Efficient Algorithm for Training Interpolation RBF Networks with Equally Spaced Nodes

Hoang Xuan Huan, Dang Thi Thu Hien, and Huynh Huu Tue

***Abstract*— This brief paper proposes a new algorithm to train interpolation Gaussian radial basis function (RBF) networks in order to solve the problem of interpolating multivariate functions with equally spaced nodes. Based on an efficient two-phase algorithm recently proposed by the authors, Euclidean norm associated to Gaussian RBF is now replaced by a conveniently chosen Mahalanobis norm, that allows for directly computing the width parameters of Gaussian radial basis functions. The weighting parameters are then determined by a simple iterative method. The original two-phase algorithm becomes a one-phase one. Simulation results show that the generality of networks trained by this new algorithm is sensibly improved and the running time significantly reduced, especially when the number of nodes is large.**

***Index Terms*— Contraction transformation, equally spaced nodes, fixed-point, output weights, radial basis functions, width parameters.**

I. Introduction

Interpolation of functions is a very important problem in numerical analysis with a large number of applications [1]–[5]. The case of 1-D had been studied and solved by Lagrange, using polynomial as forms of interpolating functions. However, the multivariable problems have attracted interest of researchers only in the second half of the 20th century, when pattern recognition, image processing, computer graphics and other technical problems dealing with partial different equations were born. Several techniques were proposed to solve the approximation and interpolation problems such as multiple-layered perceptron, radial basis function (RBF) neural networks, k-nearest neighbor (K-NN) and locally weighted linear regression [6]. Among these methods, RBF networks are commonly used for interpolating multivariable functions. The RBF approach was first proposed by Powell as an efficient technique to solve the multivariable function interpolation [7]. Broomhead and Lowe had adapted this method to build and train neural networks [8].

In a multivariate interpolation RBF network of a function f, the interpolation function is of the form: $\varphi(x) = \sum_{k=1}^{M} w_k h(||x - v^k||), \sigma_k) + w_0$ with interpolation conditions $\varphi(x^k) = y^k$, for all $k = 1, \ldots, N$, where $\{x^k\}_{k=1}^{N}$ is a set of n-dimensional vectors (called as interpolation nodes) and $y^k = f(x^k)$ is a measured value of function f at respective interpolation node (in approximation networks, these equations are approximated), real functions $h(||x - v^k||, \sigma_k)$ are called as RBFs with center $v^k (M \leq N)$, where w_k and σ_k are unknown parameters that we have to determine. The general approximation (known as generality property) was discussed in [9] and [10].

The most common kind of RBFs [2], [11], [12] is of Gaussian form $h(||x - v||, \sigma) = e^{-||x-v||^2/\sigma^2}$, where v and σ are, respectively, the center and the width parameters of the RBFs.

For noiseless data with a small number of interpolation nodes, they are employed as centers of RBFs such that the number of nodes is equal to the number of RBFs to be used ($M = N$). Given preset widths, the output weights satisfying the interpolation conditions are unique and the corresponding RBF networks are called as interpolation ones.

For the case of large number of interpolation nodes, the Gauss elimination method or other direct methods using matrix multiplication have the complexity of $O(N^3)$, furthermore, accumulated errors quickly increase. On the other hand, optimization techniques used to minimize the sum of squared errors converge too slowly and give large final errors. Therefore, one often chooses M smaller than N [12]. To choose the number of neurons M and determine the centers v^k of the corresponding RBFs are still open research problems [13], [14]. To avoid these obstacles, the authors recently proposed an efficient algorithm to train interpolation RBF networks with very large number of interpolation nodes with high precision and short training time [15], [16].

In practice, like in computer graphics as well as in technical problems involving partial differential equations, for the interpolation problem, one often has to deal with the case of equally spaced nodes [1], [3], [5]. This brief paper is based on the training algorithm proposed by Hoang, Dang, and Huynh [15], referred from now on as HDH algorithm. This HDH training algorithm has two phases: 1) in the first, it iteratively computes the RBF width parameters, and 2) in the second, the weights of the output layer are determined by the simple iterative method.

In the case of equally spaced data, their coordinates can be expressed as $x^{i1,i2,\ldots in} = (x_1^{i1}, \ldots, x_n^{in})$, where $x_k^{ik} = x_k^0 + ik * h_k$, h_k being the constant steps in the k^{th} dimension and ik varies from 1 to N_k. When the Euclidian norm $||x|| = \sqrt{x^T x}$ associated to RBF is replaced by a Mahalanobis norm $||x||_A = \sqrt{x^T A x}$ with A conveniently chosen as specified in Section III-A, by exploiting the characteristic of uniformly spaced data, the width parameters can now be predetermined so that the originally proposed technique becomes one-phase algorithm.

As the training time for the original algorithm is mainly spent in the first, the obtained one-phase algorithm is therefore very efficient. Furthermore, the generality is sensitively improved.

The rest of this brief paper is organized as follows. In Section II, interpolation RBF networks and the HDH algorithm [15] are briefly introduced. Section III is dedicated to the new algorithm for the interpolation problem with equally

Manuscript received February 11, 2010; revised February 19, 2011; accepted February 19, 2011. Date of publication May 13, 2011; date of current version June 2, 2011. This work was supported in part by the National Foundation for Science and Technology Development.

