MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY
and
CENTER FOR BIOLOGICAL INFORMATION PROCESSING
WHITAKER COLLEGE

A.I. Memo No. 1291 April 1992
C.B.I.P. Paper No.

A Connection between GRBF and MLP

Minoru Maruyama, Federico Girosi and Tomaso Poggio

Abstract

Both multilayer perceptrons (MLP) and Generalized Radial Basis Functions (GRBF) have good approximation properties, theoretically and experimentally. Are they related? The main point of this paper is to show that for normalized inputs, multilayer perceptron networks *are* radial function networks (albeit with a non-standard radial function). This provides an interpretation of the weights w as centers t of the radial function network, and therefore as equivalent to *templates*. This insight may be useful for practical applications, including better initialization procedures for MLP. In the remainder of the paper, we discuss the relation between the radial functions that correspond to the sigmoid for normalized inputs and well-behaved radial basis functions, such as the Gaussian. In particular, we observe that the radial function associated with the sigmoid is an activation function that is good approximation to Gaussian basis functions for a range of values of the bias parameter. The implication is that a MLP network can always simulate a Gaussian GRBF network (with the same number of units but less parameters); the converse is true only for certain values of the bias parameter. Numerical experiments indicate that this constraint is not always satisfied in practice by MLP networks trained with backpropagation. *Multiscale* GRBF networks, on the other hand, can approximate MLP networks with a similar number of parameters.

This paper describes research done within the Center for Biological Information Processing, in the Department of Brain and Cognitive Sciences, and at the Artificial Intelligence Laboratory. This research is sponsored by a grant from the Office of Naval Research (ONR), Cognitive and Neural Sciences Division; by the Artificial Intelligence Center of Hughes Aircraft Corporation; by the Alfred P. Sloan Foundation; by the National Science Foundation. Support for the A. I. Laboratory's artificial intelligence research is provided by the Advanced Research Projects Agency of the Department of Defense under Army contract DACA76–85–C–0010, and in part by ONR contract N00014–85–K–0124.

1 Introduction

Techniques and networks for learning from examples may be considered as methods for approximating multivariate functions from sparse data (the "examples"). In recent papers we have developed one such technique that we have called regularization networks, of which Radial Basis Functions are a special case and Hyper Basis Functions are the most general form (see for a review Poggio and Girosi, 1990 and references therein). The underlying theory is quite well developed and understood: for instance the role of the hidden units in the corresponding network (see Fig. (1)) is easily explained. The method has been also demonstrated to work well in a number of practical applications. Another technique, which is extremely popular, is associated with multilayer perceptrons, typically used in conjunction with a version of gradient descent (for estimating the parameters) called backpropagation. In this paper, we will consider multilayer perceptrons with one hidden layer and linear output units. These networks have been used successfully in many cases of learning from examples. The underlying theory is less developed, though a few theoretical results have been obtained in recent months. In particular, it has been proved that MLP have what we call the Weierstrass property, that is they can approximate arbitrarily well - provided enough units are available - any continuous function on a compact interval (Hornik, Stinchcombe and White, 1989; Stinchcombe and White, 1989; Carroll and Dickinson, 1989; Cybenko, 1989; Funahashi, 1989). Regularization networks also have this property, shared by many other approximation schemes (Girosi and Poggio, 1989).

It is natural to explore the relation between the two techniques, especially because the corresponding networks have superficially a similar structure, both having one hidden layer of units as shown by Fig. (1).

The network of Fig. (1) represents the class of functions of the type

$$f(\mathbf{x}) = \sum_{i=1}^{n} c_i H_i(\mathbf{x}) \tag{1}$$

where H_i are functions that depend on some parameters to be estimated together with the coefficients c_i. When the functions H_i are kept fixed, the function $f(\mathbf{x})$ is linear in its parameters (the c_i), and the resulting approximation scheme is *linear*.

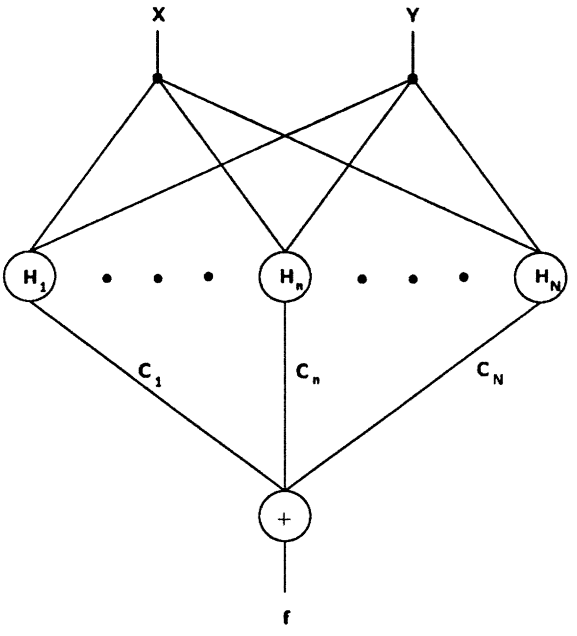


Figure 1: The most general network with one layer of hidden units. Here we show the two-dimensional case, in which $\mathbf{x} = (x, y)$. Each function H_i can depend on a set of unknown parameters, that are computed during the learning phase, as well as the coefficients c_i. When $H_i = \sigma(\mathbf{w}_i \cdot \mathbf{x} + \theta_i)$ the network is a multilayer perceptron; when $H_i = h(\mathbf{x} - \mathbf{t}_i)$ the network is a regularization network for appropriate choices of h.

Depending on the function H_i different approximation schemes can be obtained. Two common choices are the following:

1. **Regularization networks**, that correspond to the choice:

$$H_i(\mathbf{x}) = h(\|\mathbf{x} - \mathbf{t}_i\|_{\mathbf{W}})$$

where h is a (conditionally) positive definite function, the $\mathbf{t}_i$ are d-dimensional vectors called "centers", $\mathbf{W}$ is a $d \times d$ matrix, and we have defined the *weighted norm* $\|\cdot\|_{\mathbf{W}}$ as

$$\|\mathbf{x}\|^2_{\mathbf{W}} = \mathbf{x}^T\mathbf{W}^T\mathbf{W}\mathbf{x} \; .$$

We call this scheme Hyperbf. RBF is the case in which the centers coincide with the data points, and GRBF (Generalized Radial Basis Functions) is the case in which the centers $\mathbf{t}_\alpha$ are free parameters to be estimated but the weight matrix $\mathbf{W}$ is fixed and equal to the identity matrix.

This class of approximation schemes can be formally derived, in the framework of regularization theory, from a variational principle which has a Bayesian interpretation. It includes:

- kernel estimation techniques and Parzen windows
- splines
- Hyperbf and RBF

2. **Ridge functions approximation:**

$$H_i(\mathbf{x}) = h_i(\mathbf{x} \cdot \mathbf{w}_i + \theta_i)$$

where the $\mathbf{w}_i$ are d-dimensional vectors called "weights", and the parameters θ_i constitutes bias terms. This form of approximation did not have until now any variational or Bayesian interpretation. Very recent results by Poggio and Girosi (unpublished) show that a set of ridge function approximation schemes can be derived as limits of regularization networks for an appropriate class of stabilizers. Several techniques are included in this class:

- **Projection Pursuit Regression (PPR):** it corresponds to an expansion of the type eq. (1) in which all the coefficients c_i are set to 1, and

$$H_i(\mathbf{x}) = h_i(\mathbf{x} \cdot \mathbf{w}_i)$$

where the vectors $\mathbf{w}_i$ are normalized ($\|\mathbf{w}_i\| = 1$). The functions h_i are determined by means of some nonparametric estimation technique, in the following iterative way:

(a) Assume that we already have $k-1$ terms. Let

$$r_i = y_i - \sum_{j=1}^{k-1} h_j(\mathbf{x}_i \cdot \mathbf{w}_j)$$

be the residual of the approximation;

(b) Search for the next term. Calculate the following sum of the residulas

$$\sum_{i=1}^{N} (r_i - g_k(\mathbf{x}_i \cdot \mathbf{w}_k))^2$$

and find the direction $\mathbf{w}_k$ which minimize the sum of residuals and the corresponding function h_k.

- **Flexible Fourier series:** the function h_i are all equal to the cosine (or sine) function, and therefore:

$$H_i(\mathbf{x}) = \cos(\mathbf{x} \cdot \mathbf{w}_i + \theta_i) \ . \tag{2}$$

If we assume that the function g underlying the data has a Fourier transform $\tilde{g}(\mathbf{s}) = |\tilde{g}(\mathbf{s})| e^{i\theta(\mathbf{s})}$, then

$$g(\mathbf{x}) = \Re \left(\int_{R^d} d\mathbf{s} \ e^{i\mathbf{s}\cdot\mathbf{x}} |\tilde{g}(\mathbf{s})| e^{i\theta(\mathbf{s})} \right) = \int_{R^d} d\mathbf{s} \ |\tilde{g}(\mathbf{s})| \cos(\mathbf{s}\cdot\mathbf{x} + \theta(\mathbf{s})) \ , \tag{3}$$

and the expansion of eq. (1) with the choice (2) looks like a cubature formula for the integral above, where the vectors $\mathbf{w}_i$ are the points at which the integrand is sampled. In this case the

interpretation of the "weights" $\mathbf{w}_i$ is clear: they represent the fundamental frequency content of the underlying function. It can also be noticed that the problem of finding the optimal parameters of the parametric function is equivalent to the problem of finding the optimal sample points to use in the discretization of the integral above.

- **Multilayer Perceptrons:** the functions h_i are all equal to a sigmoidal function:

$$H_i(\mathbf{x}) = \sigma(\mathbf{x} \cdot \mathbf{w}_i + \theta_i) \ . \tag{4}$$

The function σ is usually defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

but other choices, (as the hyperbolic tangent), are equivalent, as long as the sigmoidal shape is mantained.

While in the approximation by trigonometric functions the parameters $\mathbf{w}_i$ have a simple interpretation in terms of frequencies, in this case the meaning of the $\mathbf{w}_i$ is less clear. If the expansion (4) is considered from the point of view of projection pursuit regression, (Friedman and Stuetzle, 1981; Huber, 1985) the $\mathbf{w}_i$ are the "relevant directions" that are supposed to encode the most information about the function.

- **Exponential sums:** when the problem is one dimensional, a well known non linear technique consists in approximation by exponential sums (Braess, 1986; Gosselin, 1986; Hobby and Rice, 1967), and a number of results are available in this case. In more than one dimension, the natural extension corresponds to ridge function approximation with the choice:

$$H_i(\mathbf{x}) = e^{-\mathbf{x} \cdot \mathbf{w}_i} \ . \tag{5}$$

Notice that the bias terms disappear, since they are absorbed by the coefficients.

We group the ridge function approximation schemes together because they all have the same general form: linear combination of nonlinear function of the scalar product of the input vector with the parameter vector.

The main difference between these two classes of approximation schemes (which, as we mentioned earlier, can be both derived from regularization in terms of two different classes of stabilizers, reflecting different prior assumptions [1]) seems to be related to the use of the scalar product $\mathbf{x} \cdot \mathbf{w}_i$ instead of the weighted Euclidean distance $\|\mathbf{x} - \mathbf{t}_i\|_{\mathbf{W}}$, as argument of the parametric functions h.

At first sight, these two broad classes of techniques do not seem to have any relationship. The main point of this paper is to show that in some special situations there is a close connection between these two classes of approximation schemes and in particular between Gaussian GRBF (i.e. Hyperbf networks with $\mathbf{W} = I$) and MLP with sigmoidal units in the hidden layer.

