CAMBRIDGE UNIVERSITY ENGINEERING DEPARTMENT
TRUMPINGTON STREET
CAMBRIDGE CB2 1PZ

Average-Case Learning Curves for Radial Basis Function Networks

Sean B. Holden[1] and Mahesan Niranjan[2]

CUED/F-INFENG/TR.212 1995

May 4, 1995

[1]Present address: Department of Computer Science, University College London, Gower Street, London WC1E 6BT, U.K. Email: s.holden@cs.ucl.ac.uk.
[2]Email: niranjan@eng.cam.ac.uk.

Abstract

The application of statistical physics to the study of the learning curves of feedforward connectionist networks has, to date, been concerned mostly with networks that do not include hidden layers. Recent work has extended the theory to networks such as committee machines and parity machines; however these are not networks that are often used in practice and an important direction for current and future research is the extension of the theory to practical connectionist networks. In this paper we investigate the learning curves of a class of networks that has been widely, and successfully applied to practical problems: the Gaussian radial basis function networks (RBFNs). We address the problem of learning linear and nonlinear, realizable and unrealizable, target rules from noise-free training examples using a stochastic training algorithm. Expressions for the generalization error, defined as the expected error for a network with a given set of parameters, are derived for general Gaussian RBFNs, for which all parameters, including centres and spread parameters, are adaptable. Specializing to the case of RBFNs with fixed basis functions we then study the learning curves for these networks in the limit of high temperature.

Contents

1 Introduction

Statistical physics has made significant contributions to the study of *learning curves* for feed-forward connectionist networks. (It has also been applied with great success to the analysis of recurrent networks, although we do not consider the latter here. An excellent reference is Amit [1].) In this context, a learning curve can be defined as a graph of expected generalization performance against training set size; the term is defined in full below. Most of the research that has appeared to date has been concerned with networks that do not have hidden layers, and which are applied to the learning of similarly structured rules; for example, a perceptron learning another perceptron from examples. Given that the use of simple networks such as these in practice is quite rare, the study of more complex networks and rules constitutes one of the most important directions for continuing research in this area (Watkin *et al.* [2]). Most, if not all of the analysis to date for more complex networks has been confined to *committee machines* (see for example Schwarze *et al.* [3] and Barkai *et al.* [4]) and *parity machines* ([2] and references therein), which are again not networks that are applied to any significant extent in practice.

This paper presents results on the extension of the theory to the class of *Radial Basis Function Networks* (RBFNs), which have been used with considerable success in practical applications (see for example Renals and Rohwer [5] and Niranjan and Fallside [6]). The learning curves for these networks have already been studied in detail in a predominantly worst-case framework based on probably approximately correct (PAC) learning theory (Anthony and Biggs [7]); the results of these studies can be found in Holden and Rayner [8], Anthony and Holden [9], and Lee *et al.* [10].

This paper is organised as follows. Section 2 reviews radial basis function networks, defines the learning problems of interest and provides a brief review of the areas of statistical physics that are relevant to the paper. Section 3 addresses the first step in the analysis of a network using this formalism, namely to calculate the *generalization error* of the network for the tasks of interest. The generalization error, in this context, is the expected error, for randomly selected input vectors, of a network having a given set of parameters. Section 4 addresses the derivation of learning curves. It has not to date been possible to obtain completely general results in this part of the analysis, and consequently we investigate learning curves for the *high-temperature limit*. Section 5 discusses the results obtained and section 6 concludes the paper.

2 Preliminaries

This section reviews briefly the theory of RBFNs, and fully defines the learning problems of interest. It then provides a short introduction to the areas of statistical physics relevant to this paper. We do not provide a full account of the statistical physics relevant to the analysis of learning curves; such an account can be found in Seung *et al.* [11], Watkin *et al.* [2], Hertz *et al.* [12], and Landau and Lifshitz [13].

2.1 General Notation

We consider feedforward networks having p real-valued inputs, denoted by the vector $\mathbf{x} \in \mathbf{R}^p$ (vectors are assumed to be column vectors throughout this paper). Input vectors are assumed to be generated independently at random according to a probability density $p(\mathbf{x})$. The network computes a function $f(\mathbf{x}, \mathbf{w}) : \mathbf{R}^p \times \mathbf{R}^q \rightarrow \mathbf{R}$ that depends on a vector $\mathbf{w} \in \mathbf{R}^q$ of q real-valued parameters. Note that we address networks with real-valued outputs; networks of this type are typically applied to problems such as function interpolation, time-series prediction *etc.* In this paper we consider functions f of a particular form, which will be discussed in the next subsection. In general, networks having binary-valued inputs, outputs or weights can also be investigated using this formalism, however we do not address such networks here.

2.2 Radial Basis Function Networks

We focus on *Radial Basis Function Networks* (RBFNs). These networks were introduced by Broomhead and Lowe [14] and have a strong theoretical basis in interpolation theory (see, for example, Powell [15, 16]). A convenient way in which to write the function computed by a neural network with a single hidden layer of k units is,

$$f(\mathbf{x}, \mathbf{w}) = \alpha_0 + \sum_{i=1}^{k} \alpha_i \phi(\mathbf{x}, \boldsymbol{\beta}_i)$$

where $\mathbf{w}$ is the vector of all variable weights,

$$\mathbf{w}^T = [\ \alpha_0 \quad \cdots \quad \alpha_k \quad \boldsymbol{\beta}_1^T \quad \cdots \quad \boldsymbol{\beta}_k^T\ ]$$

and $\phi(.)$ is called a *basis function*. In this paper we assume that $\alpha_0 = 0$. For the case of RBFNs we use,

$$\phi(\mathbf{x}, \boldsymbol{\beta}_i) = \phi(\sigma_i, ||\mathbf{x} - \mathbf{u}_i||)$$

where the $\mathbf{u}_i$ are called the *centres* of the basis functions and $||.||$ is a norm which, in this paper, is always the Euclidean norm. Several forms are available for the function ϕ, however we will use the most popular, namely the Gaussian basis function,

$$\phi(\mathbf{x}, \boldsymbol{\beta}_i) = \exp\left[-\frac{1}{\sigma_i^2}||\mathbf{x} - \mathbf{u}_i||^2\right]. \tag{1}$$

The vectors $\boldsymbol{\beta}_i$ of parameters can be written $\boldsymbol{\beta}_i^T = [\ \sigma_i \quad \mathbf{u}_i^T\ ]$. For Gaussian basis functions σ_i controls the width of the ith basis function. Figure 1 shows the function computed by a typical network of this type, for which $p = 2$, $k = 50$, the parameters α_i and centres $\mathbf{u}_i$ have been selected randomly, and $\sigma_i = 1$ for $i = 1, 2, \ldots, k$.