H. X. Huan is with the College of Technology, Vietnam National University, Hanoi, Vietnam (e-mail: huanhx@vun.edu.vn).

D. T. T. Hien is with the University of Transport and Communications, Hanoi, Vietnam (e-mail: dthien@uct.edu.vn).

H. H. Tue is with the Bac-Ha International University, Hanoi, Vietnam (e-mail: huynhhuutue@bhiu.edu.vn).

Digital Object Identifier 10.1109/TNN.2011.2120619

spaced nodes. Simulation results are shown in Section IV. Some conclusions are presented in the final section.

II. Interpolation RBF Networks and the HDH Algorithm

This section briefly presents the HDH algorithm and its related concepts (see [15] for more details).

A. Interpolation RBF Network

Multivariate Interpolation Problem: Consider the problem of interpolation with noiseless data. Let f be a multivariate function $f : D(\subset R^n) \rightarrow R^m$ and the sample set$\{x^k, y^k\}_{k=1}^N$; $\{x^k\}_{k=1}^N \subset D$ such that $f(x^k) = y^k; k = 1, \ldots, N$. Let φ be a function of a known form satisfying

$$\varphi(x^i) = y^i \qquad \forall i = 1, \ldots, N. \tag{1}$$

The points x^k and the function φ are, respectively, called as interpolation nodes and the interpolation function of f. φ is used to approximate f on the domain D. Powell proposed to exploit RBFs for the interpolation problem [7]. In the following section, we will sketch the Powell technique using Gaussian radial function (for further details see [12], [17]).

Interpolation Technique Based on RBFs: Without loss of generality, it is assumed that m is equal to 1. The interpolation function φ has the following form:

$$\varphi(x) = \sum_{k=1}^{N} w_k \varphi_k(x) + w_0 \tag{2}$$

where

$$\varphi_k(x) = e^{-\|x-x^k\|^2/\sigma_k^2}; \quad \forall k = 1, \ldots, N \tag{3}$$

where $\|u\|$ is a norm of u (in this brief paper, it is the Euclidean norm) and x^k is called as center of RBF φ_k, w_k and σ_k, are parameters such that φ satisfying interpolation conditions (1)

$$\varphi(x^i) = \sum_{k=1}^{N} w_k \varphi_k(x^i) + w_0 = y^k; \quad \forall i = 1, \ldots, N. \tag{4}$$

For each k, parameter σ_k (called width parameter of RBF) is used to control the width of the Gaussian basis function φ_k, when $\|x - x^k\| > 3\sigma_k$, then $\varphi_k(x)$ is almost negligible. Consider the $N \times N$ matrix Φ

$$\Phi = \left(\varphi_{k,i}\right)_{N\times N}$$

where

$$\varphi_{k,i} = \varphi_k(x^i) = e^{\frac{-||x^i - x^k||^2}{\sigma_k^2}} \tag{5}$$

with the chosen parameters σ_k. If all nodes x^k are pairwise different, then the matrix Φ is positive-definite [18]. Therefore, with given w_0, the solution $w_1, \ldots, w_N$ of (2) always exists and is unique.

In the case where the number of RBFs is less than N, their center might not be an interpolation node and (2) may not have any solution, the problem is then finding the best approximation of f using any optimum criteria. Usually, parameters w_k and σ_k are determined by the least mean square method [12], which does not correspond to our situation. Furthermore, determining the optimum center is still an open research problem as mentioned above.

Interpolation RBF Network Architecture: An interpolation RBF network is a 3-layer feedforward neural network which is used to interpolate a multivariable real function $f : D(\subset R^n) \rightarrow R^m$. It is composed of n nodes of the input layer, represented by the input vector $x \in R^n$; there are N neurons in the hidden layer, of which the kth neuron center is the interpolation node x^k and; its kth output is $\varphi_k(x)$; finally the output layer contains m neurons which determine interpolated values of $f(x)$. Given the fact that in the HDH algorithm, each neuron of the output layer is trained independently when $m > 1$, we can then assume $m = 1$ without loss of generality. There are different training methods for interpolation RBF networks, but as shown in [15], the HDH algorithm offers the best-known performance (with regard to training time, training error and generality) and is briefly presented in the following section.

B. Review of the HDH Algorithm

In the first phase of the two-phase HDH algorithm, radial parameters σ_k are determined by balancing between the error and the convergence rate. In the second phase, weight parameters w_k are obtained by finding the fixed point of a given contraction transformation accordingly selected. Let us denote by Section I the $N \times N$ identity matrix, $W = [w_1, \ldots, w_N]^T$, $Z = [z_1, \ldots, z_N]^T$, respectively, two vectors in N-dimensions space R^N, where

$$z_k = y_k - w_0 \quad \forall k \leq N \tag{6}$$

and let

$$\Psi = I - \Phi = \left[\psi_{k,j}\right]_{N\times N} \tag{7}$$

where Φ is given in (5), then

$$\psi_{k,j} = \begin{cases} 0; & \text{if: } k = j \\ -e^{-||x^j - x^k||^2/\sigma_k^2}; & \text{if: } k \neq j. \end{cases} \tag{8}$$

Equation (2) can now be rewritten as

$$W = \Psi W + Z \tag{9}$$

w_0 in (3) is chosen as the average of y^k values. Now, for each $k \leq N$, let us define $q_k > q$ with

$$q_k = \sum_{j=1}^{N} \left|\psi_{k,j}\right|.$$

Given an error ε and two positive constants $q < 1$ and $\alpha < 1$, the algorithm computes parameters σ_k and W^*, solution of (9). In the first phase, for each $k \leq$ N, σ_k is determined such that $q_k < q$, while replacing σ_k by σ_k/α, we have $q_k > q$. With these values, the norm $\|\psi\|_*$ of matrix Ψ given by (6) is less than q, such that an approximate solution W^* of (9) will be found in the next phase by a simple iterative method.