2 Normalized Inputs: Ridge Functions are Radial Functions

In the case in which all the inputs variables are normalized, that is they lie on the unit d-dimensional sphere, there is a simple connection between ridge and radial functions. In fact the following identity

$$\|\mathbf{x} - \mathbf{t}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{t}\|^2 - 2\, \mathbf{x} \cdot \mathbf{t} \tag{6}$$

for $\|\mathbf{x}\| = 1$ becomes:

$$\|\mathbf{x} - \mathbf{t}\|^2 = 1 + \|\mathbf{t}\|^2 - 2\, \mathbf{x} \cdot \mathbf{t} \tag{7}$$

We can now use identity (7) to obtain a ridge regression scheme from a Radial Basis Functions scheme and vice versa.

[1] The formulation of the learning problem in terms of regularization is satisfying from a theoretical point of view, since it establishes connections with a large body of results in the area of Bayes estimation and in the theory of approximation of multivariate functions.

- **From radial basis functions to multilayer perceptrons.**

Substituting identity (7) in the Radial Basis Functions expansion

$$f(\mathbf{x}) = \sum_{\alpha=1}^{N} c_\alpha H(\|\mathbf{x} - \mathbf{t}_\alpha\|^2) \tag{8}$$

we obtain

$$f(\mathbf{x}) = \sum_{\alpha=1}^{N} c_\alpha \tilde{H}(\mathbf{x} \cdot \mathbf{t}_\alpha + \theta_\alpha) \ , \tag{9}$$

where we have defined

$$\tilde{H}(x) = H(-2x) \ , \quad \theta_\alpha = -\frac{1}{2}(1 + \|\mathbf{t}_\alpha\|^2) \ . \tag{10}$$

We notice that eq. (10) is the expansion corresponding to a multilayer perceptron network. The only difference is that while in the multilayer perceptron network the bias parameter θ_α is allowed to vary along the real line, in this case it is costrained to lie in the interval $(-\infty, -\frac{1}{2}]$. We therefore can conclude that, when inputs are normalized, given the RBF network of eq. (8) with activation function H it is always possible to define a multilayer perceptron with the same number of units and with activation function $\tilde{H}$ that computes the same function. The synaptic weights connecting the input and the hidden layer are the centers of the Radial Basis Functions expansion, and the bias parameters are uniquely determined by the synaptic weights.

- **From multilayer perceptron to radial basis functions.**

In the previous case we have seen that Radial Basis Functions can be simulated by multilayer perceptron. This is not surprising since a Radial Basis Functions unit has one parameter less than a multilayer perceptron unit. For the same reason we cannot expect to simulate a multilayer perceptron unit with d inputs in terms of a Radial Basis Functions unit with the same number of inputs. However this may

be possible if we add to the Radial Basis Functions units a dummy input, so that the coordinate of the center corresponding to the dummy variable gives the missing parameter. Let us consider the multilayer perceptron expansion:

$$f(\mathbf{x}) = \sum_{\alpha=1}^{N} c_\alpha \sigma(\mathbf{x} \cdot \mathbf{w}_\alpha + \theta_\alpha) \ . \tag{11}$$

Using identity (7) we obtain:

$$f(\mathbf{x}) = \sum_{\alpha=1}^{N} c_\alpha \sigma(-\frac{1}{2}\|\mathbf{x} - \mathbf{w}_\alpha\|^2 + d_\alpha) \ , \tag{12}$$

where we have defined

$$d_\alpha = \frac{1}{2}(1 + \|\mathbf{w}_\alpha\|^2) + \theta_\alpha$$

We now rewrite the expansion as a Radial Basis Functions expansion in $d+1$ dimensions, in which one of the input variables is kept fixed, and equal to some number (1, for simplicity). Introducing the $d+1$ dimensional vectors

$$\tilde{\mathbf{x}} = (\mathbf{x}, 1) \ , \quad \tilde{\mathbf{w}}_\alpha = (\mathbf{w}, v_\alpha) \ .$$

we can rewrite the expansion (12) as

$$f(\mathbf{x}) = \sum_{\alpha=1}^{N} c_\alpha \sigma(-\frac{1}{2}\|\tilde{\mathbf{x}} - \tilde{\mathbf{w}}_\alpha\|^2 + \frac{1}{2}(1 - v_\alpha)^2 + d_\alpha) \ . \tag{13}$$

Expansion (13) becomes a Radial Basis Functions expansion if one of the two following conditions is satisfied:

1. There exists v_α such that

$$\frac{1}{2}(1 - v_\alpha)^2 = -d_\alpha \ ; \tag{14}$$

2. There exist functions g_1 and g_2 such that

$$\sigma(-\frac{1}{2}x + y) = g_1(x)g_2(y) \ . \tag{15}$$

In the first case, in fact, the multilayer perceptron expansion becomes

$$f(\mathbf{x}) = \sum_{\alpha=1}^{N} c_\alpha h(\|\tilde{\mathbf{x}} - \tilde{\mathbf{w}}_\alpha\|^2) \tag{16}$$

where we have defined the function $h(x) = \sigma(-\frac{1}{2}x)$. In the second case the multilayer perceptron expansion becomes

$$f(\mathbf{x}) = \sum_{\alpha=1}^{N} c'_\alpha g_1(\|\tilde{\mathbf{x}} - \tilde{\mathbf{w}}_\alpha\|^2) \ . \tag{17}$$

where $c'_\alpha = g_2(d_\alpha + \frac{1}{2}(1 - v_\alpha)^2)$.

Remarks

- In case (1), a solution to eq. (14) does not exist unless the condition

  $$\frac{1}{2}(1 + \|\mathbf{w}_\alpha\|^2) + \theta_\alpha \leq 0$$

  is satisfied. This condition on the weights $\mathbf{w}_\alpha$ and on the bias parameter θ_α defines a subclass of multilayer perceptrons that can be simulated by a Radial Basis Function network with the same number of units and a dummy input, and therefore the same number of parameters.
- In case (2), evaluating eq. (15) first at $x = 0$ and $y = 0$ we obtain that

  $$g_1(x) = \frac{\sigma(-\frac{1}{2}x)}{g_2(0)} \ , \quad g_2(y) = \frac{\sigma(y)}{g_1(0)} \ ,$$

and after some algebra

$$\sigma(x+y) = \frac{\sigma(x)\sigma(y)}{\sigma(0)} \ . \tag{18}$$

It is well known that this functional equation has only one class of solutions, given by

$$\sigma(x) = ce^x \ , \quad \forall c \in R \ .$$

Therefore radial units can simulate arbitrary multilayer perceptron units only in the case in which the activation function of the multilayer perceptron network is an exponential function. In this case the corresponding Radial Basis Functions network is a Gaussian network. In fact, setting $\sigma(x) = e^x$ in eq. (13) we obtain the expansion

$$f(\mathbf{x}) = \sum_{\alpha=1}^{N} c'_\alpha e^{(-\frac{1}{2}\|\mathbf{x}-\mathbf{w}_\alpha\|^2)} \ .$$

Notice also that in this case there is no need of dummy input, since the bias term can be dropped out by a redefinition of the coefficients, due to the property (18) of the exponential function.

3 Sigmoid MLPs and Gaussian GRBFs

In this section, we compare the sigmoid and Gaussian function for normalized input vectors $\|\mathbf{x}\| = 1$. Under these conditions the sigmoidal function becomes a radial function, that can approximate any Gaussian function well enough within a certain range of the bias parameter θ.

For normalized inputs, any ridge functions is equivalent to an appropriate radial function. Consider the sigmoid function given by

$$\sigma(\mathbf{w}\cdot\mathbf{x}+\theta) = \frac{1}{1+\exp(-(\mathbf{w}\cdot\mathbf{x}+\theta))} \quad \mathbf{w} \in \mathbf{R}^d, \ \theta \in \mathbf{R}$$

The corresponding radial function parametrized by $\lambda \in \mathbf{R}$ is :

$$
\begin{aligned}
s\sigma(\mathbf{w}\cdot\mathbf{x}+\theta) &= \frac{s}{1+\exp(-(\mathbf{w}\cdot\mathbf{x}+\theta))} \quad (s\in\mathbf{R},\ \mathbf{w}\in\mathbf{R}^d,\ \theta\in\mathbf{R}) \\
&= \frac{s}{1+C(\lambda)\exp(\lambda\|\mathbf{x}-\mathbf{t}\|^2)} \quad (C(\lambda)>0) \qquad (19)
\end{aligned}
$$

where we have defined :

$$
\mathbf{t}=\frac{\mathbf{w}}{2\lambda},\quad C(\lambda)=\exp(-(\theta+\lambda(1+\frac{\|\mathbf{w}\|^2}{4\lambda^2}))) \qquad (20)
$$

3.1 Sigmoids units can approximate Gaussian units

The existence of free parameter λ indicates that any element of the one-parameter family given by (19) is equivalent to each other for normalized inputs. To compare Gaussians and the radial functions associated with sigmoids, we should measure the discrepancy between the two functions. For the purpose of the comparison, we first take the *closest* function to a Gaussian among all the elements in the one-parameter family defined above. It turns out that the radial function (19) approximates the Gaussian function well, if $C(\lambda)\gg 1$ holds (see figures 2 and 3). According to this observation, adopting $C(\lambda)$ as a measure of *closeness* to the Gaussian, we consider the function whose parameter λ^* corresponds to the maximum of $C(\lambda)$:

$$
C(\lambda^*)=\max_{\lambda>0} C(\lambda)
$$

Solving the equation $\partial C/\partial\lambda = 0$, we get :

$$
\lambda^*=\|\mathbf{w}\|/2
$$

Substituting $\lambda=\|\mathbf{w}\|/2$, we obtain the following radial function $\mathbf{R}^d\rightarrow\mathbf{R}$, which has a *center* on a unit sphere S^{d-1}:

$$
\begin{aligned}
s\sigma(\mathbf{w}\cdot\mathbf{x}+\theta) &= \frac{s}{1+C(\theta,\|\mathbf{w}\|)\exp(\frac{\|\mathbf{w}\|}{2}\|\mathbf{x}-\frac{\mathbf{w}}{\|\mathbf{w}\|}\|^2)} \qquad (21) \\
C(\theta,\|\mathbf{w}\|) &= \exp(-(\theta+\|\mathbf{w}\|))
\end{aligned}
$$

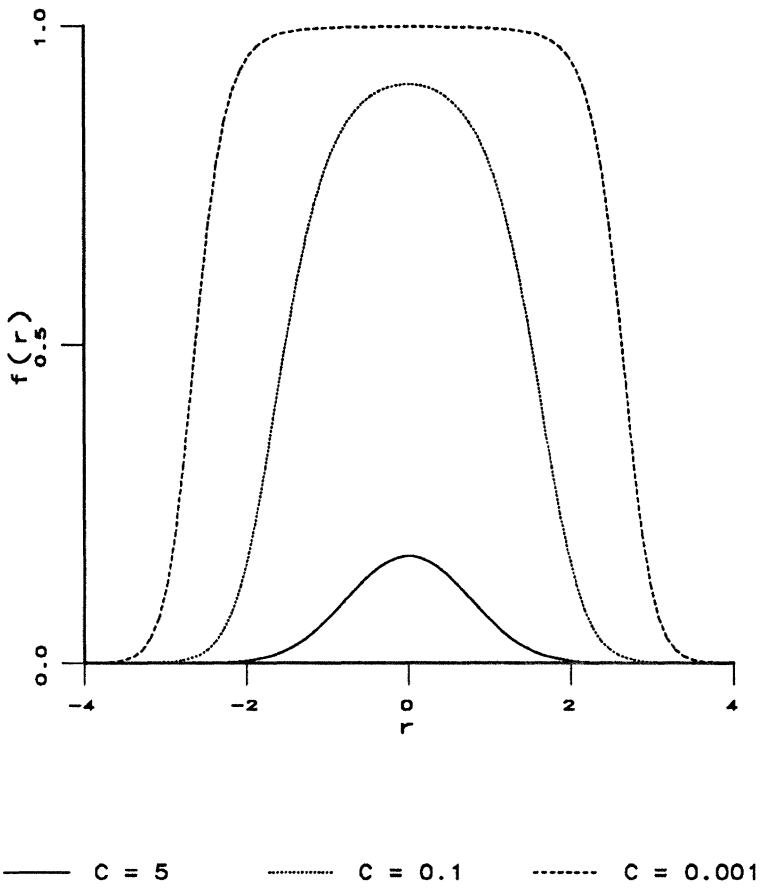


Figure 2: The radial function $f(r) = 1/(1 + Ce^{r^2})$ associated with sigmoids (see eq. 19), for 3 different values of C: $C = 5, 10^{-1}, 10^{-3}$.

