\varphi}_{1,(1,j)} & \tilde{\varphi}_{2,(1,j)} & \cdots & \tilde{\varphi}_{N,(1,j)} \\ \tilde{\varphi}_{1-p,(1,j)} & \tilde{\varphi}_{2-p,(1,j)} & \cdots & \tilde{\varphi}_{N-p,(1,j)} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\varphi}_{-p(M-1)+1,(1,j)} & \tilde{\varphi}_{-p(M-1)+2,(1,j)} & \cdots & \tilde{\varphi}_{-p(M-1)+N,(1,j)}. \end{pmatrix} \tag{9}$$

Also, $N(m,j)$, $m = 1,\ldots,M$ become

$$\mathrm{diag}(A_{M,j}A_{M,j}^T) = [N(1,j)\ N(2,j)\ \ldots N(M,j)]^T \tag{10}$$

where the full matrix multiplication can be omitted in the implementation since only the diagonal is used. As mentioned in the previous section, due to the grid and the equidistant data, most of the elements of the $A_{M,j}$ matrices are identical which decreases the number of computations: In particular $\tilde{\varphi}_{i-k,(1,j)} = \tilde{\varphi}_{k-i+2,(1,j)}$ for $i-k<1$. For details see [6].

With the insights from above an RBF network of the form (2) for given K is initialized as follows.

Algorithm 3.1 *1)Define a grid according to (7).*
2) For the first basis function, set $k=1$ and define $\tilde{y}_i^{(1)} = \tilde{y}_i$, $i = 1,\ldots,N$.
3) Calculate $N(m,j)$ using (10).
4) With $\tilde{y}_i^{(k)}$ replacing $\tilde{y}_i$, calculate $D(m,j)$ using (8).
5) Select $\beta_k = \tilde{\beta}_m$ and $\gamma_k = \tilde{\gamma}_j$ for those $\{m,j\}$ maximizing $D(m,j)^2/N(m,j)$.
6) If $k = K$ go to step 7. Else, regress $\tilde{y}_i^{(k)}$ onto $\{\tilde{\varphi}(|x_i - \beta_\kappa|/\gamma_\kappa)\}_{\kappa=1}^k$ and the constant to obtain estimates $\{\hat{\alpha}_\kappa^{(k)}\}_{\kappa=1}^k$ and $\hat{c}^{(k)}$, and the residuals $\tilde{y}_i^{(k+1)}$. Increase k to $k+1$ and go to Step 4.
7) Recalculate the regression of y_i onto $\{\varphi(|x_i - \beta_k|/\gamma_k)\}_{k=1}^K$ and the constant to obtain estimates $\{\hat{\alpha}_k\}_{k=1}^K$ and $\hat{c}$, and the residuals $\varepsilon_i(\hat{\theta})$.

4 Non-equidistant and Bivariate Data

Algorithm 3.1 requires equidistantly sampled data. A straight forward approach to extend the algorithm proposed is to apply some interpolation or smoothing technique and re-sample for the initialization. Many interpolation techniques exists, see, eg, [7, 8].

In the last section the algorithm has been explained for the case of one dimensional input space. The extension to higher dimensional input spaces is straight-forward. The main challenge is to keep track of the matrix elements in the $A_{M,j}$ matrices which are common. See [6] for some more details.

5 Simulation Study

The proposed algorithm is tried on a series of 10 two-dimensional examples. The true functions consist of 50 basis functions of the form

$$f(x) = \sum_{k=1}^{50} \alpha_k \exp(-\|x - \beta_k\|^2/\gamma_k^2) + 14$$

where β_k is chosen uniformly on $[0, 20] \times [-5, 20]$, γ_k is chosen uniformly on $[1, 4]$, and α_k is chosen uniformly on $[0, 1]$.

For each of the true functions, 1000 observations are generated as (1) where x_i is drawn bivariate uniformly from the same domain as β and $\{e_i\}$ is a realization of a Gaussian stochastic variable with variance 0.1.

The models to be fitted in these 10 problems consist of 12 basis function. Hence, the true functions are not in the model set. Since data is randomly distributed, the first step is to use interpolation to resample in order to obtain an equidistant data set which is used in the initialization.