The norm of N-dimension vector u is given by

$$\|u\|_* = \max\left\{\frac{|u_j|}{j} \leq N\right\}. \tag{10}$$

The ending condition is chosen by the following:

$$\frac{q}{1-q}\left\|W^1 - W^0\right\|_* \le \varepsilon. \tag{11}$$

The above algorithm always ends after a finite number of steps, of which the solution satisfies the following inequality:

$$\left\|W^1 - W^*\right\|_* \le \varepsilon. \tag{12}$$

Its complexity is $O\left((T+c)\,nN^2\right)$, where c and T are given constants [15]. The training time of phase 1 only depends on the number of interpolation nodes and that of phase 2 on $\|Z\|_* = \max|Z_i = y^i - 1/n\sum_{i=1}^{N} y_i|$ but not on the variation of the interpolated function f.

III. Interpolation Problem with Equally Spaced Nodes and New Training Algorithm

A nice feature of the HDH algorithm is that it computes the RBFs widths in such a way that the matrix Φ to be used in the second phase is of diagonal dominance, which is the desired property that allows for a very efficient determination of the output weights by the simple iterative method. Due to this efficiency, the HDH algorithm can handle interpolation networks with a very large number of nodes.

Experimental results show that the first phase of the HDH algorithm consumes a high percentage of the total running time (see Section IV-A below). The objective of this brief paper is to precompute these RBF widths for the case of equally spaced nodes so that the HDH algorithm will become one-phase algorithm.

A. Problem with Equally Spaced Nodes

From now on, we consider a problem that the interpolation nodes are equally spaced. In these cases, we can express each interpolation node by a multi-index node as

$$x^{i1,i2,\ldots in} = (x_1^{i1},\ldots,x_n^{in});\quad x_k^{ik} = x_k^0 + ik * h_k;\quad k = 1,\ldots,n \tag{13}$$

where $h_k (k = 1,\ldots,n)$ is the changing step of parameter x_k, n is the number of dimensions, ik are taken in range between 1 and N_k (N_k are scale numbers of the kth dimension).

In (3), the values of each radial function are the same at points which are equidistant to the center, and its level surfaces are spherical. This choice does not conveniently suit situations where interpolation steps $\{h_k; k = 1,\ldots,n\}$ strongly deviate from each other. In these cases, instead of Euclidean norm, we consider a Mahalanobis norm defined by $\|x\|_A = \sqrt{x^T A x}$, where A is a diagonal matrix

$$A = \begin{pmatrix} \frac{1}{a_1^2} & 0 & \ldots & 0 \\ 0 & \frac{1}{a_2^2} & \ldots & 0 \\ \ldots & \ldots & . & .... \\ 0 & 0 & \ldots & \frac{1}{a_n^2} \end{pmatrix}.$$

a_k are fixed positive parameters which will be conveniently chosen later on, in order to allow for constructing our proposed efficient algorithm. Equations (1) and (3) are then rewritten as follows:

$$\varphi(x) = \sum_{i1,\ldots in=1}^{N_1,\ldots N_n} w_{i1,\ldots in}\varphi_{i1,\ldots in}(x) + w_0 \tag{14}$$

where

$$\varphi_{i1,\ldots in}(x) = e^{-\left\|x - x^{i1,\ldots in}\right\|_A^2/\sigma_{i1,\ldots in}^2}. \tag{15}$$

The $N \times N$ matrix Φ expressed in (5) is rewritten as: $\Phi = \left(\varphi_{i1,\ldots,in}^{j1,\ldots,jn}\right)_{N\times N}$ $(N = N_1 \ldots N_n)$, where

$$\varphi_{i1,\ldots,in}^{j1,\ldots,jn} = \varphi_{i1,\ldots in}(x^{j1,\ldots,jn}) = e^{-\left\|x^{j1,\ldots,jn} - x^{i1,\ldots in}\right\|_A^2/\sigma_{i1,\ldots in}^2}. \tag{16}$$

The entries of the Matrix $\Psi = I - \Phi$ are defined as follows:

$$\Psi_{i1,\ldots,in}^{j1,\ldots,jn} = \begin{cases} 0; & \text{if : } j1,\ldots,jn = i1,\ldots,1n \\ -e^{-\left\|x^{j1,\ldots,jn} - x^{i1,\ldots,in}\right\|_A^2/\sigma_{i1,\ldots,in}^2}. \end{cases} \tag{17}$$

Radii $\sigma_{i1,\ldots,in}$ are determined so that matrix Ψ is a contraction transformation, in order to ensure that the phase two of the HDH algorithm can be correctly applied.