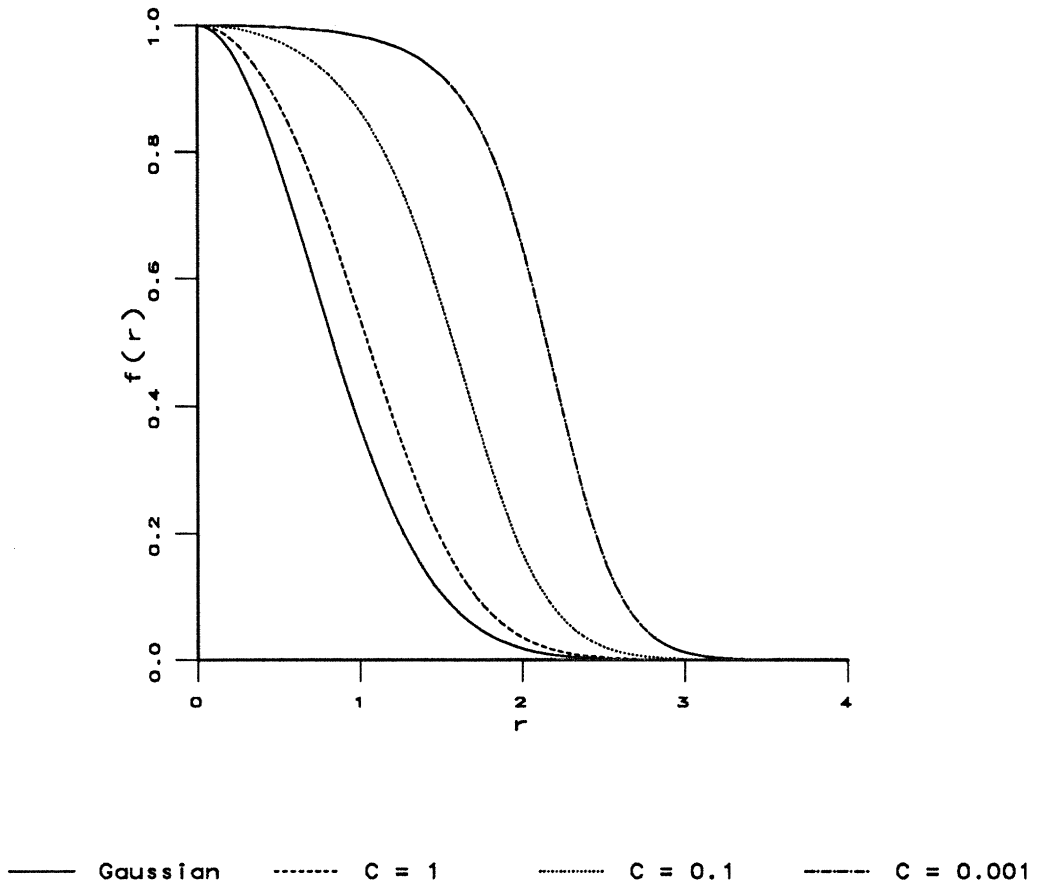


Figure 3: The gaussian function $f(r) = e^{-r^2}$ and the radial functions $f(r) = \frac{(1+C)}{(1+Ce^{x^2})}$ for 5 different values of C: $C = 1, 10^{-1}, 10^{-3}$. The role of the scale factor $1 + C$ is to enforce the value at the origin to be 1, to ease the comparison with the gaussian.

As shown in figures (2) and (3), the radial function (21) is quite similar to Gaussian functions if C satisfies $C \gg 1$, or equivalently, if the bias parameter θ satisfies

$$\theta \ll -\|\mathbf{w}\| \tag{22}$$

This implies that a MLP network can always simulate any Gaussian GRBF network well enough, if both of the networks consist of the same number of units (of course the MLP network has one more free parameter per hidden unit than a GRBF network), since θ can always be chosen to satisfy (22).

3.2 The Gaussian Radial Basis Function on the sphere

Let us first consider a Gaussian function for normalized inputs. When the d-dimensional input vectors $\mathbf{x} \in \mathbf{R}^d$ are normalized as $\|\mathbf{x}\| = 1$, for any Gaussian function, the following relation holds:

$$c\exp(-\mu^2\|\mathbf{x}-\mathbf{t}\|^2) = c'(\lambda)\exp(-\mu'(\lambda)^2\|\mathbf{x}-\mathbf{t}'(\lambda)\|^2)$$

where $\lambda \in \mathbf{R}$ $(\lambda \neq 0)$ is an arbitrary parameter and

$$c'(\lambda) = c\exp(-\mu^2(1-\lambda)(1-\frac{\|\mathbf{t}\|^2}{\lambda})),\ \mu'(\lambda) = \lambda\mu^2,\ \mathbf{t}'(\lambda) = \frac{\mathbf{t}}{\lambda}$$

This shows the representation $c\exp(-\mu^2\|\mathbf{x}-\mathbf{t}\|^2)$ has redundancy for normalized inputs. For example, for any Gaussian function $c\exp(-\mu^2\|\mathbf{x}-\mathbf{t}\|^2)$, the following types of Gaussians, which are given by setting $\lambda = \|\mathbf{t}\|$ and $\lambda = \mu^{-2}$, respectively, are equivalent for normalized inputs.

$$\begin{aligned} c\exp(-\mu^2\|\mathbf{x}-\mathbf{t}\|^2) &= c'\exp(-\mu^2\|\mathbf{t}\|\|\mathbf{x}-\frac{\mathbf{t}}{\|\mathbf{t}\|}\|^2)\ (c' = c\exp(-\mu^2(1-\|\mathbf{t}\|)^2)) \\ &= c''\exp(-\|\mathbf{x}-\mu^2\mathbf{t}\|^2)\ (c'' = c\exp((1-\mu^2)(1-\frac{\|\mathbf{t}\|^2}{\mu^2}))) \end{aligned}$$

The above indicates that the total number of free parameters for a Gaussian for normalized inputs is $d+1$, and that we may use either Gaussians with normalized centers,

$$c\exp(-\mu^2\|\mathbf{x}-\mathbf{t}\|^2),\ \|\mathbf{t}\|=1$$

or those given by

$$c\exp(-\|\mathbf{x}-\mathbf{t}\|^2)$$

as basis functions for normalized inputs.

3.3 Can Gaussian units approximate sigmoid units?

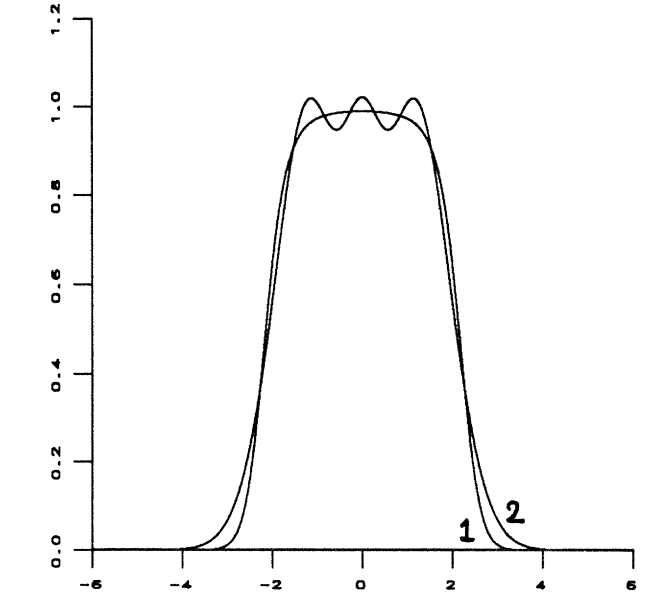
¿From the above observation, we see that MLP can be more flexible than Radial Basis Functions for normalized inputs, if the total number of units is the same, and therefore the total number of parameters is larger for MLPs. This is based upon one-to-one comparison of each unit of the networks. We expect that it may be possible to approximate the radial function (21) using a set of m Gaussians with the same *center*:

$$\frac{s}{1+Ce^{\beta r^2}} \sim \sum_{j=1}^{m} a_j e^{-b_j r^2}\ (b_j > 0) \tag{23}$$

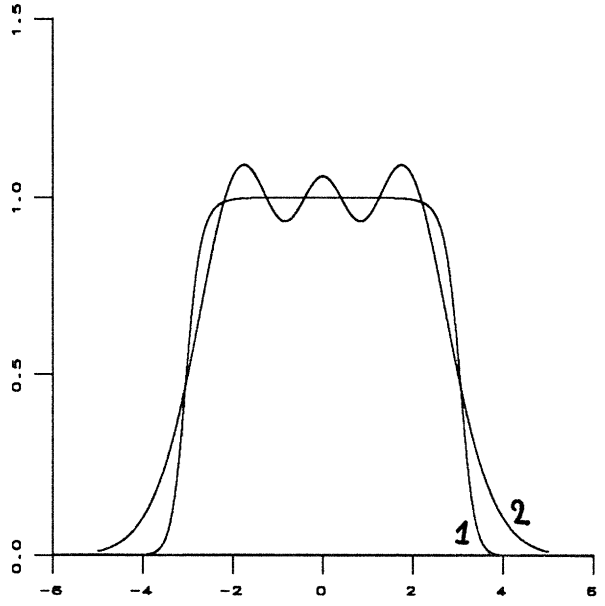
Figure (4) shows the experimental results of approximating $C = 0.01, 0.0001$ using 3 Gaussians for each radial function (sigmoid). The results imply the possibility of approximating a sigmoid by a set of Gaussians. The number of parameters of a sigmoid is $d+2$. However, $d+2m-1$ parameters are required for Gaussians to approximate a sigmoid according to (23) (thus in the experiments, the total number of parameters of a sigmoid and a set of Gaussians are $d+2$ and $d+5$, respectively.) In this subsection, we consider the possibility of approximating a sigmoid by a set of Gaussians with similar number of parameters.

3.3.1 Approximation by Gaussians with Constant Scales

In (Poggio and Girosi, 1990a) , we have extended our learning theory based on regularization and proposed the use of radial basis functions at multiple scales. To explore the possibility of approximating MLP by *Multiscale* GRBF with a similar number of parameters, we consider the following basis function with constant scales $\{B_i\}_{1=1}^{m}$:



a.



b.

