ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 7-12

A Supervised Radial Basis Function Neural Network

S. Wu and C. Van den Broeck

Limburgs Universitair Centrum, B-3590 Diepenbeek, Belgium

Abstract. We introduce a radial basis function neural network, in which the variances of the radial basis function are corrected by a supervised feedback, with very little cost in training time. A significant decrease of the generalization error is observed in two test problems.

1. Introduction

Radial basis function neural networks (RBF) share with the multilayer perceptrons (MLP) the property of being a universal approximator, but their training time is much shorter [1, 2, 3]. One weak point of RFB however is the absence of supervised feedback in the construction of the radial basis function itself. In the present contribution, we show how a very simple modification of the RFB, that is motivated by theory, allows to incorporate such a supervised correction.

One of the easiest way to introduce RBF is in the traditional framework of MLP. The function implemented by a feed-forward network can be written as

$$y(X) = \sum_{j=1}^{M} w_j \phi_j(X) \tag{1}$$

where $X = \{x_1, \cdots, x_d\}$ is the input vector of dimensionality d, w_j, for $j = 1, \ldots, M$, are the network weights, $\phi_j(X)$ is the jth basis function or the activation of the jth hidden unit, and $y(X)$ is the network output. If the basis functions are chosen to be linear, one recovers the familiar single layer perceptron. To go beyond linear theory, the two most common choices of the basis function are the sigmoid one, leading to the MLP, and the radial basis function, yielding the basic architecture for RBF. Usually, one opts for a multi-Gaussian form:

$$\phi_j(X) = e^{-\frac{1}{2}(X-\mu_j)^T G_j (X-\mu_j)} \tag{2}$$

where μ_j is the center of the jth basis function and G_j is a $d \times d$ positive definite matrix.

Unlike MLP, in which all the layers are trained in a supervised way using backpropagation, the training procedure of RBF is implemented in two stages. The first stage is to determine the basis functions by unsupervised learning,

ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 7-12

using input data points alone. In the second stage, the basis functions are frozen and the second layer is optimized by supervised learning.

The fact that the internal representation of hidden units is the result of an unsupervised process is at the same time the strength and the weakness of the network. On the one hand, it ensures short training times, but on the other hand the internal representation can be quite inappropriate, notably in the presence of input variables which have large variance but have little impact on the corresponding outputs. Here we present a modified RBF technique that optimizes the basis function by a supervised feedback, while keeping intact the most important advantage of RBF, namely its short training time.

2. Optimizing the Basis Function of RBF

2.1. Theoretical motivation

For convenience we introduce a new vector $R = \{r_1, \cdots, r_M\}$, representing the activations r_j of the hidden units j for an input vector$X$:

$$r_j(X) = -\ln \phi_j(X) = \frac{1}{2}(X - \mu_j)^T G_j (X - \mu_j) \tag{3}$$

The cost function for the second layer is

$$E = \frac{1}{N} \sum_{n=1}^{N} \{y(R^n, W) - t^n\}^2 \tag{4}$$

where W is the vector of the network weights. $y(R^n, W)$ represents the network output as a function of the hidden layer state R^n corresponding to the training pattern X^n with target value t^n. In the limit in which the number of examples goes to infinity, the expression (4) becomes

$$E = \int \int \{y(R, W) - t\}^2 p(t|R) p(R) dt dR \tag{5}$$

where $p(t|R)$ is the distribution of the target value conditioned on R and $p(R)$ is the unconditional distribution of R. Denoting $\langle t|R\rangle \equiv \int t p(t|R) dt$ and $\langle t^2|R\rangle \equiv \int t^2 p(t|R) dt$, we can rewrite this expression as follows:

$$\begin{aligned} E &= \int \{y(R, W) - \langle t|R\rangle\}^2 p(R) dR \\ &+ \int \{\langle t^2|R\rangle - \langle t|R\rangle^2\} p(R) dR \end{aligned} \tag{6}$$

The first term in the r.h.s. of Eq. (6) is the cost related to the neural network training. We will call its minimum value the error of training (EOT). The second term is the variance of the target value conditioned on R. We will call it the error of variance (EOV). Different choices of the basis functions will

ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 7-12

normally influence the values of both EOT and EOV. However one can expect that in a real application, with the number of data points much larger than the number of hidden units, EOV will be much more sensitive than EOT to the choice of the radial basis functions. Furthermore, in many cases EOV is found to be much larger than EOT. This serves as a motivation for a supervised optimization procedure of the radial basis functions based on the consideration of EOV alone.

2.2. The method

To stay in line with the main advantage of the RBF, namely the speed of its training, we are looking for a choice of the radial basis functions based on a straightforward optimization procedure. To do so, we need to make a number of simplifying assumptions.

1. We assume all basis functions are well localized on their associated clusters, so that the EOV reduces to the summation of a contribution for each cluster. As a result we can concentrate on minimizing EOV per cluster:

$$EOV = \int \{\langle t^2|r\rangle - \langle t|r\rangle^2\} p(r) dr \tag{7}$$

   in which we have dropped the index referring to the hidden unit under consideration.

2. We consider a common statistical model, assuming that t is equal to the sum of a deterministic function $f(r)$ of r plus a Gaussian noise, with zero mean and standard deviation σ. Going back to the case of finite training examples, the EOV is given by:

$$EOV = \frac{1}{N} \sum_{n=1}^{N} \{t^n - f(r^n)\}^2 \tag{8}$$

3. One can assume that $r < 1$ so that $f(r)$ can be approximated as $f(r) \sim f(0) + f'(0)r + o(r^2)$, where $f'(0)$ is the derative of $f(r)$ at the cluster center.