It means that, given a constant $q \in (0, 1)$, choose $\sigma_{i1,\ldots,in}$ such that

$$q_{i1,\ldots,in} = \sum_{j1,\ldots,jn}\left|\psi_{i1,\ldots,in}^{j1,\ldots,jn}\right| \le q < 1. \tag{18}$$

Taking (14) into account, it implies that

$$\Psi_{i1,\ldots,in}^{j1,\ldots,jn} = \begin{cases} 0, \text{where : } j1,\ldots,jn = i1,\ldots,in \\ -e^{-\sum_{p=1}^{n}(jp-ip)^2\frac{h_p^2}{a_p^2}/\sigma_{i1,\ldots,in}^2}. \end{cases} \tag{19}$$

Then, $q_{i1,\ldots,in}$ can be re-written as follows:

$$\begin{aligned} q_{i1,\ldots,in} &= \sum_{j1,\ldots,jn\neq i1,\ldots,in} e^{-\sum_{p=1}^{n}(jp-ip)^2\frac{h_p^2}{a_p^2}/\sigma_{i1,\ldots,in}^2} \\ &= \prod_{p=1}^{n}\sum_{jp=1}^{N_p} e^{-\frac{h_p^2}{\sigma_{i1,\ldots,in}^2 a_p^2}(jp-ip)^2} - 1. \end{aligned} \tag{20}$$

Finally, if we set $a_p = h_p$, then

$$q_{i1,\ldots,in} = \prod_{p=1}^{n}\sum_{jp=1}^{N_p} e^{-\frac{(jp-ip)^2}{\sigma_{i1,\ldots,in}^2}} - 1. \tag{21}$$

The following theorem is the basis of the new algorithm.

Theorem 6: For all $q \in (0, 1)$, if all $\sigma_{i1,\ldots,in}$ are chosen such that

$$\sigma_{i1,\ldots,in} \le \left[\ln\left(\frac{6}{\sqrt[n]{1+q} - 1}\right)\right]^{-\frac{1}{2}} \text{ then } q_{i1,\ldots,in} < q < 1. \tag{22}$$

Proof: In fact, with $j_p, i_p \in \{1,\ldots,N_p\}$, the right-hand side member (RHS) of (21) can be bounded by

$$q_{i1,\ldots,in} < \left(1 + 2\sum_{k=1}^{\infty} e^{-\frac{k^2}{\sigma_{i1,\ldots,in}^2}}\right)^n - 1. \tag{23}$$

Procedure QHDH Algorithm
Begin
Setting $\sigma_{i1,..,\text{in}} = \sigma$// chosen among Eqs (28), (29);
Find W* by simple iterative method;// The same phase 2 of HDH Algorithm described in section II;
End

Fig. 1. Procedure of training RBF network with equally spaced nodes.

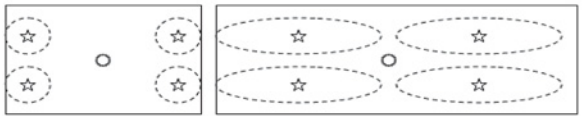

Fig. 2. Influence of RBFS with star as center. (a) Euclidean norm. (b) Mahalanobis norm.

A sufficient condition to insure (18) is

$$\left(1 + 2\sum_{k=1}^{\infty} e^{-\frac{k^2}{\sigma_{i1,\ldots,in}^2}}\right)^n - 1 \le q < 1. \tag{24}$$

That is equivalent to

$$\sum_{k=1}^{\infty} e^{-\frac{k^2}{\sigma_{i1,\ldots,in}^2}} \le \frac{\sqrt[n]{1+q} - 1}{2}. \tag{25}$$

To simplify the notation, let us denote $\sigma_{i1,\ldots,in}$ by σ. Given the condition that $q < 1$ and $n \in N$, the RHS of (25) is all the time upper-bounded by 1/2. Let us consider just the first term of the left-hand side member (LHS) of (25). We then have

$$e^{-\frac{1}{\sigma^2}} < \frac{1}{2}, \text{ that gives } \sigma < \frac{1}{\sqrt{\ln 2}}. \tag{26}$$

On the other hand, the LHS of (25) can be bounded as follows:

$$\sum_{k=1}^{\infty} e^{-\frac{k^2}{\sigma^2}} = e^{-\frac{1}{\sigma^2}} \sum_{k=1}^{\infty} e^{-\frac{k^2-1}{\sigma^2}} = e^{-\frac{1}{\sigma^2}} \left(1 + \sum_{k=2}^{\infty} e^{-\frac{k^2-1}{\sigma^2}}\right)$$
$$= e^{-\frac{1}{\sigma^2}} \left(1 + \sum_{k=1}^{\infty} e^{-\frac{(k+1)^2-1}{\sigma^2}}\right) < e^{-\frac{1}{\sigma^2}} \left(1 + \sum_{k=1}^{\infty} e^{-\frac{k^2}{\sigma^2}}\right)$$
$$< e^{-\frac{1}{\sigma^2}} \left(1 + \int_0^{\infty} e^{-\frac{t^2}{\sigma^2}} dt\right) = e^{-\frac{1}{\sigma^2}} \left(1 + \sigma\sqrt{\frac{\pi}{2}}\right).$$