Figure 4: Approximation of the radial function $f(r) = 1/(1+C\exp(r^2))$ (plot 1) by superposition of 3 Gaussians (plot 2): (a) $C = 0.01$, (b) $C = 0.0001$

$$\phi_\alpha = \sum_{j=1}^{m} a_{\alpha j} e^{-B_j \|\mathbf{x}-\mathbf{t}_\alpha\|^2}, \ (\|\mathbf{t}_\alpha\| = 1) \tag{24}$$

which approximate the radial function (21)

$$\frac{1}{1 + C \exp(\frac{w_\alpha}{2} \|\mathbf{x} - \mathbf{t}_\alpha\|^2)}$$

In this case, the above *multiscale* function ϕ_α has $d+m-1$ parameters. Since the center $\mathbf{t}_\alpha$ and the input $\mathbf{x}$ are normalized, the condition

$$0 \leq \|\mathbf{x} - \mathbf{t}_\alpha\| \leq 2$$

is satisfied. Therefore the coefficients $\{a_j\}$ are determined solving the minimization problem :

$$H \equiv \int_0^2 \{\frac{1}{1 + Ce^{\frac{w}{2}r^2}} - \sum_{j=1}^{m} a_j e^{-B_j r^2}\}^2 dr \rightarrow \min ,$$

and are given by the solution of the following linear system:

$$U\mathbf{a} = \mathbf{v} ,$$

$$U_{ij} = \int_0^2 e^{-(B_i+B_j)r^2} dr, \ v_i = \int_0^2 \frac{e^{-B_i r^2}}{1 + Ce^{\frac{w}{2}r^2}} dr .$$

In Fig. 5, 6 and 7 we show experimental results of the approximation error for $(m = 3, 4, 5)$ (the number of parameters of each scheme is thus $d+2, d+3, d+4$, respectively). The approximation errors are evaluated as :

$$\epsilon = H / \int_0^2 \frac{dr}{(1 + Ce^{\frac{w}{2}r^2})^2}$$

In the experiments, the scales $\{B_j\}$ are given by the results of experiment (23). The results show that, if the length of the *weight* is not too large, sigmoid for normalized inputs can be well approximated by superposition of Gaussian with the same number of parameters.

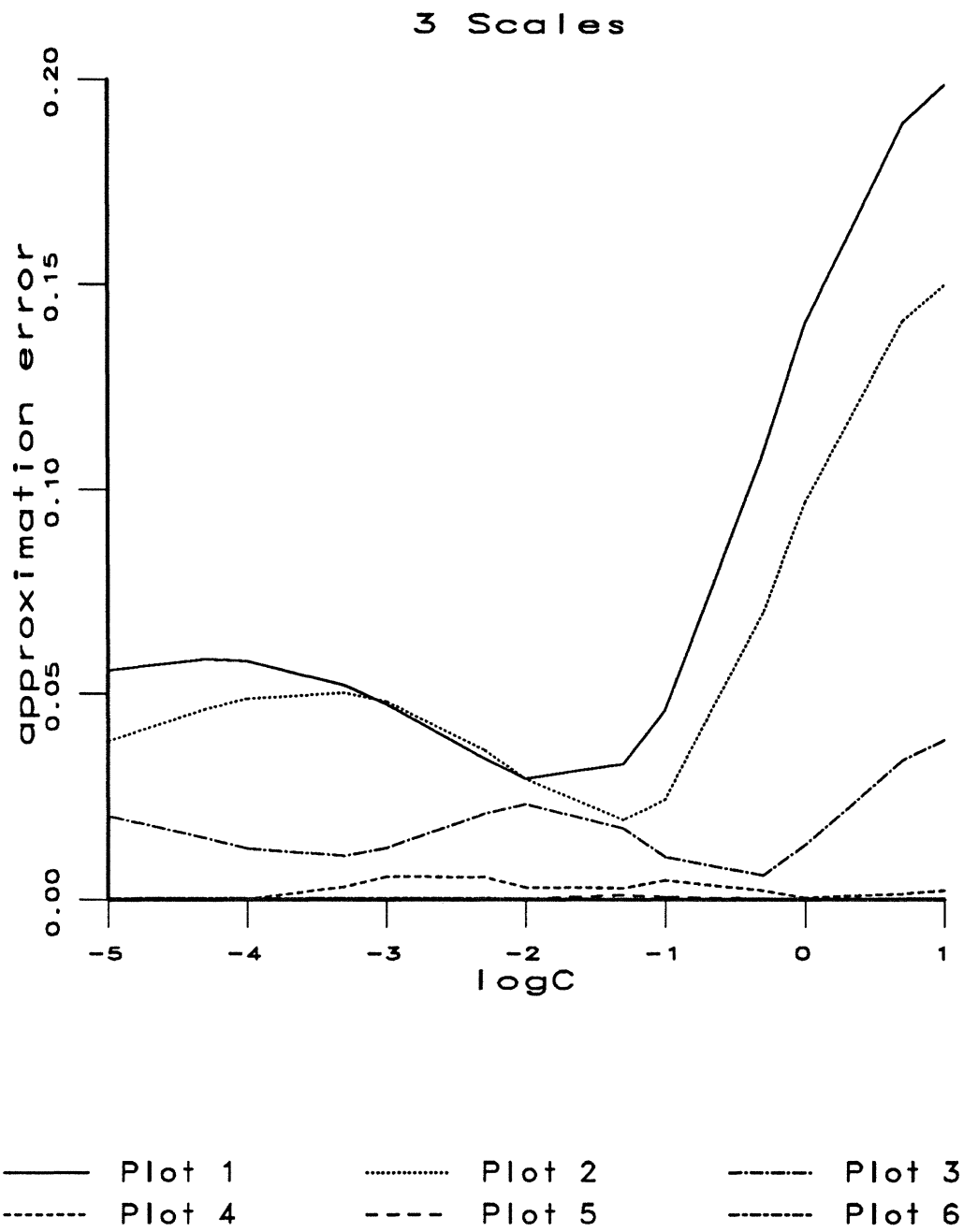


Figure 5: Approximation of radial function by Multiscale Gaussian: 3 scales.

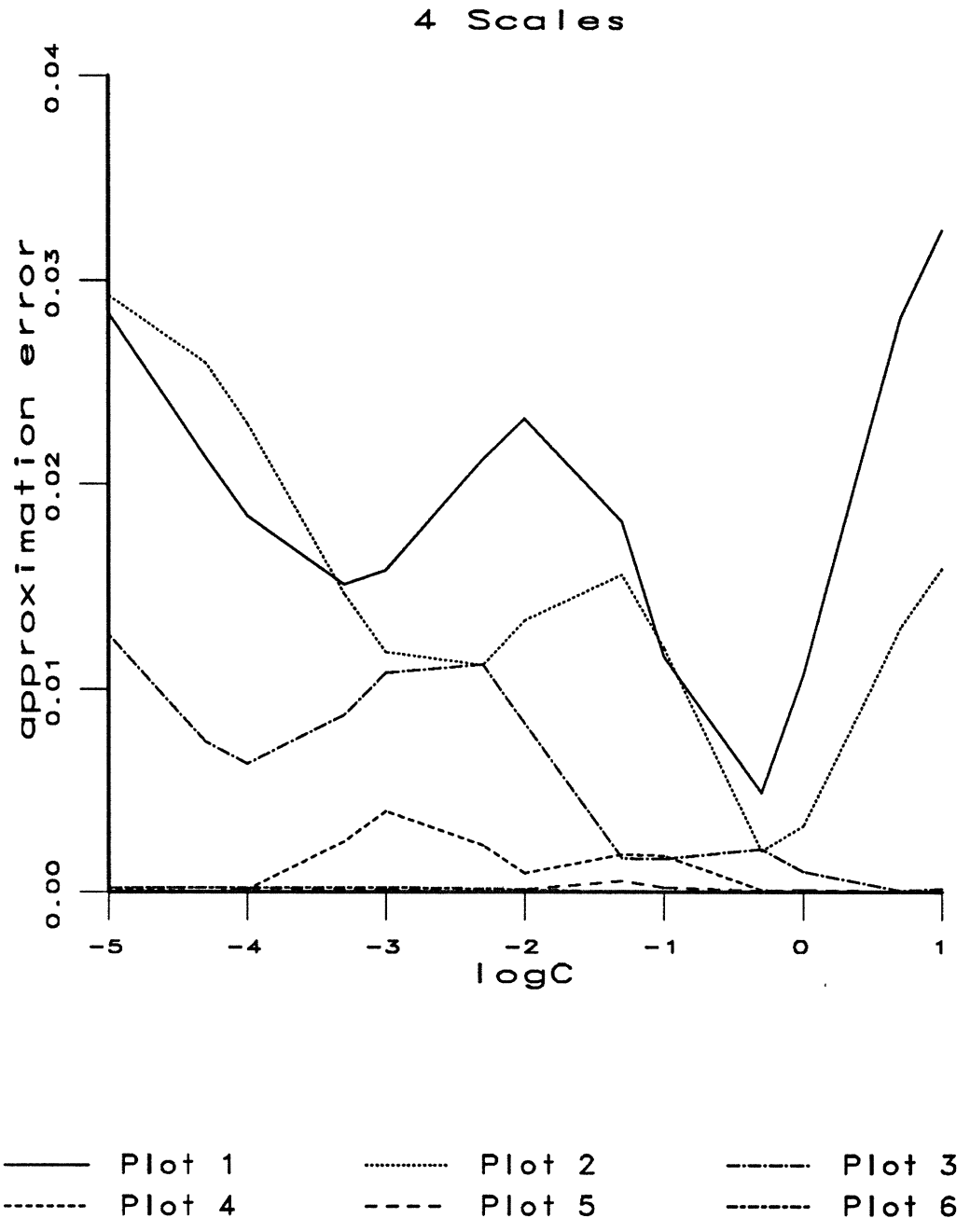


Figure 6: Approximation of radial function by Multiscale Gaussian: 4 scales.

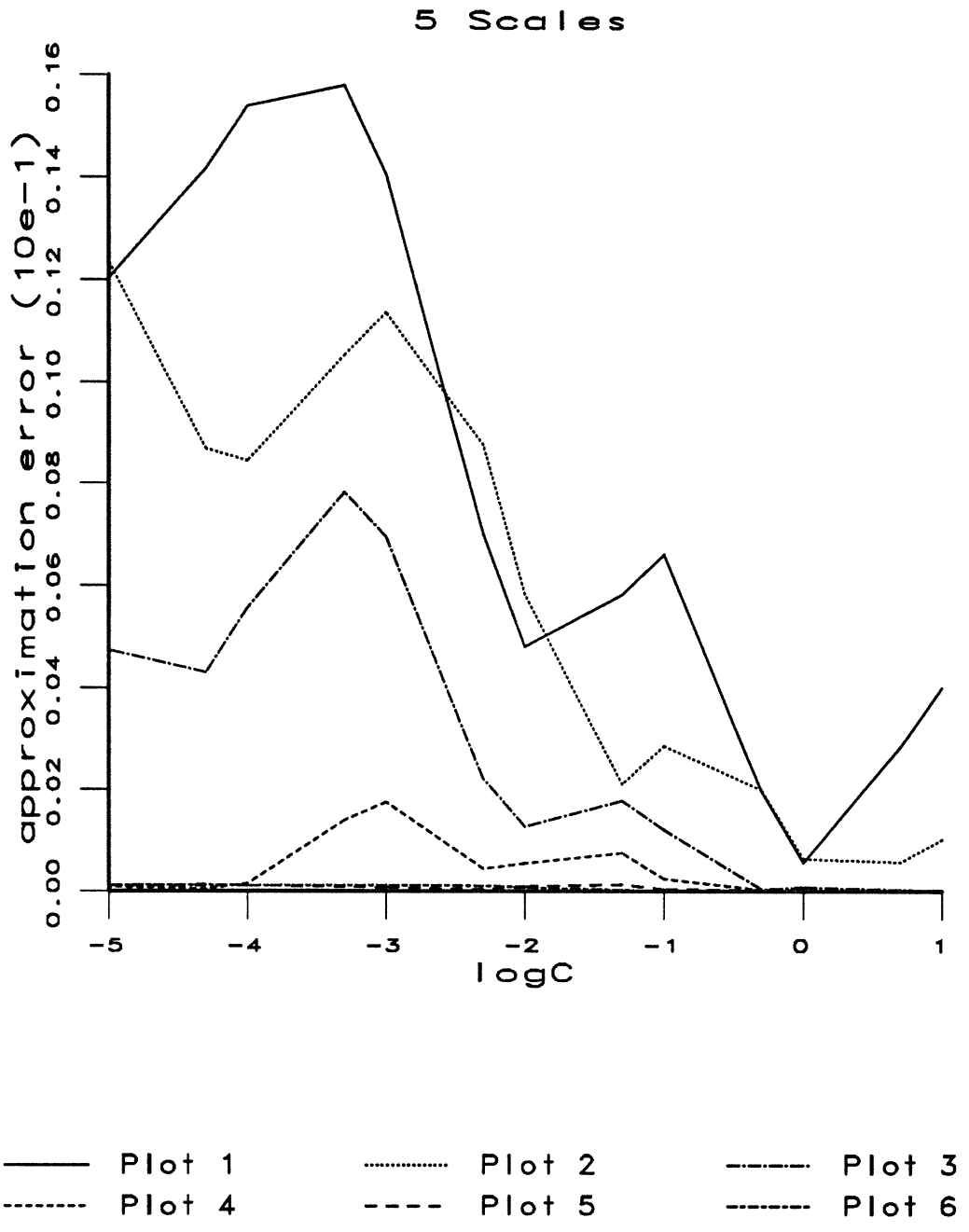


Figure 7: Approximation of radial function by Multiscale Gaussian: 5 scales.