In applying these networks in their full generality it is usual to adapt all the available parameters in response to training examples. In this case we have $q = k(p + 2)$ variable parameters in total. However, good results can also be obtained using simplified networks in which only the parameters α_i are adapted, and hence $q = k$. In the latter case we deal with networks that compute functions,

$$f(\mathbf{x}, \boldsymbol{\alpha}) = \frac{1}{\sqrt{k}} \sum_{i=1}^{k} \alpha_i \exp\left[-\frac{1}{\sigma_i^2}||\mathbf{x} - \mathbf{u}_i||^2\right]$$

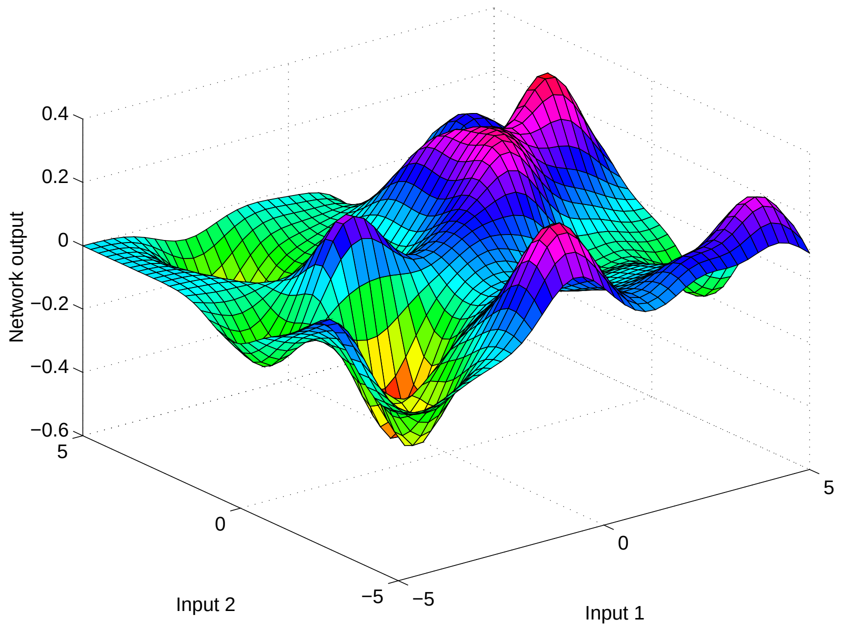

Figure 1: The function $f : \mathbf{R}^p \to \mathbf{R}$ computed by a typical RBFN where $p = 2$, $k = 50$, the parameters α_i and the centres $\mathbf{u}_i$ have been selected randomly, and $\sigma_i = 1$ for $i = 1, 2, \ldots, k$.

where the $\mathbf{u}_i$ are fixed and distinct, the σ_i are fixed, and $\boldsymbol{\alpha}^T = [\ \alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_k\ ]$ is the vector of adaptable weights. For the purposes of this paper, we assume that when only weights in $\boldsymbol{\alpha}$ are adapted, the centres and the σ_i parameters are chosen without reference to the training examples. For example, centres can be chosen randomly, but can not be chosen using clustering techniques (Moody and Darken [17]) or such that they coincide with a subset of the training examples [14].

An important observation can be made at this point. When only the parameters in $\boldsymbol{\alpha}$ are adaptable, the networks of interest operate by mapping input vectors of dimension p to a new space of dimension k, and mapping the resulting vectors to an output using a linear network having parameter vector $\boldsymbol{\alpha}$. Many results are available for such linear networks [2, 11], however such results usually apply to networks having Gaussian-distributed inputs, whereas in this case such inputs are clearly non-Gaussian, as the outputs of the basis functions of equation (1) are strictly positive. Some of the later results presented in this paper can therefore be regarded as an extension of earlier results to certain situations where inputs are not Gaussian-distributed.

2.3 Target Rules and the Learning Problem

We study the supervised learning of a rule from a sequence of P noise-free training examples. We let $P = \alpha q$ where q is the number of variable parameters used by the network, and we assume that training examples are generated using a *target rule* $f_T : \mathbf{R}^p \to \mathbf{R}$. The *training sequence* T_P is generated by drawing P inputs independently at random according to a fixed probability density $p(\mathbf{x})$ and forming,

$$T_P = ((\mathbf{x}_1, f_T(\mathbf{x}_1)), \ldots, (\mathbf{x}_P, f_T(\mathbf{x}_P))).$$

Throughout this paper we assume that $p(\mathbf{x})$ is the Gaussian density function,

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}} \exp\left[-\frac{1}{2}\mathbf{x}^T\mathbf{x}\right]. \tag{2}$$

We consider two types of target rule. In the first, f_T has the same form as the RBFN of interest, and the function f_T therefore has the form,

$$f_T(\mathbf{x}) = \frac{1}{\sqrt{k}} \sum_{i=1}^{k} \overline{\alpha}_i \exp\left[-\frac{1}{\overline{\sigma}_i^2}\|\mathbf{x} - \overline{\mathbf{u}}_i\|^2\right]$$

where $\overline{\alpha}_i$, $\overline{\sigma}_i$, and $\overline{\mathbf{u}}_i$ for $i = 1, 2, \ldots, k$ are fixed parameters that define f_T. Overlines are used in this manner throughout the paper to denote parameters associated with the target rule. Note that if $\overline{\mathbf{u}}_i = \mathbf{u}_i$ and $\overline{\sigma_i} = \sigma_i$ for $i = 1, 2, \ldots, k$ then the target rule is realizable; if the stated equalities do not hold then this may not be the case. The second type of target rule considered is a simple linear rule,

$$f_T(\mathbf{x}) = \frac{1}{\sqrt{p}}\overline{\gamma}^T\mathbf{x}$$

where $\overline{\gamma} \in \mathbf{R}^p$ is a fixed vector of parameters.