Using (26), we obtain

$$\sum_{k=1}^{\infty} e^{-\frac{k^2}{\sigma^2}} < e^{-\frac{1}{\sigma^2}} \left(1 + \frac{\sqrt{\pi/\log(1)}}{2}\right) \approx 2.085 e^{-\frac{1}{\sigma^2}} < 3e^{-\frac{1}{\sigma^2}}. \tag{27}$$

Equation (27) shows that (25) is satisfied when $3e^{-1/\sigma_{i1,\ldots,in}^2} \le (\sqrt[n]{1+q} - 1)/2$ or equivalently, (25) is satisfied when

$$\sigma_{i1,\ldots,in} \le \sqrt{\frac{1}{\ln\left(\frac{6}{\sqrt[n]{1+q}-1}\right)}}.$$

■

TABLE I
COMPARISON OF TRAINING TIME OF NETWORKS

Number of Nodes	QHDH	HDH
1071 ($N_1 = 51$, $h_1 = 0.2$, $N_2 = 21$, $h_2 = 1$)	10 in	32 in
5271 ($N_1 = 251$, $h_1 = 0.04$, $N_2 = 21$, $h_2 = 1$)	275 in	1315 in
10251($N_1 = 201$, $h_1 = 0.05$, $N_2 = 51$, $h_2 = 0.4$)	765 in	> 2h

TABLE II
COMPARISON OF TRAINING ERROR AND TRAINING TIME OF NETWORKS

Test Function	QHDH, $q = 0.9$ $\sigma = 0.5568$ Training Time = 18 in	HDH, $q = 0.9$ $\alpha = 0.9$ Training Time = 35 in	QTH $\sigma =$ 0.07252 SSE = 0.0013856 Training Time = 48 in	QTL $\sigma =$ 0.07215 SSE = 0.0016743 Training Time = 46 in
	Average Error	Average Error	Average Error	Average Error
Y_2	3.85E-09	7.84E-08	6.95E-05	7.16E-05

Remark 8: For practical purpose, it is more convenient to choose all $\sigma_{i1,\ldots,in}$ identical and equal to σ with two different possibilities.

1)
$$\sigma_{i1,\ldots,in} = \sigma_0 = \left[\ln\left(\frac{6}{\sqrt[n]{1+q} - 1}\right)\right]^{-\frac{1}{2}}. \tag{28}$$

2) With given n, q and $\gamma > 1$ choosing $\sigma = \sigma_0 \gamma^m$, where m is the largest integer such that

$$\sum_{k=1}^{N} e^{-\frac{k^2}{\sigma^2}} \le \frac{\sqrt[n]{1+q} - 1}{2}. \tag{29}$$

In this case, using the same approach as in (21)–(24), it is easy to show that (22) is satisfied. The complexity is of the order $O(N)$, which is almost negligible, compared to other orders.

B. New Training Algorithm QHDH

Now, with given positive constant $q < 1$, parameters $\sigma_{i1,\ldots,in}$ are preset by one of the three possible choices defined in (28) and (29). Based on the above theorem, the output layer weights can be determined by using the second phase of the HDH algorithm. Thus, the new algorithm is named as QHDH (abbreviation of Quick HDH) which is specified in Fig. 1.

C. Algorithm Complexity

The complexity of this algorithm is due to two actions: computing Φ and computing the output weights. The complexity associated to the computation of Φ is $O(nN^2)$. To compute the output weights warranting a given error ε, we need at most T iterations with $T = (\ln(\varepsilon(1-q)/\|Z\|_*)/\ln q)$ [15], each iteration has the complexity $O(N^2)$ so that the total complexity of the algorithm is $O((n+T)N^2)$.

D. Discussion on the Algorithm

One good feature of Gauss RBF neural networks is their local character, meaning that only data in their neighborhood can influence their behavior (see [6]). For this reason it is suggested to choose small width for RBFs [19, p.289]. However, for points far away from the center, values of RBFs are negilible so that the interpolation errors at these points become inacceptable. This behavior is illustrated in Fig. 2(a). The following experiments show that the width chosen by our method gives better performance when compared to Haykin or Looney choices [12], [17]. Moreover, with the Mahalanobis norm $\|x\|_A = \sqrt{x^T A x}$ defined in Section III-A, with any h_k, points far from centers are still strongly influenced by RBFs [see Fig. 2(b)]. Thank to this property, the generality of the networks when using the Mahalanobis norm offers better performance compared to Euclidean norm. These features are in fact observed in following simulation results.

IV. Simulation Study

In [15], the complexity and the convergence of the HDH algorithm have been analyzed, as the QHDH algorithm is a sibling of the HDH one, it still has all advantages of the HDH one. The goal of the following simulations is to compare the training time, training error and the generality of the networks trained by the QHDH algorithm with respect to those trained by the HDH algorithm and some one-phase gradient algorithms.

In the scenarios of simulation, we are interested in comparing the running time, the training error and the generality of QHDH, HDH [15], LMS/SSE [12, pp.98–100] with two different choices of width parameters $\sigma = 1/(2N)^{1/n}$ [12, p.99] and ($\sigma = D\max/\sqrt{2N}$) [17, p.299], where Dmax is the maximum distance between two interpolation nodes. In the following, the last two algorithms are denoted as QTL and QTH. To avoid repetition, we only present numerical results for the case where σ is defined by (28).