3.3.2 Approximation by Gaussians with Fixed Ratio of Scales

In the previous subsection, we have shown that superposition of Gaussians could approximate sigmoid units very well for certain range of parameters with the same (or similar) number of parameters. In this subsection we consider, as another possibility of approximating any sigmoid with *Multiscale* GRBF, the following basis function :

$$\phi_\alpha = \sum_{j=1}^{3} a_{\alpha j} e^{-\beta_\alpha B_j \|\mathbf{x}-\mathbf{t}_\alpha\|^2}, \ (\|\mathbf{t}_\alpha\| = 1) \tag{25}$$

where *ratio* of scales B_j $(j = 1, 2, 3)$ are constants. The total number of parameters of this basis function is $d + 3$. In Figure (8), we show some results of approximation experiments. In the experiments, $\{B_j\}$ are given from the results of experiment (23) as before and $\{a_j\}$ are determined so that

$$H = \int_0^\infty \{\frac{1}{1+Ce^{r^2}} - \sum_{j=1}^{3} a_j e^{-B_j r^2}\}^2 dr \rightarrow min$$

Optimal $\{a_j\}$ are obtained by solving the linear equations :

$$U\mathbf{a} = \mathbf{v}$$

$$U_{ij} = \frac{1}{2}\sqrt{\frac{\pi}{B_i + B_j}}, \ v_i = \frac{1}{\sqrt{B_i}} \int_0^\infty \frac{e^{-r^2}}{1 + Ce^{r^2/B_i}} dr$$

Table (1) shows the approximation errors given by :

$$\epsilon = H / \int_0^\infty \frac{dr}{(1+Ce^{r^2})^2}$$

4 Experiments

In the previous section, we have shown that sigmoid MLPs can always simulate any gaussian GRBF network when both of the networks consist of the same number of units (sigmoids and Gaussians). The converse is also true if MLP's bias parameters are restricted to a certain range given by (22). To investigate whether this constraint is always satisfied in practice, numerical

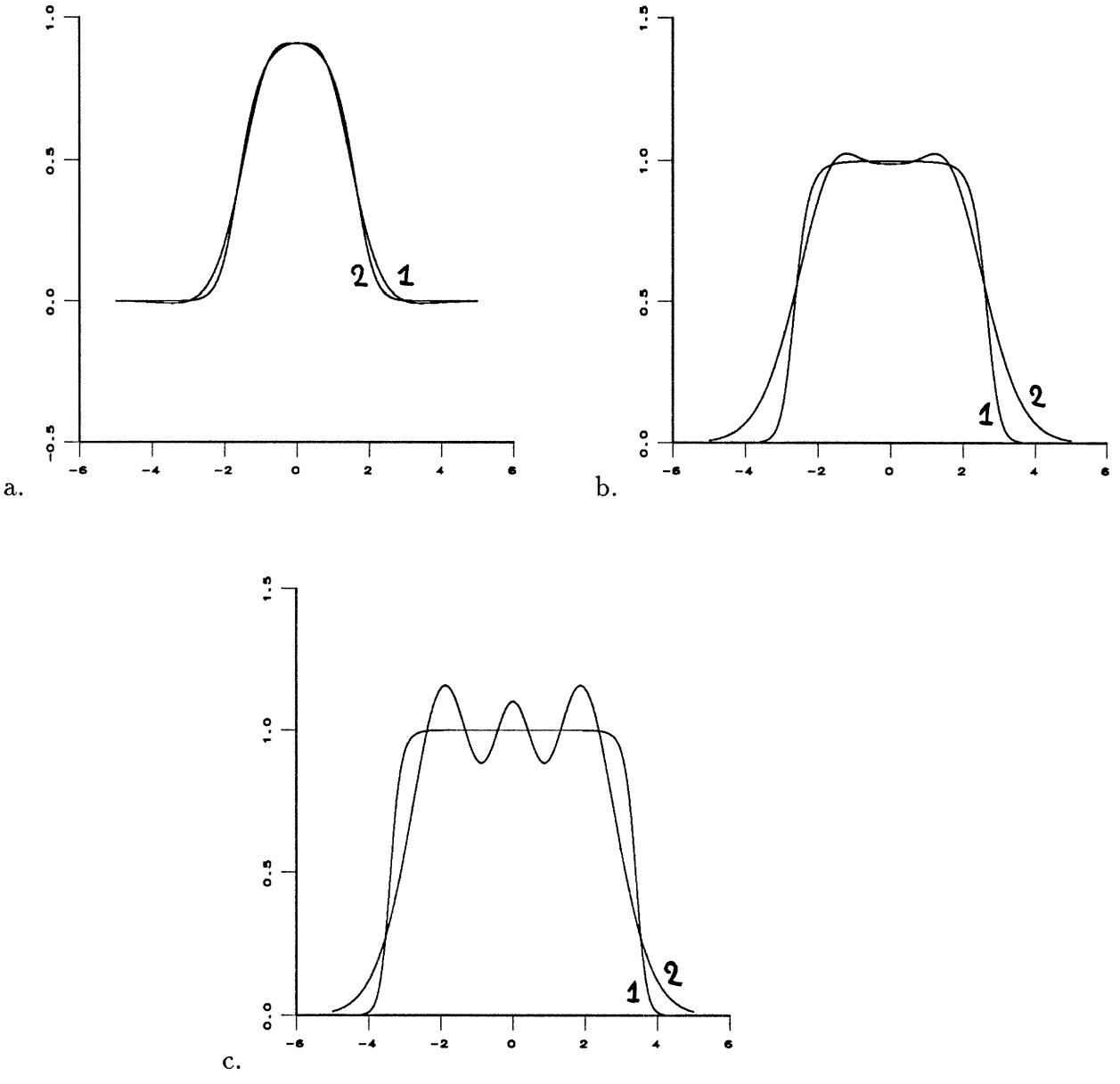


Figure 8: Approximation of the radial function $f(r) = 1/(1 + C\exp(r^2))$ (plot 1) by superposition of 3 Gaussians with fixed ratio of scales (plot 2): (a) $C = 0.1$, (b) $C = 0.001$, (c) $C = 0.00001$.

C	1	0.1	0.01	0.001	0.0001	0.00001
ϵ	0.00165	0.00243	0.0171	0.0204	0.0162	0.0307

Table 1: The relative approximation error ϵ that is obtained approximating a sigmoid with a multiscale GRBF in which the ratio of the variances of the Gaussians are kept fixed.

experiments were carried out. Even if the original inputs are not normalized, it is always possible to get normalized inputs by adding one more dimension as follows:

$$\mathbf{x} = (x_1, \ldots, x_d) \quad |x_i| \leq \ell$$

$$\mathbf{x} \rightarrow \mathbf{x}' = (x'_1, \ldots, x'_d, x'_{d+1}) \quad x'_i = \frac{x_i}{\ell\sqrt{d}} \; (i = 1, \ldots, d) \quad x'_{d+1} = 1 - \sum_{i=1}^{d} x_i'^2$$

In the experiments, we tried to approximate one dimensional functions which are mapped onto a (hemi)circle using the technique shown above as :

$$F(x, y) = F(x) \quad |x| \leq 1, \quad y = \sqrt{1 - x^2}$$

Functions used in our experiments are as follows (Fig (9):

- $F_1 = x$
- $F_2 = e^{-x^2} \cos(\frac{3}{4}\pi x)$
- $F_3 = e^{-x^2} \cos(\frac{3}{2}\pi x)$
- $F_4 = \frac{\sin(3\pi x)}{3\pi x}$
- $F_5 = e^{-x} \sin(5x)$

They were approximated with 2 dimensional MLP(sigmoids), constrained MLP, GRBF (Gaussians), and *Multiscale* Gaussians given below.

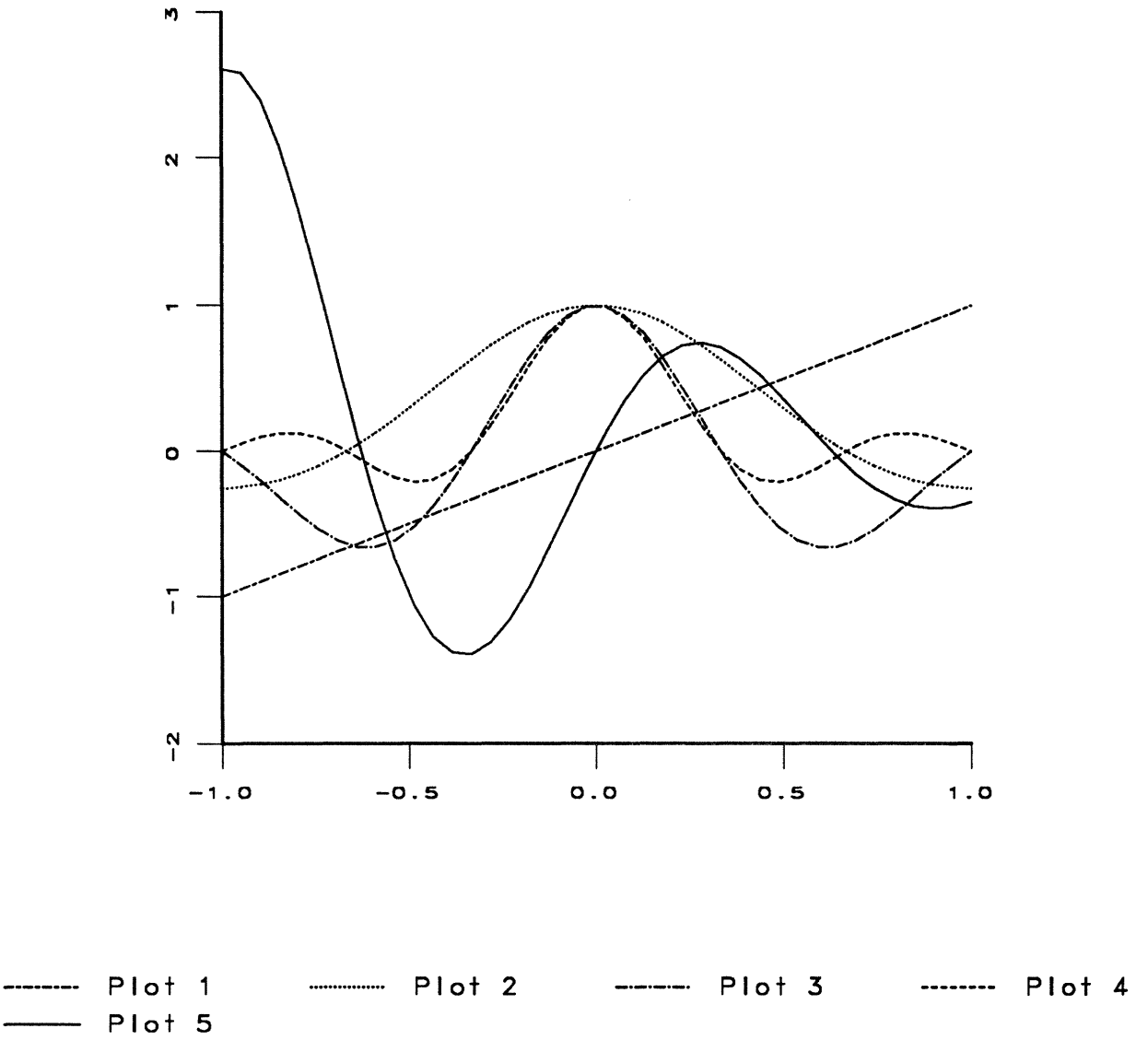


Figure 9: The test functions we used in our experiments: $F_1 = x$ (plot1), $F_2 = e^{-x^2}\cos(\frac{3}{4}\pi x)$ (plot2), $F_3 = e^{-x^2}\cos(\frac{3}{2}\pi x)$ (plot3), $F_4 = \frac{\sin(3\pi x)}{3\pi x}$ (plot4), $F_5 = e^{-x}\sin(5x)$ (plot5).