2.4 Statistical Physics

We begin with a few definitions. The *error* $\epsilon(\mathbf{x}, \mathbf{w})$ of a network on a given input is, as usual, a measure of the distance between its output and the output of the target rule for the corresponding input,

$$\epsilon(\mathbf{x}, \mathbf{w}) = \frac{1}{2}[f(\mathbf{x}, \mathbf{w}) - f_T(\mathbf{x})]^2.$$

Further, we use the usual measure for the *training error*, $E(\mathbf{w}, T_P)$, defined as,

$$E(\mathbf{w}, T_P) = \sum_{i=1}^{P} \epsilon(\mathbf{x}_i, \mathbf{w})$$

for the $\mathbf{x}_i$ in the training sequence T_P. We denote by,

$$\epsilon(\mathbf{w}) = \int \epsilon(\mathbf{x}, \mathbf{w}) p(\mathbf{x})\, d\mathbf{x} \tag{3}$$

the expected error of the network for a given weight vector, which is called the *generalization error* for that weight vector.

A central assumption in much of the treatment of learning curves using statistical physics is that the network is trained using a noisy dynamics leading at equilibrium to the Gibbs density (see [2, 11] and Levin *et al.* [18]),

$$p_G(\mathbf{w}) = \frac{1}{Z} p(\mathbf{w}) \exp[-\beta E(\mathbf{w}, T_P)]$$

where $\beta = 1/T$, T denotes the *temperature*, $p(\mathbf{w})$ is a prior density on the weights, and the relevant *partition function* is,

$$Z = \int p(\mathbf{w}) \exp[-\beta E(\mathbf{w}, T_P)] \, d\mathbf{w}.$$

The temperature, in this context, denotes the amount of noise present in the training dynamics. It is usual in this type of analysis to use the prior $p(\mathbf{w})$ to impose constraints on the ranges of the weights. We will not use this approach in this paper; this point is discussed in more detail in section 4.

Averages with respect to the Gibbs density, referred to as *thermal averages*, are as usual denoted by $\langle . \rangle_T$ and similarly, *quenched averages* over P-element training sequences are denoted by $\langle\langle . \rangle\rangle = \int \prod_{i=1}^{P} p(\mathbf{x}_i) d\mathbf{x}_i$ such that the quenched average of a random variable r is,

$$\langle\langle r(\mathbf{x}_1, \ldots, \mathbf{x}_P) \rangle\rangle = \int r(\mathbf{x}_1, \ldots, \mathbf{x}_P) p(\mathbf{x}_1) \cdots p(\mathbf{x}_P) d\mathbf{x}_1 \cdots d\mathbf{x}_P.$$

The primary quantity of interest in this paper is the *expected generalization error*,

$$\epsilon_g(P, T) = \langle\langle \, \langle \epsilon(\mathbf{w}) \rangle_T \rangle\rangle.$$

Plots of $\epsilon_g(P, T)$ against P for fixed T are called the *learning curves* for the network of interest. We will also be interested in the expected training error,

$$\epsilon_t(P, T) = \langle\langle \, \langle P^{-1} E(\mathbf{w}, T_P) \rangle_T \rangle\rangle.$$

It is straightforward to verify that,

$$\epsilon_t(P, T) = P^{-1} \frac{\partial(\beta F)}{\partial \beta} \tag{4}$$

where,

$$F = -T \langle\langle \ln Z \rangle\rangle \tag{5}$$

is the *free energy*.

2.5 The High-Temperature Limit

In this paper we will use a simplified version of the full theory known as the *high-temperature limit* (Seung *et al.* [11]). In the high-T limit we let $T \to \infty$ and $\alpha \to \infty$ while $\alpha\beta = $ constant. In this case the training error $E(\mathbf{w}, T_P)$ can be replaced by its average value $\langle\langle E(\mathbf{w}, T_P) \rangle\rangle$ and the expected training and generalization errors become equal. Consequently, the partition function takes the form,

$$Z_0 = \int p(\mathbf{w}) \exp\left[-q\alpha\beta\epsilon(\mathbf{w})\right] \, d\mathbf{w}$$

and it is straightforward to show using equations (4) and (5) that,

$$\epsilon_g(P, T) = -\frac{1}{P} \frac{\partial}{\partial \beta} \ln Z_0 \tag{6}$$

The high-T limit has been used in the analysis of other types of network by Schwarze *et al.* [3] and by Sompolinsky *et al.* [19].

3 The Generalization Error $\epsilon(\mathbf{w})$

A central quantity in the analysis of neural networks using this approach, and the first thing that we must calculate, is the generalization error $\epsilon(\mathbf{w})$ defined in equation (3). This quantity can be derived for the networks and target rules of interest for cases in which all parameters in the network are variable ($\mathbf{u}_i$ and σ_i are *not* assumed to be fixed for $i = 1, 2, \ldots, k$ as described above), assuming that the density $p(\mathbf{x})$ is the Gaussian density of equation (2). Although later in this paper we specialize to the case of more restricted networks, we include the fully general derivation here as it is likely to prove useful in further work on the statistical physics of these networks.