Given the fact that the QHDH running time linearly depends on the data space dimensions, for the simulation convenience, we just need low dimension spaces to illustrate its performance. On the other hand, the performance of QHDH and HDH are perfectly determined so that for the convenience of comparison, to avoid the burden of presentation, we just look at 10 randomly chosen points farthest from centers in the interpolation domain for the interpolation error.

Noiseless data are generated with four different functions.

The first function with two-variables $y_1 = 1 + (2x_1 + \cos(3x_1))/(x_1x_2 + 1)$ where $x_1 \in [0, 3.5]$ and $x_2 \in [0, 7]$ provide a case where different numbers of interpolation nodes are used to compare the training time with respect to the HDH algorithm.

The last three ones of three-variables

$$y_2 = x_1 + \cos(x_2 + 1) + \sin(x_3 + 1) + 2, \quad x_1 \in [0, 1], x_2 \in [0, 2], x_3 \in [0, 3]$$

$$y_3 = x_1^2 x_2 + \sin(x_2 + x_3 + 1) + 1, \quad x_1 \in [0, 1], x_2 \in [0, 2], x_3 \in [0, 3]$$

$$y_4 = x_1^2 x_2 + x_3 + \sin(x_2 + x_3 + 1) + 1, \quad x_1 \in [1, 2], x_2 \in [0, 2], x_3 \in [0, 3]$$

give more complex cases to be studied, in order to illustrate the performance of the QHDH algorithm. We are going to compare the training error and the network generality of QHDH with respect to the ones of HDH algorithm, the QTL algorithm and the QTH algorithm. Furthermore, comparing the algorithm generality for different choices of σ will show the best choice for the training process.

The tests are run on a computer with the following configuration: on Intel Pentium IV Processor, 3.0 GHz, and 512 MB DDR RAM. The ending condition is the error $\varepsilon = 10^{-6}$.

A. Comparison of Training Time

The simulation results are presented in Table I for the two-variable function in order to compare training time of networks trained by the QHDH algorithm and by the HDH algorithm.

The training time of networks trained by the QHDH algorithm is reduced significantly in comparison to those of networks trained by the HDH algorithm.

B. Comparison of Training Error

The experiment results are presented in Table II for the three-variable function y_2 with 1331 nodes, where $N_1 = 11$, $h_1 = 0.1$; $N_2 = 11$; $h_2 = 0.2$; $N_3 = 11$; $h_3 = 0.3$. After the training is completed, the average training error is computed over 100 randomly chosen interpolation nodes.

The experiment results have shown that the training error and the training time of the QHDH algorithm are the best among these four algorithms.

C. Comparison of Generality

The network generality trained by different algorithms is analyzed for two cases by computing errors: 1) at 10 points which are farthest ones from interpolation nodes, and 2) at 100 random points using cross validation method [20].

1) Comparison at the Farthest Points: The experiment results are presented in Table III for the three-variable function y_2 with 1331 nodes, where $N_1 = 11$, $h_1 = 0.1$, $N_2 = 11$, $h_2 = 0.2$, $N_3 = 11$, $h_3 = 0.3$. After the training is finished, we take 10 random points which are the farthest ones from interpolation nodes in the interpolated domain.

The experiment results have shown that the QHDH algorithm has a runtime much shorter and its generality much better than those trained by other algorithms.

2) Comparison by Cross-Validation: In this section, we are going to compare the average absolute error computed over 100 randomly chosen nodes, namely the cross-validation error, for the three training methods: QHDH, QTL, and QTH for all four functions with $\sigma = \gamma^m \sigma_0$, where $\gamma = 1.1$ and m ≥ -1.

Table IV shows the cross-validation error corresponding to the two-variable function y_1 with 1296 interpolation nodes and with $N_1 = N_2 = 36$, $h_1 = 0.1$, $h_2 = 0.2$. Table V shows results for three other functions y_2, y_3 and y_4 with 1331 interpolation nodes with $N_1 = N_2 = N_3 = 11$, $h_1 = 0.1$, $h_2 = 0.2$, $h_3 = 0.3$.