- MLP (Sigmoids)

$$f(x,y) = \sum_{\alpha=1}^{K} c_\alpha \sigma(\mathbf{w}_\alpha \cdot \mathbf{x} + \theta_\alpha),\ \sigma(u) = \frac{1}{1+e^{-u}}$$

- Constrained MLP(Sigmoid)

$$f(x,y) = \sum_{\alpha=1}^{K} c_\alpha \sigma(\mathbf{w}_\alpha \cdot \mathbf{x} + \theta_\alpha),\ \theta_\alpha = -(\frac{1}{4}\|\mathbf{w}_\alpha\|^2 + 1)$$

$$C(\theta_\alpha, \|\mathbf{w}_\alpha\|) = \exp(\frac{1}{4}(\|\mathbf{w}_\alpha\| - 2)^2) \geq 1$$

- GRBF (Gaussians)

$$f(x,y) = \sum_{\alpha=1}^{K} c_\alpha \exp(-\|\mathbf{x} - \mathbf{t}_\alpha\|^2)$$

- *Multiscale* GRBF

$$f(x,y) = \sum_{\alpha=1}^{K} \sum_{\beta=1}^{m} c_{\alpha\beta} \exp(-\mu_{\alpha\beta}^2 \|\mathbf{x} - \mathbf{t}_\alpha\|^2)$$

Approximation performances were compared according to normalized L_2 and L_∞ measured on both training set and evaluation set as follows.

$$L_2 = \frac{\sum_{i=1}^{N}(F(\mathbf{x}_i) - f(\mathbf{x}_i))^2}{\sum_{i=1}^{N} F(\mathbf{x}_i)^2},\quad L_\infty = \frac{\max_{i=1}^{N} |F(\mathbf{x}_i) - f(\mathbf{x}_i)|}{\max_{i=1}^{N} |F(\mathbf{x}_i)|}$$

$$L_2' = \frac{\sum_{p=1}^{M}(F(\mathbf{x}_p) - f(\mathbf{x}_p))^2}{\sum_{p=1}^{M} F(\mathbf{x}_p)^2},\quad L_\infty' = \frac{\max_{p=1}^{M} |F(\mathbf{x}_p) - f(\mathbf{x}_p)|}{\max_{p=1}^{M} |F(\mathbf{x}_p)|},$$

where $\{\mathbf{x}_i\}_{i=1}^{N}$, $\{\mathbf{x}_p\}_{p=1}^{M}$ are training set and evaluation set, respectively. In our experiments, these sets are randomly chosen and the number of points in the training set and evaluation set are $N = 20$ and $M = 100$ respectively.

In Table 2 we report the training and test errors, both in the L_2 and L_∞ norm, that we obtained applying the techniques described above to the set of test functions $F_1, \ldots, F_5$. In table 3 we report the value of $C(\theta_\alpha, \|\mathbf{w}_\alpha\|)$ and $\|\mathbf{w}_\alpha\|$ for the hidden units of the MLP network. These results show that condition (22) is not always satisfied. To see how target functions were approximated by MLP (sigmoids), we show in fig. (10), (11), (12), (13) and (14) the solutions obtained by our experiments.

5 Conclusions

This paper explores the relationships between MLPs and GRBF. Its main point is that under the condition of normalized inputs the two representations are very similar, though not identical. This is somewhat surprising, especially since MLPs anf RBF can be regarded as representative of two different ways of approximating functions based, respectively on scalar products and euclidean distances.

For normalized d-dimensional input vectors sigmoidal MLPs can always approximate essentially arbitrarily well a given GRBF (which has M less parameters, where M is the number of hidden units). The converse is true only for a certain range of the bias parameter in the sigmoidal unit. We have verified experimentally that in MLPs trained with backpropagation the bias parameters do not always respect this condition. Therefore MLPs generated by backpropagation cannot in general be approximated arbitrarily well by GRBFs with the same number of hidden units (and d less parameters). GRBFs that have 3 Gaussians per center and therefore the same number of parameters as a MLP network with the same number of hidden units yield a reasonable approximation of a given sigmoid MLP but, again, not for all parameters values. Within the framework of this paper, there seems to be no simple answer to the question of what is the relation between MLPs and *radial* HyperBF with full or diagonal $\mathbf{W}$ (a HyperBF network with diagonal $\mathbf{W}$ has the same number of parameters as a MLP network with the same number of hidden units). All this implies that MLPs network are – for normalized inputs – more *powerful* than GRBFs (and of course than RBF) networks with the same number of hidden units. Notice that the property of being more *powerful* is not necessarily an advantage here, since the number of parameters is larger (parameters to be learned are 1 per hidden unit in the

Function	Scheme	Units	Parameters	L_∞	L_2	L'_∞	L'_2
F_1	MLP	2	8	0.00632	2.0×10^{-5}	0.00853	2.2×10^{-5}
	Constrained MLP	2	6	0.00558	3.3×10^{-5}	0.00567	3.3×10^{-5}
	GRBF	2	6	0.0111	9.3×10^{-5}	0.0513	0.000212
	Multiscale GRBF	2-2	12	0.00724	5.2×10^{-5}	0.0243	8.0×10^{-5}
F_2	MLP	3	12	0.00786	2.6×10^{-5}	0.00895	5.3×10^{-5}
	Constrained MLP	3	9	0.00764	5.7×10^{-5}	0.00966	8.8×10^{-5}
	GRBF	3	9	0.0122	1.00×10^{-4}	0.0144	1.30×10^{-4}
	Multiscale GRBF	2-2	12	0.00579	1.6×10^{-5}	0.00740	2.4×10^{-5}
F_3	MLP	3	12	0.0106	5.4×10^{-5}	0.0236	7.6×10^{-5}
	Constrained MLP	3	9	0.0935	0.0102	0.158	0.0159
		4	12	0.0268	0.000257	0.134	0.00105
	GRBF	3	9	0.0588	0.00323	0.0645	0.00484
		4	12	0.0675	0.00351	0.0684	0.00503
	Multiscale GRBF	2-2	12	0.0122	7.5×10^{-5}	0.00941	6.4×10^{-5}
		3-2	18	0.0157	9.9×10^{-5}	0.140	9.2×10^{-5}
F_4	MLP	3	12	0.0130	0.000130	0.0153	0.000312
	Constrained MLP	3	9	0.340	0.182	0.392	0.231
		4	12	0.181	0.0512	0.212	0.0689
	GRBF	3	9	0.325	0.167	0.365	0.209
		4	12	0.122	0.0283	0.281	0.0391
	Multiscale GRBF	2-2	12	0.103	0.00725	0.157	0.0114
		3-2	18	0.00877	9.0×10^{-5}	0.00862	9.1×10^{-5}
F_5	MLP	3	12	0.00874	9.9×10^{-5}	0.0579	0.000719
	Constrained MLP	3	9	0.0513	0.00307	0.151	0.00743
		4	12	0.0375	0.00420	0.102	0.00810
	GRBF	3	9	0.200	0.0740	0.334	0.0917
		4	12	0.0668	0.0127	0.0872	0.0169
	Multiscale GRBF	2-2	12	0.0260	0.00112	0.0775	0.00210
		3-2	18	0.00821	9.3×10^{-5}	0.0661	0.000689

Table 2: Training and test errors, both in the L_2 and L_∞ norm, for the set of test functions $F_1, \ldots, F_5$.

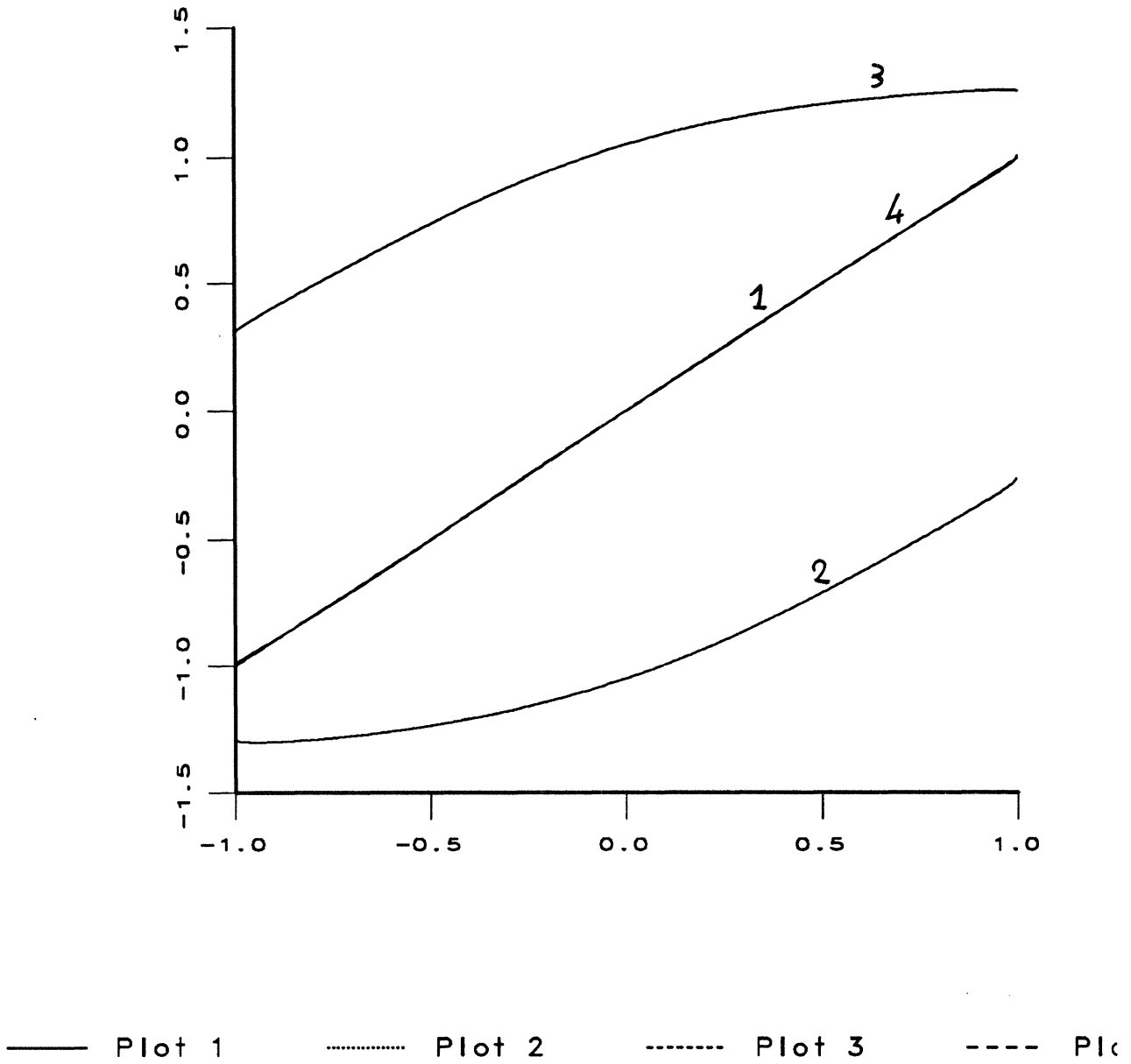


Figure 10: Approximation of the function F_1 by a MLP with 2 hidden units: (plot 1) $F_1 = x$, (plot2) basis 1, (plot 3) basis 2, (plot 4) result. Notice the complete overlapping between the original function (plot 1) and the MLP result (plot 4).