3.1 Useful Integrals

We make extensive use in this paper of the standard identity,

$$\int \exp\left[-\frac{1}{2}\left(\mathbf{x}^T\mathbf{X}\mathbf{x} + \mathbf{x}^T\mathbf{y} + z\right)\right] d\mathbf{x} = (2\pi)^{p/2} \mid \mathbf{X} \mid^{-1/2} \exp\left[-\frac{1}{2}\left(z - \frac{\mathbf{y}^T\mathbf{X}^{-1}\mathbf{y}}{4}\right)\right]. \quad (7)$$

where $\mathbf{X}$ is a constant, symmetric matrix, $\mathbf{y}$ and z are constants, and p is the dimension of $\mathbf{x}$. A derivation of this result can be found in Godsill [20]. We also make use of the identity,

$$\begin{aligned} I = \int \mathbf{a}^T\mathbf{x} \exp\left[-\frac{1}{2}\left(\mathbf{x}^T\mathbf{X}\mathbf{x} + \mathbf{x}^T\mathbf{y} + z\right)\right] d\mathbf{x} &= -\frac{1}{2}(2\pi)^{p/2} \mid \mathbf{X} \mid^{-1/2} \left(\mathbf{a}^T\mathbf{X}^{-1}\mathbf{y}\right) \times \\ & \exp\left[-\frac{1}{2}\left(z - \frac{\mathbf{y}^T\mathbf{X}^{-1}\mathbf{y}}{4}\right)\right] \end{aligned} \quad (8)$$

where $\mathbf{X}$ is again a symmetric matrix and $\mathbf{a}$ is a further constant vector. This is not, as far as we are aware, a standard result. It can be derived from equation (7) using Leibnitz's rule (see Kaplan [21]) by noting that,

$$I = \sum_{i=1}^{p} a_i \int x_i \exp\left[-\frac{1}{2}\left(\mathbf{x}^T\mathbf{X}\mathbf{x} + \mathbf{x}^T\mathbf{y} + z\right)\right] d\mathbf{x}$$

and,

$$\frac{\partial}{\partial y_i}\left\{\exp\left[-\frac{1}{2}\left(\mathbf{x}^T\mathbf{X}\mathbf{x} + \mathbf{x}^T\mathbf{y} + z\right)\right]\right\} = -\frac{1}{2}x_i \exp\left[-\frac{1}{2}\left(\mathbf{x}^T\mathbf{X}\mathbf{x} + \mathbf{x}^T\mathbf{y} + z\right)\right].$$

Finally, we use the identity,

$$\int \mathbf{x}^T\mathbf{X}\mathbf{x} \exp[-\mathbf{x}^T\mathbf{Y}\mathbf{x}]\, d\mathbf{x} = \frac{\pi^{p/2}}{\mid \mathbf{Y} \mid^{1/2}} \frac{\mathrm{Trace}(\mathbf{Y}^{-1}\mathbf{X})}{2} \quad (9)$$

where $\mathbf{Y}$ is constant and non-singular. This result was derived by Ó Ruanaidh [22].

3.2 Nonlinear Target Rules

The network of interest computes the function,

$$f(\mathbf{x}, \mathbf{w}) = \frac{1}{\sqrt{k}} \sum_{i=1}^{k} \alpha_i \phi(\mathbf{x}, \boldsymbol{\beta}_i)$$

where ϕ is the Gaussian basis function of equation (1). In this case the target rule is,

$$f_T(\mathbf{x}) = \frac{1}{\sqrt{k}} \sum_{i=1}^{k} \overline{\alpha}_i \phi(\mathbf{x}, \overline{\boldsymbol{\beta}}_i)$$

for fixed, unknown weights $\overline{\alpha}_i$, $\overline{\boldsymbol{\beta}}_i$ where $i = 1, 2, \ldots, k$, and we can therefore write the error $\epsilon(\mathbf{x}, \mathbf{w})$ of a network with weight vector $\mathbf{w}^T = [\ \boldsymbol{\alpha}^T \quad \boldsymbol{\beta}_1^T \quad \cdots \quad \boldsymbol{\beta}_k^T\ ]$ and inputs $\mathbf{x}$ as,

$$\begin{aligned} \epsilon(\mathbf{x}, \mathbf{w}) &= \frac{1}{2}\left[f(\mathbf{x}, \mathbf{w}) - f_T(\mathbf{x})\right]^2 \\ &= \frac{1}{2k}\left[\sum_{i=1}^{k} \alpha_i \phi(\mathbf{x}, \boldsymbol{\beta}_i) - \sum_{i=1}^{k} \overline{\alpha}_i \phi(\mathbf{x}, \overline{\boldsymbol{\beta}}_i)\right]^2 . \end{aligned}$$

Note that in the interests of generality we regard for the time being the vectors $\boldsymbol{\beta}_i$ as containing variable weights. Introducing the notation $\sum_{ij} \equiv \sum_{i=1}^{k} \sum_{j=1}^{k}$ we can write,

$$\begin{aligned} \epsilon(\mathbf{x}, \mathbf{w}) &= \frac{1}{2k}\left[\sum_{ij} \alpha_i \alpha_j \phi(\mathbf{x}, \boldsymbol{\beta}_i)\phi(\mathbf{x}, \boldsymbol{\beta}_j) + \sum_{ij} \overline{\alpha}_i \overline{\alpha}_j \phi(\mathbf{x}, \overline{\boldsymbol{\beta}}_i)\phi(\mathbf{x}, \overline{\boldsymbol{\beta}}_j)\right. \\ &\quad \left. -2\sum_{ij} \alpha_i \overline{\alpha}_j \phi(\mathbf{x}, \boldsymbol{\beta}_i)\phi(\mathbf{x}, \overline{\boldsymbol{\beta}}_j)\right] \end{aligned}$$

and hence,

$$\begin{aligned} \epsilon(\mathbf{w}) &= \int \epsilon(\mathbf{x}, \mathbf{w})\, p(\mathbf{x})\, d\mathbf{x} \\ &= \frac{1}{2k}\sum_{ij} \alpha_i \alpha_j \int \phi(\mathbf{x}, \boldsymbol{\beta}_i)\phi(\mathbf{x}, \boldsymbol{\beta}_j)\, p(\mathbf{x})\, d\mathbf{x} + \frac{1}{2k}\sum_{ij} \overline{\alpha}_i \overline{\alpha}_j \int \phi(\mathbf{x}, \overline{\boldsymbol{\beta}}_i)\phi(\mathbf{x}, \overline{\boldsymbol{\beta}}_j)\, p(\mathbf{x})\, d\mathbf{x} \\ &\quad -\frac{1}{k}\sum_{ij} \alpha_i \overline{\alpha}_j \int \phi(\mathbf{x}, \boldsymbol{\beta}_i)\phi(\mathbf{x}, \overline{\boldsymbol{\beta}}_j)\, p(\mathbf{x})\, d\mathbf{x} \end{aligned} \tag{10}$$