TABLE III

COMPARISON OF GENERALITY OF NETWORKS AT 10 FARTHEST POINTS

Co-ordinate of Checked Point			Original Function Value	QHDH: $q = 0.9$, $\sigma = 0.5568$ Training Time = 18 in		HDH: $q = 0.9$, $\alpha = 0.9$ Training Time = 35 in		QTH: $\sigma = 0.07252$, SSE = 0.0013856 Training Time = 48 in		QTL: $\sigma = 0.07215$ SSE = 0.0016743 Training Time = 46 in	
$x1$	$x2$	$x3$		*Interpolation Value*	*Error*	*Interpolation Value*	*Error*	*Interpolation Value*	*Error*	*Interpolation Value*	*Error*
0.25	1.1	0.2	2.67719298	2.61873	0.058462981	2.587851	0.0893423	0.108851	2.568342	0.298539	2.378654
0.45	0.9	0.4	3.11216016	2.97086	0.141300163	3.325614	0.2134542	0.237326	2.874834	0.424728	2.687432
0.35	1.3	0.5	2.68121897	2.61761	0.063608965	2.602756	0.0784632	0.493676	2.187543	0.462473	2.218746
0.15	0.9	1	2.73600786	2.65512	0.08088786	2.82574	0.0897323	0.640525	2.095483	0.53864	2.197368
0.45	1.1	1.3	2.69085911	2.63282	0.058039108	2.612116	0.0787432	0.561016	2.129844	0.360984	2.329875
0.25	1.3	1.6	2.09922535	2.28295	0.183724649	2.249048	0.1498224	0.223732	1.875493	0.204642	1.894583
0.35	0.7	2.1	2.26273617	2.34536	0.082623832	2.361169	0.0984326	0.07928	2.183456	0.049862	2.212875
0.45	0.9	1.9	2.36595976	2.44444	0.078480238	2.463603	0.0976436	0.217583	2.148377	0.078399	2.287561
0.65	0.7	1.7	2.94853539	2.7636	0.184935386	3.146968	0.1984324	0.60088	2.347655	0.420098	2.528438
0.75	0.9	1.9	2.66595976	2.62388	0.042079762	2.578279	0.0876803	0.833472	1.832488	0.183307	2.482652
Average error					0.097414294		0.1181747		2.224351		2.321818

TABLE IV

COMPARISON OF GENERALITY OF NETWORKS AT 100 RANDOM POINTS FOR Y_1

Test Function	QHDH, $q = 0.9$					QTH: $\sigma = 0.1537218$ SSE = 0.001552	QTL: $\sigma = 0.0196419$ SSE = 0.00174
	$\sigma = 0.546818$	$\sigma = 0.6015$	$\sigma = 0.66165$	$\sigma = 0.7218$	$\sigma = 0.79398$		
	Average Error	*Average Error*	*Average Error*	*Average Error*	*Average Error*	*Average Error*	*Average Error*
Y_1	0.0932212	0.0512532	0.025595	0.0152543	*diverging*	1. 583921	5. 548693

TABLE V

COMPARISON OF GENERALITY OF NETWORKS AT 100 RANDOM POINTS FOR Y_2, Y_3 AND Y_4

Test Function	QHDH, $q = 0.9$					QTH: $\sigma = 0.07252$ SSE = 0.0013856	QTL: $\sigma = 0.07215459$ SSE = 0.0016743
	$\sigma = 0.50618$	$\sigma = 0.5568$	$\sigma = 0.61248$	$\sigma = 0.66816$	$\sigma = 0.734976$		
	Average Error	*Average Error*	*Average Error*	*Average Error*	*Average Error*	*Average Error*	*Average Error*
Y_2	0.10701	0.0677356	0.0375895	0.0192477	*diverging*	2.0143854	2.1349864
Y_3	0.11147	0.0708092	0.0392766	0.0202813	*diverging*	2.1013045	2.1835982
Y_4	0.23456731	0.0851456	0.0813783	0.0787494	*diverging*	2.158693	2. 2178432

From experimental results we can conclude that networks trained by the QHDH algorithm offer much better performance than trained by the QTH algorithm and by the QTL algorithm. Furthermore, it is observed that when σ increases, under the constraint defined by (29), the interpolation network generality is improved.

V. CONCLUSION

The HDH algorithm for training interpolation RBF networks presented in [15] improves significantly the quality of networks. On the other hand, in cases of equally spaced nodes, it does not exploit any advantage of this uniform distribution of nodes. By replacing Euclidean norm in Gaussian radial functions by an appropriately chosen Mahalanobis norm, we can conveniently preset the width parameters, and then use the second phase of the HDH algorithm to train interpolation networks. This new one-phase algorithm does not only reduce seriously the networks training time but also improves significantly the network generality. Simulation results show that the QHDH is really powerful when applied to problems with large number of interpolation nodes.

In practice, for arbitrarily distributed nodes of noisy data, the approximation problem might be solved by the following approach. The first step is to construct an appropriate uniform grid and at the nodes of this newly formed grid, the values of the target approximation function are computed by the linear regression technique applied to their K-NN points. Finally the interpolation RBF networks can be constructed in applying the QHDH algorithm over this new uniform grid.

One last interesting point to be mentioned is that in a recent research work [20], a kind of "optimum" choice for

the RBF parameters is proposed; but its complexity is $O(N^3)$. Our proposed method, while not optimum in any sense, has the complexity $O(N^2)$. This is why for large size problems, our algorithm is up to now the only one that can handle the situation.

REFERENCES

[1] R. H. Bartels, J. C. Beatty, and B. A. Barsky, *An Introduction on Splines for Use in Computer Graphics & Geometric Modeling*. San Mateo, CA: Morgan Kaufmann, 1987.

[2] E. Blanzieri, "Theoretical interpretations and applications of radial basis function networks," Inf. Telecomun., Univ. Trento, Trento, Italy, Tech. Rep. DIT-03-023, 2003.

[3] M. D. Buhmann, *Radial Basis Functions: Theory and Implementations*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[4] R. S. Buss, *3-D Computer Graphics: A Mathematical Introdution with OpenGL*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[5] P. J. Olver, "On multivariate interpolation," *Studies Appl. Math.*, vol. 116, no. 2, pp. 201–240, Feb. 2006.

[6] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[7] J. D. Powell, "Radial basis function approximations to polynomials," in *Proc. Numer. Anal.*, Dundee, U.K., 1987, pp. 223–241.