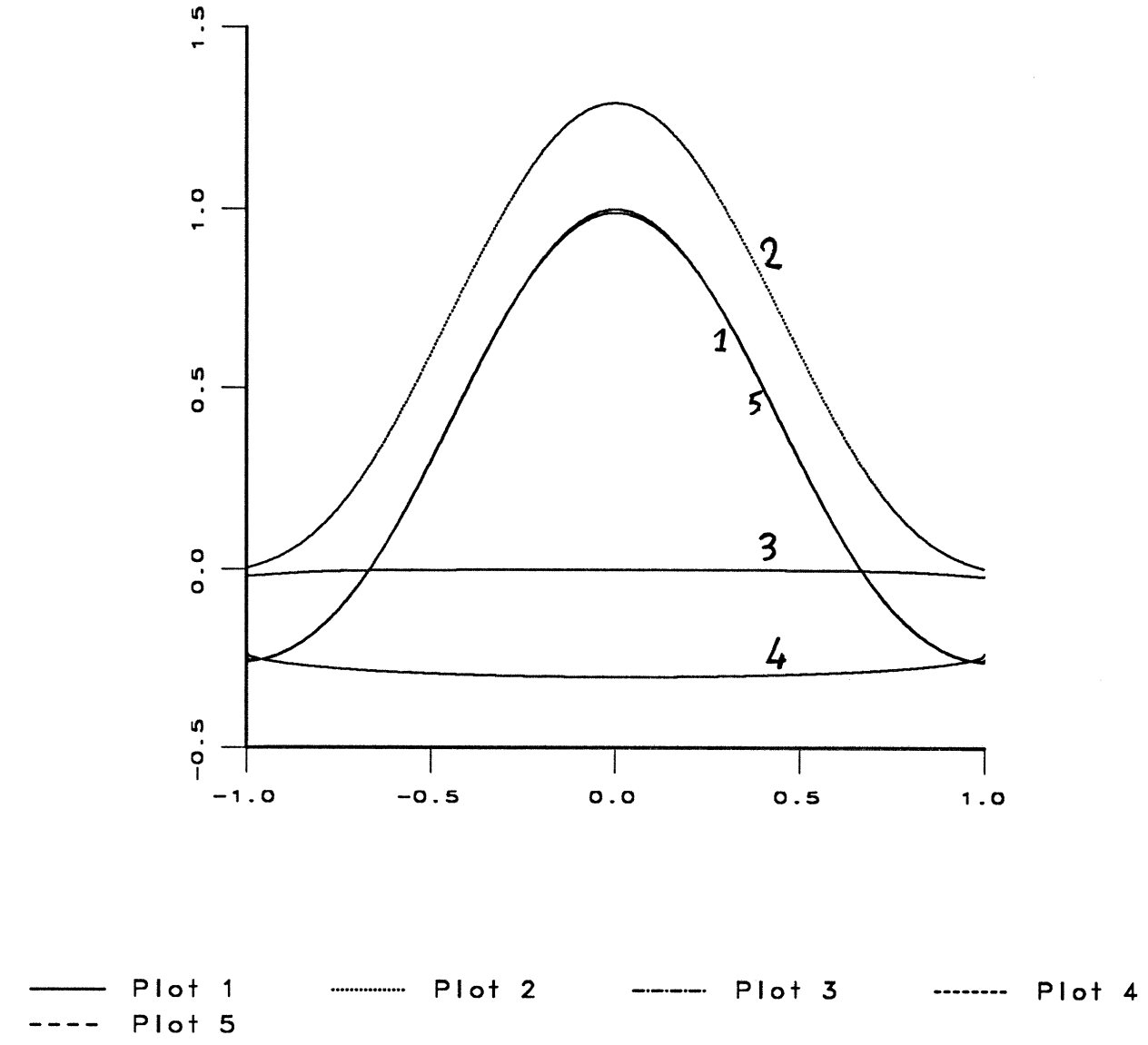


Figure 11: Approximation of the function F_2 by a MLP with 3 hidden units: (plot 1) $F_2 = e^{-x^2}\cos(\frac{3}{4}\pi x)$, (plot 2) basis 1, (plot 3) basis 2, (plot 4) basis 3, (plot 5) result. Notice the complete overlapping between the original function (plot 1) and the MLP result (plot 5).

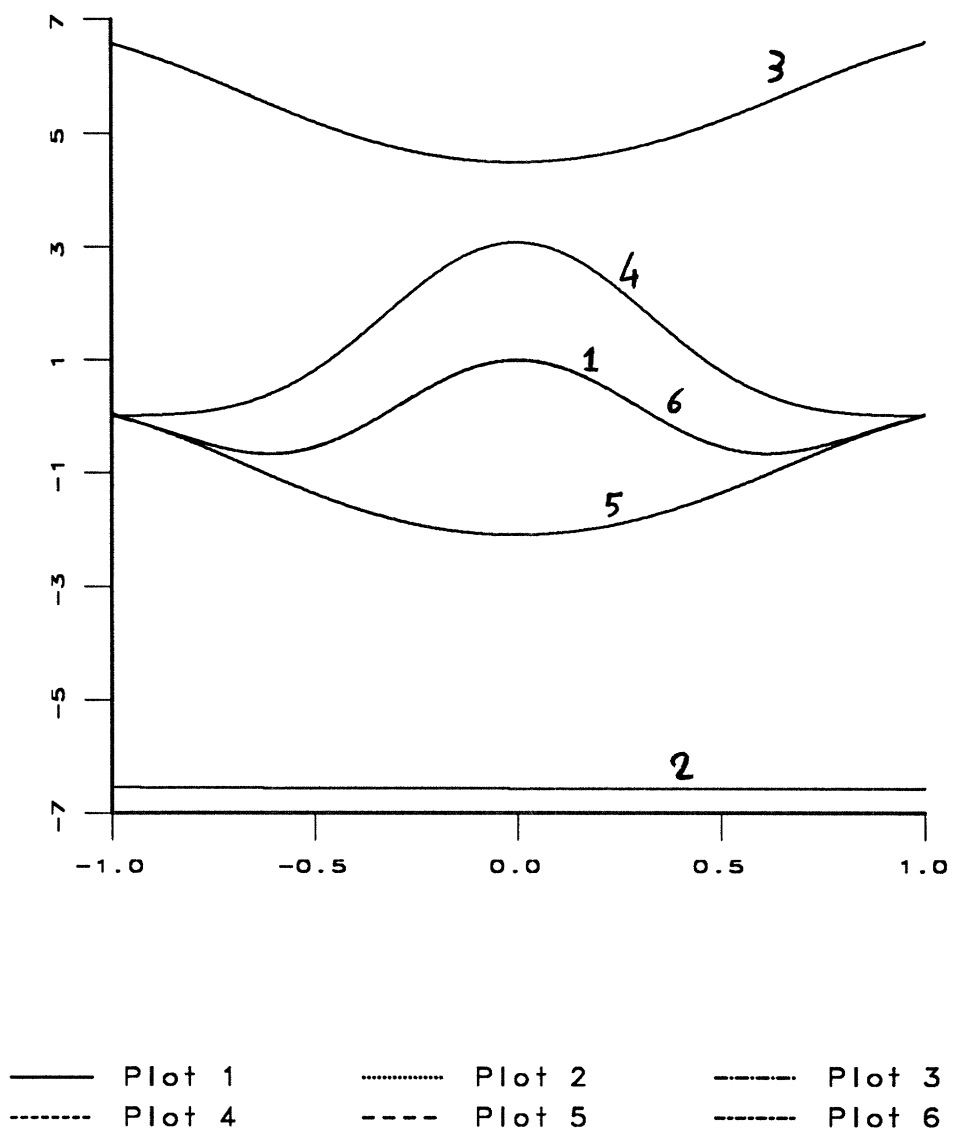


Figure 12: Approximation of the function F_3 by a MLP with 3 hidden units: (plot 1) $F_3 = e^{-x^2}\cos(\frac{3}{2}\pi x)$, (plot 2) basis 1, (plot 3) basis 2, (plot 4) basis 3, (plot 5) basis 1 + basis 2, (plot 6) result. Notice the complete overlapping between the original function (plot 1) and the MLP result (plot 6).

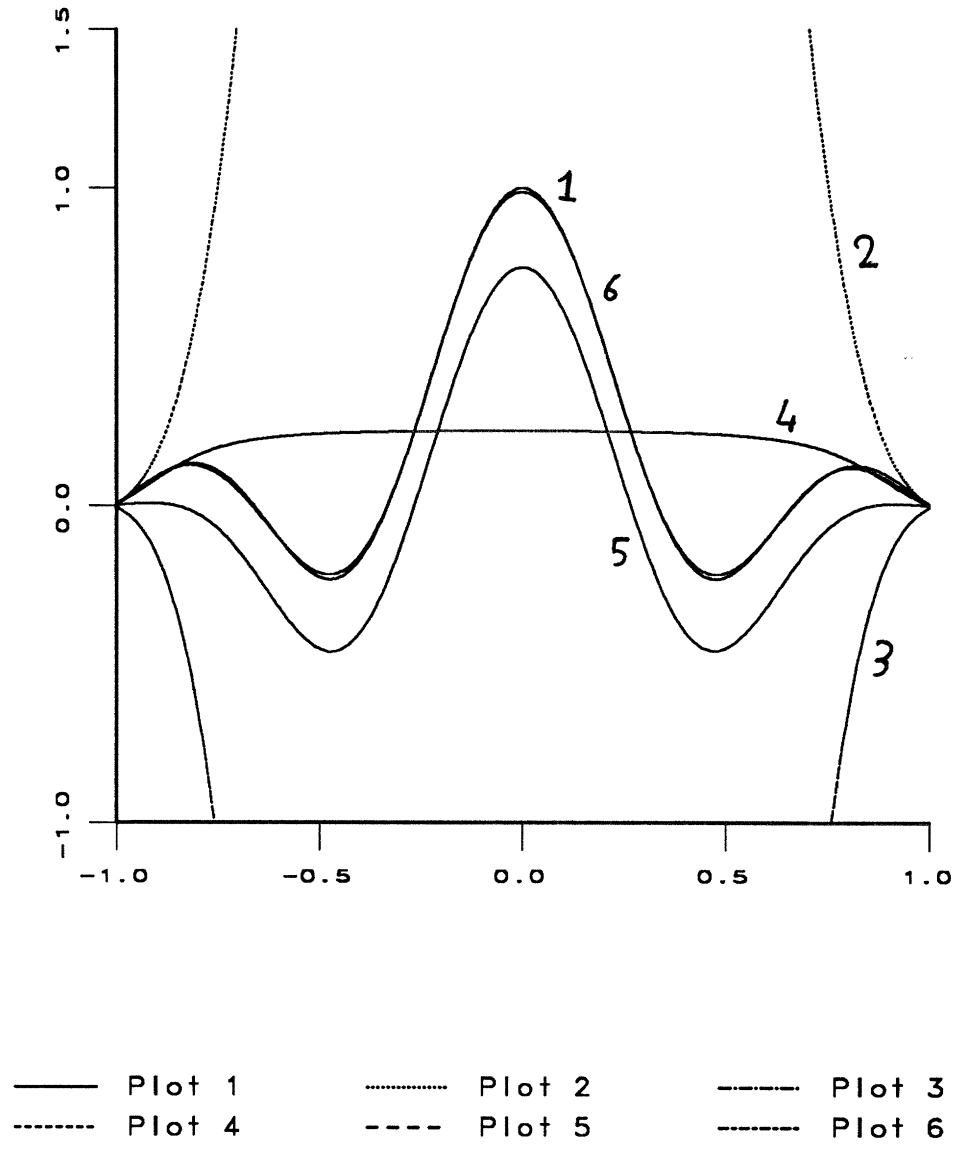


Figure 13: Approximation of the function F_4 by a MLP with 3 hidden units: (plot 1) $F_4 = \frac{\sin(3\pi x)}{3\pi x}$,(plot 2) basis 1, (plot 3) basis 2, (plot 4) basis 3, (plot 5) basis 1 + basis 2, (plot 6) result. Notice the complete overlapping between the original function (plot 1) and the MLP result (plot 6).