and we therefore need to evaluate integrals of the form,

$$I(\mathbf{a}, \mathbf{b}) = \int \phi(\mathbf{x}, \mathbf{a})\phi(\mathbf{x}, \mathbf{b})\, p(\mathbf{x})\, d\mathbf{x} \tag{11}$$

for $\mathbf{a}, \mathbf{b} \in \mathbf{R}^{(p+1)}$. We assume that the Gaussian basis function,

$$\begin{aligned} \phi(\mathbf{x}, \mathbf{a}) &= \exp\left[-\frac{1}{\sigma_a^2}\|\mathbf{x} - \mathbf{u}_a\|^2\right] \\ &= \exp\left[-\frac{1}{\sigma_a^2}(\mathbf{x} - \mathbf{u}_a)^T(\mathbf{x} - \mathbf{u}_a)\right] \end{aligned} \tag{12}$$

is used, where $\mathbf{u}_a$ is the centre of the basis function and σ_a controls its width. The parameter vectors $\mathbf{a}$ and $\mathbf{b}$ are written as,

$$\begin{aligned}
\mathbf{a}^T &= [\ \sigma_a \quad \mathbf{u}_a^T\ ] \\
\mathbf{b}^T &= [\ \sigma_b \quad \mathbf{u}_b^T\ ].
\end{aligned}$$

Inserting (2) and (12) in (11) we obtain,

$$\begin{aligned}
I(\mathbf{a},\mathbf{b}) &= \frac{1}{(2\pi)^{p/2}}\int \exp\left[-\frac{1}{2}\left(\frac{2}{\sigma_a^2}(\mathbf{x}-\mathbf{u}_a)^T(\mathbf{x}-\mathbf{u}_a)+\frac{2}{\sigma_b^2}(\mathbf{x}-\mathbf{u}_b)^T(\mathbf{x}-\mathbf{u}_b)+\mathbf{x}^T\mathbf{x}\right)\right] d\mathbf{x} \\
&= \frac{1}{(2\pi)^{p/2}}\int \exp\left[-\frac{1}{2}\left(\mathbf{x}^T\mathbf{x}\left(\frac{2}{\sigma_a^2}+\frac{2}{\sigma_b^2}+1\right)+\mathbf{x}^T\left(-\frac{4}{\sigma_a^2}\mathbf{u}_a-\frac{4}{\sigma_b^2}\mathbf{u}_b\right)\right.\right. \\
&\quad \left.\left.+2\left(\frac{1}{\sigma_a^2}\mathbf{u}_a^T\mathbf{u}_a+\frac{1}{\sigma_b^2}\mathbf{u}_b^T\mathbf{u}_b\right)\right)\right] d\mathbf{x}
\end{aligned}$$

We now apply the identity stated in equation (7) in order to obtain,

$$\begin{aligned}
I(\mathbf{a},\mathbf{b}) &= \frac{1}{(1+(2/\sigma_a^2)+(2/\sigma_b^2))^{p/2}}\exp\left[-\frac{1}{2}\left(2\left(\frac{1}{\sigma_a^2}-\frac{2}{\sigma_a^4\,(1+(2/\sigma_a^2)+(2/\sigma_b^2))}\right)\mathbf{u}_a^T\mathbf{u}_a\right.\right. \\
&\quad +2\left(\frac{1}{\sigma_b^2}-\frac{2}{\sigma_b^4\,(1+(2/\sigma_a^2)+(2/\sigma_b^2))}\right)\mathbf{u}_b^T\mathbf{u}_b \\
&\quad \left.\left.-\frac{8}{\sigma_a^2\sigma_b^2\,(1+(2/\sigma_a^2)+(2/\sigma_b^2))}\mathbf{u}_a^T\mathbf{u}_b\right)\right]. \qquad (13)
\end{aligned}$$

Simplifying equations (10) and (13) we obtain the final result,

$$\begin{aligned}
\epsilon(\mathbf{w}) &= \int \epsilon(\mathbf{x},\mathbf{w})p(\mathbf{x})\,d\mathbf{x} \\
&= \frac{1}{2k}\sum_{ij}\alpha_i\alpha_j I(\mathbf{u}_i,\mathbf{u}_j,\sigma_i,\sigma_j)+\frac{1}{2k}\sum_{ij}\overline{\alpha}_i\overline{\alpha}_j I(\overline{\mathbf{u}}_i,\overline{\mathbf{u}}_j,\overline{\sigma}_i,\overline{\sigma}_j) \\
&\quad -\frac{1}{k}\sum_{ij}\alpha_i\overline{\alpha}_j I(\mathbf{u}_i,\overline{\mathbf{u}}_j,\sigma_i,\overline{\sigma}_j)
\end{aligned}$$

where,

$$\begin{aligned}
I(\mathbf{u}_i,\mathbf{u}_j,\sigma_i,\sigma_j) &= (\sigma')^{-p/2}\exp\left[\frac{4}{\sigma_i^2\sigma_j^2\sigma'}\mathbf{u}_i^T\mathbf{u}_j-\left(\frac{1}{\sigma_i^2}-\frac{2}{\sigma_i^4\sigma'}\right)\mathbf{u}_i^T\mathbf{u}_i\right. \\
&\quad \left.-\left(\frac{1}{\sigma_j^2}-\frac{2}{\sigma_j^4\sigma'}\right)\mathbf{u}_j^T\mathbf{u}_j\right]
\end{aligned}$$

and

$$\sigma' = 1+\frac{2}{\sigma_i^2}+\frac{2}{\sigma_j^2}.$$

This is a general expression that applies when all the parameters in the network are regarded as variable weights. Figures 2, 3, and 4 illustrate the form of $\epsilon(\mathbf{w})$ for a very simple network