4. In order to get a simple linear optimization problem, we keep the location of the cluster center μ fixed to the value obtained by unsupervised learning, and only use the matrix G as the parameters for minimizing EOV.

5. In order to guarantee that G retains the crucial property of being positive definite, we consider only the diagonal form which each diagonal element $g_{ii} = e^{\gamma_i}$ positive. We finally get:

$$EOV = \frac{1}{N} \sum_{n=1}^{N} \{\tilde{t^n} - \sum_{i=1}^{d} \tilde{W}_i \tilde{X}_i^n\}^2 \tag{9}$$

ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 7-12

where $\tilde{t^n} = t^n - f(0)$, $\tilde{W}_i = f'(0)e^{\gamma_i}$ and $\tilde{X_i^n} = (X_i^n - \mu_i)^2$. The optimization is thus converted into the training a single layer perceptron, since Eq. (9) can be viewed as the cost function of a single layer perceptron with $\tilde{X_i^n}$ as the input, $\tilde{t^n}$ as the target, and $\tilde{W}_i$ as the weight.

In practice we don't know the values of $f(0)$ and $f'(0)$, but it does not matter. The constant term $f(0)$ can be easily eliminated by including a bias term in the neural network. For $f'(0)$ the situation is more subtle. First, we note that the value of $|f'(0)|$ is irrelevant, since the EOV is invariant under a linear transformation of r, while the width of the cluster is anyways chosen by heuristics. Furthermore the performance of the network is usually not very sensitive to its precise value. To deal with the problem of the unknown sign of $f'(0)$, the single layer network is trained two times with two cost functions, each of which corresponding to one possibility of the sign, namley

$$\frac{1}{N}\sum_{n=1}^{N}\{\tilde{t^n} - \sum_{i=1}^{d} e^{\gamma_i}\tilde{X_i^n}\}^2 \tag{10}$$

and

$$\frac{1}{N}\sum_{n=1}^{N}\{\tilde{t^n} + \sum_{i=1}^{d} e^{\gamma_i}\tilde{X_i^n}\}^2 \tag{11}$$

The proper $\tilde{W}_i$ is chosen from the one having the smaller cost function value.

So far we have not considered the unsupervised learning part. From the several unsupervised techniques proposed in the literature we have chosen the Gaussian mixture model method [2, 4], because it is widely applicable and it can be implemented using a powerful algorithm, the Expectation-Maximization method (EM).

3. Simulation Experiments

To distinguish our method from the usual RBF model, we call it the modified RBF model (MRBF) hereafter. Note that no extra parameter is introduced when going to the MRBF. Since the training error is further minimized, we expect that the generalization error will also decrease. This is confirmed by applying the algorithm to two different test cases. It shows that the assumptions going into the construction of the algorithm and not too restrictive. To compare the performance of MRBF with that of RBF, we have estimated the averaged normalized prediction error (ANPE), which is defined by

$$ANPE = \langle \frac{\sum_{n=1}^{N}[y(X^n, W) - t_n]^2}{\sum_{n=1}^{N}[t^n - \bar{t}]^2} \rangle_{trial} \tag{12}$$

where $\bar{t} = (1/N)\sum_{n=1}^{N} t^n$ is the averaged target value. The notation $\langle\cdot\rangle_{trial}$ refers to the average over different trials of the example set.

ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 7-12

3.1. A radial function regression problem

The first experiment is a radial function regression problem, cf. [5], which is

$$\begin{aligned} f(x_1, x_2) &= 24.234[r^2(0.75 - r^2)] \\ r^2 &= (x_1 - 0.5)^2 + a(x_2 - 0.5)^2. \end{aligned}$$

Ten different training sets, each of which has 300 points, are generated from the uniform distribution $[0, 1]^2$. The test set, of size 10000, is generated from a regularly spaced grid on $[0, 1]^2$. The simulation results with different choices of the parameter a are shown in Table.1.

	a=0.2	a=0.5	a=0.8
RBF	0.071041	0.053879	0.037143
MRBF	0.023826	0.031173	0.035671

Table 1: The simulation results of ANPE for RBF and MRBF both having 7 hidden units.

3.2. A chaotic time series prediction problem

The second experiment is a chaotic time series prediction problem, namely the prediction of the next value of the logistic map $x_{n+1} = 4x_n(1 - x_n)$, as a function of the two previous values. Ten example sets, each of which consists of 500 examples, and the test set of size 10 000, are generated from the stationary attractor distribution. The simulation results are shown in Table.2.

	M=4	M=8	M=12
RBF	0.183074	0.024980	0.012980
MRBF	0.066597	0.010555	0.006707

Table 2: The simulation results of ANPE for RBF and MRBF with varied number of hidden units M.

4. Discussion

We can conclude that MRFB leads to a lower training and generalization error as compared to the RBF. The algorithm is simple and only necessitates the training of single layer networks. The training time is only a few times slower than RBF with the same structure. Simulation shows that it works well in some cases. Even though problems can arise, that can be traced back to the

ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 7-12

failure of some of the assumptions such as the non-overlapping character of the clusters, the algorithm is so fast and simple that there seems to be no reason to just try it out.

5. Acknowledgments

We thank the Program on Inter-University Attraction Poles of the Belgian Government and the Flemish FWO for their support.

References

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

[2] C. M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[3] E. J. Hartman, J. D. Keeler and J. M. Kowalski, Layered neural networks with Gaussian hidden units as universal approximations, Neural Computation, **2**, 210-215, 1990.

[4] A. P. Dempster, N. M. Laird and D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society, B **39**, 1-38, 1977.

[5] J. N. Hwang, S. R. Lay, M. Maechler, D. Martin and J. Schimert, Regression modeling in backpropagation and projection pursuit learning, IEEE Trans. Neural Networks, **5**, 342-353, 1994.