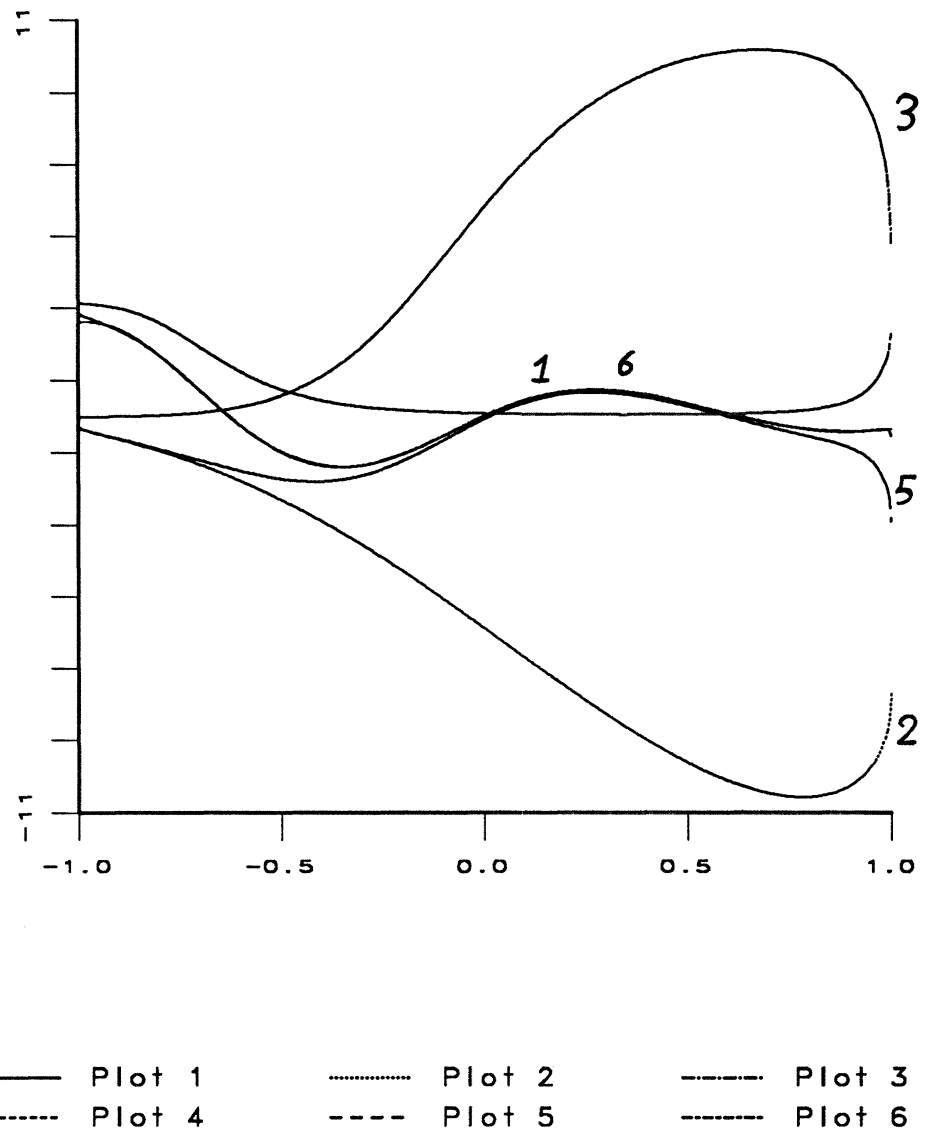


Figure 14: Approximation of the function F_5 by a MLP with 3 hidden units: (plot 1) $F_5 = e^{-x}\sin(5x)$, (plot 2) basis 1, (plot 3) basis 2, (plot 4) basis 3, (plot 5) basis 1 + basis 2, (plot 6) result. Notice the complete overlapping between the original function (plot 1) and the MLP result (plot 6).

	$(C(\theta_1, \|\mathbf{w}_1\|), \|\mathbf{w}_1\|)$	$(C(\theta_2, \|\mathbf{w}_2\|), \|\mathbf{w}_2\|)$	$(C(\theta_3, \|\mathbf{w}_3\|), \|\mathbf{w}_3\|)$
F_1	(0.0893,2.03)	(0.0509,2.14)	
F_2	(7.67,6.30)	(0.829,0.492)	$(1.4 \times 10^{-5}, 7.54)$
F_3	(11.4,10.5)	(0.00371,0.504)	$(1.15 \times 10^{-4}, 4.17)$
F_4	(7.48,8.01)	(1.60,8.63)	(0.0271,7.70)
F_5	(1.25,2.37)	(0.182,6.63)	$(1.2 \times 10^{-5}, 7.64)$

Table 3: The values of $C(\theta_\alpha, \|\mathbf{w}_\alpha\|)$ and $\|\mathbf{w}_\alpha\|$ corresponding to the hidden units of the MLP network ($\alpha = 1, 2, 3$).

case of RBFs, $d+1$ in the case of GRBF and $d+2$ in the case of MLPs) and actual performance should be considered (see Maruyama et al., 1991, for an experimental comparison between different approximation schemes).

How critical is the condition of normalized inputs for the above arguments? Mathematically there is no simple way of extending them to inputs that are not normalized. We have already mentioned the very recent results of Poggio and Girosi, which show that HyperBF and some set of ridge function approximation schemes are regularization networks corresponding to two different classes of stabilizers (and therefore different priors in the associated Bayesian interpretation). In the context of the arguments of this paper, it is interesting to remark that normalized inputs have been used in several well-known applications, with good reported performances. NETtalk is one example. In NETtalk the input layer consists of 7 groups of units and each group contains 29 units (i.e. number of units in the input layer is 203). Each group corresponds to a letter presented to NETtalk, and each units represents an alphabet (including period, blank, etc.). Therefore, when input letters are presented, only one unit among those of each group has the value "1" and the others have "0" as :

$$\mathbf{x} = (\underbrace{0, \cdots, 1, \cdots, 0}_{group\ 1}, \ldots, \underbrace{0, \cdots, 1, \cdots, 0}_{group\ 7})$$

Clearly, $\|\mathbf{x}\|^2 = const = 7$ always.

For normalized inputs it seems therefore that there is a strict relation, almost an equivalence, between the vector of the weights $\mathbf{w}$ in a MLP network

and the vector of the centers in the associated GRBF network. This fact has several potentially interesting consequences. The first one is that it may be possible to efficiently initialize the weights of a MLP. In particular, if we have a sufficient number of units (same as data size), we can design an *optimal* initial network, utilizing the properties of RBF and of the sigmoid-Gaussian quasi-equivalence. Several fast learning algorithms have been proposed for radial (basis) functions. They include Moody and Darken's method (Moody and Darken, 1989) based on the k-mean algorithm and Kohonen's LVQ. Our arguments imply that it should be possible to exploit these algorithms also for determining an initial structure of a MLP network.

The second point has to do with the common interpretation of the weights. T. Sejnowski (Sejnowski and Rosenberg, 1987) writes " In NETtalk, the intermediate code was semidistributed – around 15 % of the hidden units were used to represent each letter-to-sound correspondence. The vowels and the consonants were fairly well segregated, arguing for local coding at a gross population level (something seen in the brain) but distributed coding at the level of single units (also observed in the brain)." Our result seem to suggest that the "intermediate code" may often be a collection of appropriate "templates", in particular some of the examples.

References

[1] D. Braess. *Nonlinear Approximation Theory.* Springer-Verlag, Berlin, 1986.

[2] S.M. Carrol and B.W. Dickinson. Construction of neural nets using the Radon transform. In *Proceedings of the International Joint Conference on Neural Networks*, pages I–607–I–611, Washington D.C., June 1989. IEEE TAB Neural Network Committee.

[3] G. Cybenko. Approximation by superposition of a sigmoidal function. *Math. Control Systems Signals*, 2(4):303–314, 1989.

[4] D.L. Donoho and I.M. Johnstone. Projection-based approximation and a duality with kernel methods. *The Annals of Statistics*, 17(1):58–106, 1989.

[5] R.L. Eubank. *Spline Smoothing and Nonparametric Regression*, volume 90 of *Statistics, textbooks and monographs*. Marcel Dekker, Basel, 1988.

[6] J.H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–823, 1981.

[7] K. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.

[8] F. Girosi and T. Poggio. Networks and the best approximation property. *Biological Cybernetics*, 63:169–176, 1990.

[9] R.P. Gosselin. Nonlinear approximation using rapidly increasing functions. *Constr. Approx.*, 2:253–261, 1986.

[10] R. Hecht-Nielsen. Theory of backpropagation neural network. In *Proceedings of the International Joint Conference on Neural Networks*, pages I–593–I–605, Washington D.C., June 1989. IEEE TAB Neural Network Committee.

[11] C.R. Hobby and J.R. Rice. Approximation from a curve of functions. *Arch. Rat. Mech. Anal.*, 27:91–106, 1967.

[12] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[13] P.J. Huber. Projection pursuit. *The Annals of Statistics*, 13(2):435–475, 1985.

[14] C. A. Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constr. Approx.*, 2:11–22, 1986.

[15] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

[16] T. Poggio and F. Girosi. A theory of networks for approximation and learning. A.I. Memo No. 1140, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1989.

[17] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9), September 1990.

[18] T. Poggio and F. Girosi. Hyperbf: A powerful approximation technique for learning. In Patrick H. Winston and Sarah A. Shellard, editors, *Artificial Intelligence at MIT: Expanding Frontiers, Vol. 1.* M.I.T. Press, Cambridge, MA, 1990a.

[19] T. Poggio and F. Girosi. A theory of networks for learning. *Science*, 247:978–982, 1990a.

[20] M. J. D. Powell. Radial basis functions for multivariable interpolation: a review. In J. C. Mason and M. G. Cox, editors, *Algorithms for Approximation.* Clarendon Press, Oxford, 1987.

[21] J.R. Rice. *The approximation of functions, Vol. 1.* Addison-Wesley, Reading, MA, 1964.

[22] J.R. Rice. *The approximation of functions, Vol. 2.* Addison-Wesley, Reading, MA, 1969.

[23] L.L. Schumaker. *Spline functions: basic theory.* John Wiley and Sons, New York, 1981.

[24] T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce english text. *Complex Systems*, 1:145–168, 1987.

[25] M. Stinchcombe and H. White. Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. In *Proceedings of the International Joint Conference on Neural Networks*, pages I–607–I–611, Washington D.C., June 1989. IEEE TAB Neural Network Committee.

[26] G. Wahba. *Splines Models for Observational Data.* Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.