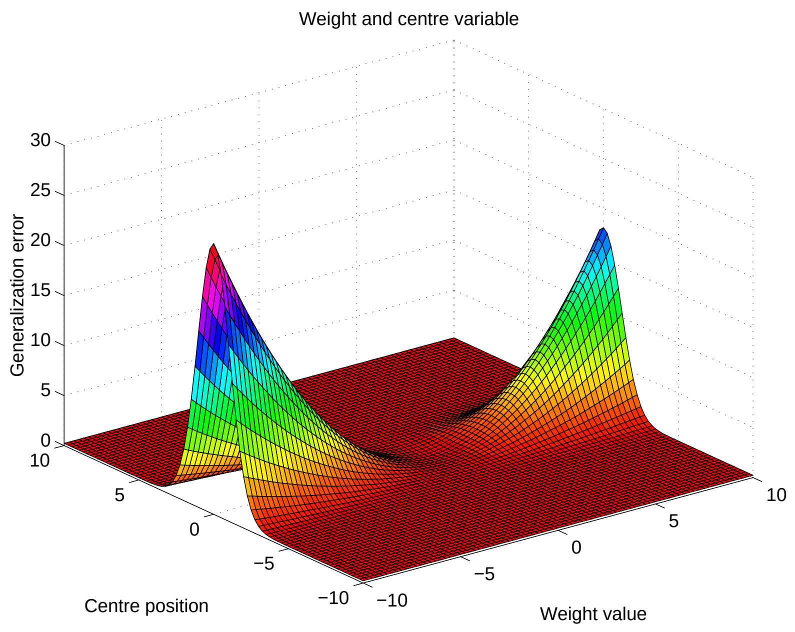

Figure 2: Generalization error $\epsilon(\mathbf{w})$ for the simple network described in the text for varying centre u and weight α.

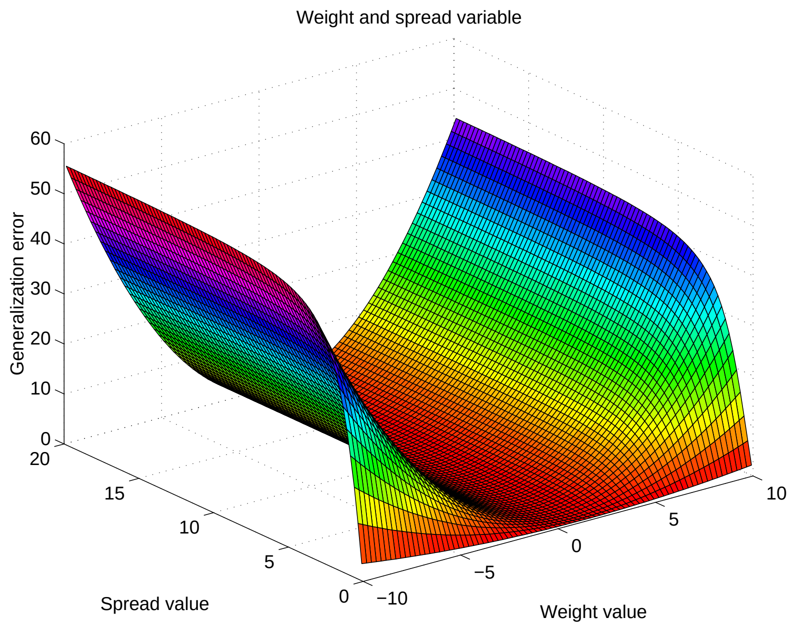

Figure 3: Generalization error $\epsilon(\mathbf{w})$ for the simple network described in the text for varying spread σ and weight α.

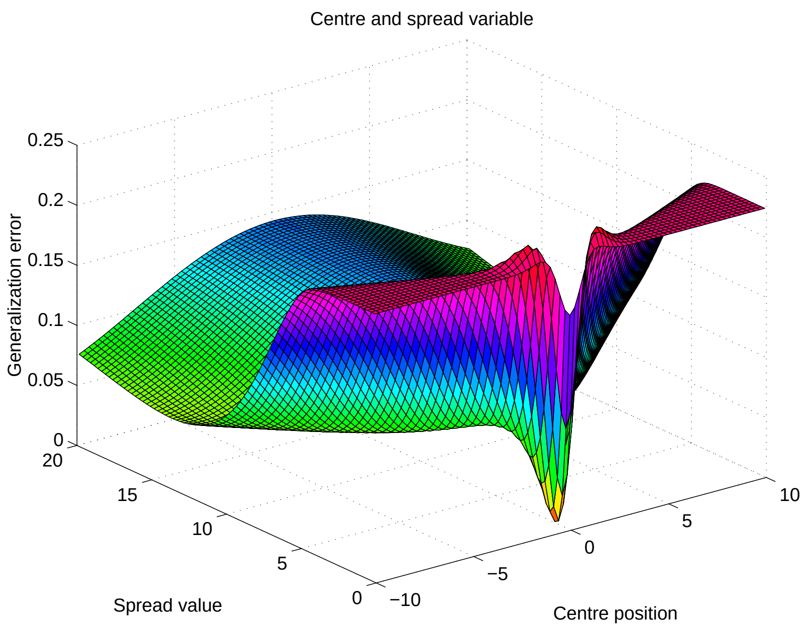

Figure 4: Generalization error $\epsilon(\mathbf{w})$ for the simple network described in the text for varying spread σ and centre u.

and target rule. The target rule in this case corresponds to a network having a single input, a single centre at the origin, and $\overline{\sigma} = \overline{\alpha} = 1$. The network has a single input, a single centre, a single weight α and a single spread parameter, and the three figures show the variation of $\epsilon(\mathbf{w})$ as two of the three network parameters are varied, the third parameter in each case taking the same value as the corresponding parameter for the target rule.