The suggested method was applied and it needed on average 40 CPU seconds to obtain a model in each of the problems. This time is almost equally distributed for the interpolation, the computation of the initial estimate and the iterative minimization. On average 200 iterations were needed in the final iterative minimization. In one of the 10 problems, the chosen upper limit of 1000 iterations was reached.

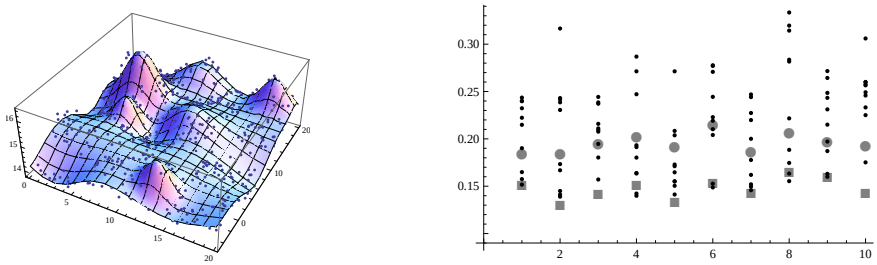
The result from the proposed algorithm is benchmarked against random initialized models where the distributions of the randomly chosen parameters are adapted to the data range. For each problem, 10 different random initializations were used in the iterative fitting. In 17 cases, out of in total 100 minimizations, the minimization was stopped at the upper limit, 1000. On average 400 iterations were used. The ten trials in generally took between 8 and 12 times longer than the proposed algorithm. Hence, the computational cost of the proposed algorithm in this case is approximately the same as for one single realization with the random initialization. This directly relates to the average 400 iterations in the search algorithm for the random initialization compared to approximately 200 for the proposed initialization.

In Figure 1 (a) the result for one of the ten problems is shown. Figure 1 (b) shows an overview of the results from the 10 problems. The random initialization is better in three of the 10 problems, but the difference is small. In most of the problems the suggested algorithm outperforms the random initialization clearly. Considering CPU time used, the random initialization has been given approximately 10 times more time. If both algorithms would have been given an equal amount of time, only one random alternative could be tested.

In [6] a number of additional simulations in the univariate case can be found which show the same behaviour.

6 Conclusions

An algorithm to initialize radial basis function models has been proposed. Compared to a random initialization, the proposed algorithm has a much lower com-



(a) (b)

Fig. 1: (a) Obtained final model using the proposed algorithm for one of the test examples. (b) MSE (vertical axis) for the 10 problems (horizontal axis). The small dots illustrate the result using different random initializations. The grey large dot is the fit for the initial model given the suggested initialization and the grey squared box is the result at the minimum using the suggested initialization.

putational load. The reason for this is that a grid with certain regularity of positions and widths of the basis functions is used which opens up for a re-use of a lot of the needed computations.

7 Acknowledgment

Parts of this work have been financed by project "OD-Metrics" under the "Intelligent Infrastructure" program financed by the Austrian bmfit which is gratefully acknowledged.

References

[1] S. Chen, C.F.N. Cowan, and P.M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *Neural Networks, IEEE Transactions on*, 2(2):302–309, Mar 1991.

[2] C. Jutten and R. Chentouf. A new scheme for incremental learning. In *Neural Processing Letters*, pages 1–4, 1995.

[3] S. Hosseini and C. Jutten. Maximum likelihood neural approximation in presence of additive colored noise. *Neural Networks, IEEE Transactions on*, 13(1):117–131, Jan 2002.

[4] A. Guillen, I. Rojas, J. Gonzalez, H. Pomares, L.J. Herrera, O. Valenzuela, and F. Rojas. Output value-based initialization for radial basis function neural networks. *Neural Processing Letters*, 25(3):209–25, 2007.

[5] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing Company, 866 Third Avenue, NY, second edition, 1998.

[6] D. Bauer and J. Sjöberg. Fast initialization of rbf networks. Technical report, Dept. of Signals and Systems, 2009.

[7] E. T Whittaker adn G. Robinson. *The calculus of observations: An introduction to numerical analysis*, chapter 1, pages 1–34. Dover Publications, New York, 4 edition, 1967.

[8] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Interpolation and Extrapolation*, chapter 3, pages 99–122. Numerical Recipes in FORTRAN: The Art of Scientific Computing. Cambridge University Press, Cambridge, England, 2 edition, 1992.