[8] D. S. Bromhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Syst.*, vol. 2, no. 3, pp. 321–355, 1988.

[9] J. Park and I. W. Sandberg, "Approximation and radial-basis-function networks," *Neural Comput.*, vol. 5, no. 2, pp. 305–316, Mar. 1993.

[10] T. Poggio and F. Girosi, "Networks for approximating and learning," *Proc. IEEE*, vol. 78, no. 9, pp. 1481–1497, Sep. 1990.

[11] E. Hartman, J. D. Keeler, and J. M. Kowalski, "Layered neural networks with Gaussian hidden units as universal approximations," *Neural Comput.*, vol. 2, no. 2, pp. 210–215, 1990.

[12] C. G. Looney, *Pattern Recognition Using Neural Networks: Theory and Algorithm for Engineers and Scientist*. New York: Oxford Univ. Press, 1997.

[13] M. Bortman and M. A. Aladjem, "A growing and pruning method for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 20, no. 6, pp. 1039–1045, Jun. 2009.

[14] J. P.-F. Sum, C.-S. Leung, and K. I.-J. Ho, "On objective function, regularizer, and prediction error of a learning algorithm for dealing with multiplicative weight noise," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 124–138, Jan. 2009.

[15] H. X. Huan, D. T. T. Hien, and H. T. Huynh, "A novel efficient two-phase algorithm for training interpolation radial basis function networks," *Signal Process.*, vol. 87, no. 11, pp. 2708–2717, Nov. 2007.

[16] D. T. T. Hien, H. X. Huan, and H. T. Huynh, "Multivariate interpolation using radial basis function networks," *Int. J. Data Mining, Model. Manage.*, vol. 1, no. 3, pp. 291–309, Jul. 2009.

[17] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[18] C. A. Micchelli, "Interpolation of scattered data: Distance matrices and conditionally positive definite functions," *Const. Approx.*, vol. 2, no. 1, pp. 11–22, 1986.

[19] M. H. Mousoun, *Fundamental of Artificial Neural Networks*. Cambridge, MA: MIT Press, 1995.

[20] G. E. Fasshauer and J. G. Zhang, "On choosing 'optimal' shape parameters for RBF approximation," *Numer. Algorith.*, vol. 45, nos. 1–4, pp. 345–368, 2007.

Embedded Feature Ranking for Ensemble MLP Classifiers

Terry Windeatt, Rakkrit Duangsoithong, and Raymond Smith

***Abstract*— A feature ranking scheme for multilayer perceptron (MLP) ensembles is proposed, along with a stopping criterion based upon the out-of-bootstrap estimate. To solve multi-class problems feature ranking is combined with modified error-correcting output coding. Experimental results on benchmark data demonstrate the versatility of the MLP base classifier in removing irrelevant features.**

***Index Terms*— Classification, multilayer perceptrons, pattern analysis, pattern recognition.**

I. INTRODUCTION

Whether an individual classifier or an ensemble of classifiers is employed to solve a supervised learning problem, finding relevant features for discrimination is important. Most previous research on feature relevancy has focussed on individual classifiers, but in this brief the issue is addressed for an ensemble of multilayer perceptron (MLP) classifiers. The extension of feature relevancy to classifier ensembles is not straightforward, because of the inherent trade-off between accuracy and diversity [1]. The trade-off has long been recognised, and arises because diversity must decrease as base classifiers approach the highest levels of accuracy. There is no consensus on the best way to measure ensemble diversity, and the relationship between irrelevant features and diversity is not known.

Feature relevancy is particularly important for small sample size problems, that is when the number of patterns is fewer than the number of features [2]. With tens of features in the original set, feature selection using an exhaustive search is computationally prohibitive. Since the problem is known to be NP-hard [3], a greedy search scheme is required, and filter, wrapper and embedded approaches have been developed [4]. The advantage of an embedded method is that feature selection is inherent in the classifier itself, and there is no reliance upon a measure that is independent of the classifier.

Feature ranking is conceptually one of the simplest search schemes for feature selection, and has the advantage of scaling up to hundreds of features. Uni-dimensional feature-ranking methods consider each feature in isolation, but are disadvantaged by the implicit orthogonality assumption [4], whereas multi-dimensional methods consider correlations with remaining features. In this brief, we propose an ensemble of MLP classifiers that incorporates multi-dimensional feature ranking based on MLP weights. The ensemble contains a simple parallel multiple classifier system (MCS) architecture with homogenous MLP base classifiers.

It is generally believed that MLP weights in a single classifier are not suitable for identifying relevant features [5].

Manuscript received November 9, 2010; revised March 24, 2011; accepted March 27, 2011. Date of publication May 13, 2011; date of current version June 2, 2011. This work was supported in part by the U.K. Government, Engineering and Physical Sciences Research Council, under Grant E061664/1.

The authors are with the Centre for Vision Speech and Signal Processing, Faculty of Electronics and Physical Sciences, University of Surrey, Guildford Surrey GU2 7XH, U.K. (e-mail: t.windeatt@surrey.ac.uk; R.Duangsoithong@surrey.ac.uk; Raymond.Smith@surrey.ac.uk).

Digital Object Identifier 10.1109/TNN.2011.2138158