Consider now the restricted case in which only the parameters in $\boldsymbol{\alpha}$ are considered to be variable weights. Let $\mathbf{M}$ be the symmetric matrix having elements $M_{ij} = I(\mathbf{u}_i, \mathbf{u}_j, \sigma_i, \sigma_j)$. Also, let $\mathbf{N}$ be the matrix having elements $N_{ij} = I(\mathbf{u}_i, \overline{\mathbf{u}}_j, \sigma_i, \overline{\sigma}_j)$ and let $\overline{\boldsymbol{\alpha}}^T = [\ \overline{\alpha}_1 \quad \overline{\alpha}_2 \quad \cdots \quad \overline{\alpha}_k\ ]$. Then we can write,

$$\epsilon(\boldsymbol{\alpha}) = \frac{1}{2k}\boldsymbol{\alpha}^T\mathbf{M}\boldsymbol{\alpha} - \frac{1}{k}\boldsymbol{\alpha}^T\mathbf{n} + \frac{c_1}{k} \tag{14}$$

where $\mathbf{n} = \mathbf{N}\overline{\boldsymbol{\alpha}}$ and,

$$c_1 = \frac{1}{2}\sum_{ij}\overline{\alpha}_i\overline{\alpha}_j I(\overline{\mathbf{u}}_i, \overline{\mathbf{u}}_j, \overline{\sigma}_i, \overline{\sigma}_j).$$

The weight vector $\boldsymbol{\alpha}_0$ that minimizes $\epsilon(\boldsymbol{\alpha})$ can now be found by differentiating equation 14. We require that

$$\frac{\partial\epsilon(\boldsymbol{\alpha})}{\partial\boldsymbol{\alpha}} = \frac{1}{k}[\mathbf{M}\boldsymbol{\alpha} - \mathbf{n}] = 0$$

and so,

$$\boldsymbol{\alpha}_0 = \mathbf{M}^{-1}\mathbf{n}.$$

Consequently, the minimum value for $\epsilon(\boldsymbol{\alpha})$ is,

$$\epsilon_0 = \epsilon(\boldsymbol{\alpha}_0) = \frac{1}{k}\left[c_1 - \frac{1}{2}\mathbf{n}^T\mathbf{M}^{-1}\mathbf{n}\right]. \tag{15}$$

3.3 Linear Target Rules

Having obtained expressions for the generalization error of a RBFN learning a similar RBFN we now address the case of a RBFN learning a linear target rule. In this case the target rule is,

$$f_T(\mathbf{x}) = \frac{1}{\sqrt{p}}\overline{\gamma}^T\mathbf{x}$$

where $\overline{\gamma}$ is a fixed, unknown vector of parameters. We therefore have,

$$\begin{aligned}\epsilon(\mathbf{x},\mathbf{w}) &= \frac{1}{2}\left[\frac{1}{\sqrt{k}}\sum_{i=1}^{k}\alpha_i\phi(\mathbf{x},\boldsymbol{\beta}_i) - \frac{1}{\sqrt{p}}\overline{\gamma}^T\mathbf{x}\right]^2 \\ &= \frac{1}{2k}\sum_{ij}\alpha_i\alpha_j\phi(\mathbf{x},\boldsymbol{\beta}_i)\phi(\mathbf{x},\boldsymbol{\beta}_j) - \frac{1}{(kp)^{1/2}}\overline{\gamma}^T\mathbf{x}\sum_{i=1}^{k}\alpha_i\phi(\mathbf{x},\boldsymbol{\beta}_i) + \frac{1}{2p}\mathbf{x}^T\overline{\gamma}\,\overline{\gamma}^T\mathbf{x}.\end{aligned}$$

This leads to,

$$\begin{aligned}\epsilon(\mathbf{w}) &= \int \epsilon(\mathbf{x},\mathbf{w})p(\mathbf{x})\,d\mathbf{x} \\ &= \frac{1}{2k}\sum_{ij}\alpha_i\alpha_j\int \phi(\mathbf{x},\boldsymbol{\beta}_i)\phi(\mathbf{x},\boldsymbol{\beta}_j)p(\mathbf{x})\,d\mathbf{x} \\ &\quad +\frac{1}{2p}\int \mathbf{x}^T\overline{\gamma}\,\overline{\gamma}^T\mathbf{x}\,p(\mathbf{x})\,d\mathbf{x} \\ &\quad -\frac{1}{(kp)^{1/2}}\sum_{i=1}^{k}\alpha_i\int \overline{\gamma}^T\mathbf{x}\phi(\mathbf{x},\boldsymbol{\beta}_i)p(\mathbf{x})\,d\mathbf{x}.\end{aligned}$$

For Gaussian basis functions and Gaussian-distributed inputs, the first term in this expression is exactly as derived above. The second term can be evaluated using equation (9), and is given by,

$$\frac{1}{2p}\int \mathbf{x}^T\overline{\gamma}\,\overline{\gamma}^T\mathbf{x}\,p(\mathbf{x})\,d\mathbf{x} = \frac{1}{2p}\overline{\gamma}^T\overline{\gamma}.$$

The integral in the final term can be evaluated using equation (8), and is given by,

$$\int \overline{\gamma}^T\mathbf{x}\phi(\mathbf{x},\boldsymbol{\beta}_i)p(\mathbf{x})\,d\mathbf{x} = \frac{2}{((2/\sigma_i^2)+1)^{p/2}(2+\sigma_i^2)}\overline{\gamma}^T\mathbf{u}_i\exp\left[\mathbf{u}_i^T\mathbf{u}_i\left(\frac{2}{(2\sigma_i^2+\sigma_i^4)} - \frac{1}{\sigma_i^2}\right)\right].$$

For linear target rules we therefore obtain the final result,

$$\begin{aligned}\epsilon(\mathbf{w}) &= \int \epsilon(\mathbf{x},\mathbf{w})p(\mathbf{x})\,d\mathbf{x} \\ &= \frac{1}{2k}\sum_{ij}\alpha_i\alpha_j I(\mathbf{u}_i,\mathbf{u}_j,\sigma_i,\sigma_j) - \frac{1}{(kp)^{1/2}}\sum_{i=1}^{k}\alpha_i J(\mathbf{u}_i,\sigma_i,\overline{\gamma}) + \frac{c_2}{p} \qquad (16)\end{aligned}$$

where $I(\mathbf{u}_i,\mathbf{u}_j,\sigma_i,\sigma_j)$ is as defined above,

$$J(\mathbf{u}_i,\sigma_i,\overline{\gamma}) = \frac{2}{((2/\sigma_i^2)+1)^{p/2}(2+\sigma_i^2)}\overline{\gamma}^T\mathbf{u}_i\exp\left[\mathbf{u}_i^T\mathbf{u}_i\left(\frac{2}{(2\sigma_i^2+\sigma_i^4)} - \frac{1}{\sigma_i^2}\right)\right]$$

and,

$$c_2 = \frac{1}{2}\overline{\gamma}^T\overline{\gamma}.$$

Equation (16) again applies when all parameters, including centres and spread parameters, are variable. Let $\mathbf{o}$ be the vector having elements $o_i = J(\mathbf{u}_i, \sigma_i, \overline{\gamma})$. Then in the restricted case, when only the parameters in $\boldsymbol{\alpha}$ are considered to be variable weights, we can write,

$$\epsilon(\boldsymbol{\alpha}) = \frac{1}{2k}\boldsymbol{\alpha}^T\mathbf{M}\boldsymbol{\alpha} - \frac{1}{(kp)^{1/2}}\boldsymbol{\alpha}^T\mathbf{o} + \frac{c_2}{p}.$$

Consequently, we obtain,

$$\frac{\partial\epsilon(\boldsymbol{\alpha})}{\partial\boldsymbol{\alpha}} = \frac{1}{k}\mathbf{M}\boldsymbol{\alpha} - \frac{1}{(kp)^{1/2}}\mathbf{o}$$

such that,

$$\boldsymbol{\alpha}_0 = \frac{k}{(kp)^{1/2}}\mathbf{M}^{-1}\mathbf{o}$$

and

$$\epsilon_0 = \epsilon(\boldsymbol{\alpha}_0) = \frac{1}{p}\left[c_2 - \frac{1}{2}\mathbf{o}^T\mathbf{M}^{-1}\mathbf{o}\right].$$

4 Learning Curves

In this section we consider the learning curves for RBFNs learning the target rules considered in the previous section in the high-temperature limit. For this limit we have the partition function,

$$Z_0 = \int p(\boldsymbol{\alpha})\exp[-k\alpha\beta\epsilon(\boldsymbol{\alpha})]\, d\boldsymbol{\alpha}.$$

as discussed above and the quantity of interest is,

$$\epsilon_g(P,T) = -\frac{1}{P}\frac{\partial}{\partial\beta}\ln Z_0$$

(equation (6)). In order to calculate Z_0 we need to specify a prior density $p(\boldsymbol{\alpha})$ for the weight vectors $\boldsymbol{\alpha}$. The usual approach here is to use this prior density to constrain the ranges of the weights, and a density is usually specified that forces weight vectors to have a specified length (see for example [11]). This is not the approach that is in general applied in practical network design, and accordingly we will use a Gaussian prior,

$$p(\boldsymbol{\alpha}) = (2\pi)^{-k/2}\sigma_\alpha^{-k}\exp\left[-\frac{1}{2\sigma_\alpha^2}\boldsymbol{\alpha}^T\boldsymbol{\alpha}\right]. \tag{17}$$

We begin by considering nonlinear target rules. In this case we have,

$$\begin{aligned} Z_0 &= (2\pi)^{-k/2}\sigma_\alpha^{-k}\int \exp\left[-\alpha\beta\left(\frac{1}{2}\boldsymbol{\alpha}^T\mathbf{M}\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\mathbf{n} + c_1\right) - \frac{1}{2\sigma_\alpha^2}\boldsymbol{\alpha}^T\boldsymbol{\alpha}\right] d\boldsymbol{\alpha} \\ &= (2\pi)^{-k/2}\sigma_\alpha^{-k}\int \exp\left[-\frac{1}{2}\left(\boldsymbol{\alpha}^T\mathbf{N}\boldsymbol{\alpha} - 2\alpha\beta\boldsymbol{\alpha}^T\mathbf{n} + 2\alpha\beta c_1\right)\right] d\boldsymbol{\alpha} \end{aligned}$$

where,

$$\mathbf{N} = \alpha\beta\mathbf{M} + \sigma_\alpha^{-2}\mathbf{I}.$$

Applying the standard integral identity given in equation (7), and re-arranging we obtain,

$$Z_0 = \sigma_\alpha^{-k} \mid \mathbf{N} \mid^{-1/2} \exp\left[-\frac{1}{2}\left(2\alpha\beta c_1 - \alpha^2\beta^2\mathbf{n}^T\mathbf{N}^{-1}\mathbf{n}\right)\right].$$

The learning curve for the high-T limit is now given by,

$$\begin{aligned}\epsilon_g(P,T) &= -\frac{1}{P}\frac{\partial}{\partial\beta}\ln Z_0 \\ &= -\frac{1}{P}\left[\frac{1}{2}\frac{\partial}{\partial\beta}\left(\alpha^2\beta^2\mathbf{n}^T\mathbf{N}^{-1}\mathbf{n}\right) - \frac{1}{2}\frac{\partial}{\partial\beta}\ln \mid \mathbf{N} \mid -\alpha c_1\right].\end{aligned}$$

At this point the need to deal with the inverse and determinant of the matrix $\mathbf{N}$ becomes problematic, as we need to differentiate expressions containing these quantities with respect to β and it does not appear to be possible to do this analytically. We therefore consider the case in which σ_α is large—in effect the case in which we place no constraints on the ranges of the weights—such that,

$$\mathbf{N} \simeq \alpha\beta\mathbf{M}$$

and hence,

$$\begin{aligned}\epsilon_g(P,T) &= -\frac{1}{P}\left[\frac{1}{2}\alpha\mathbf{n}^T\mathbf{M}^{-1}\mathbf{n} - \alpha c_1 - \frac{k}{2\beta}\right] \\ &= \frac{1}{k}\left[c_1 - \frac{1}{2}\mathbf{n}^T\mathbf{M}^{-1}\mathbf{n}\right] + \frac{1}{2\alpha\beta}.\end{aligned}$$

Comparing this expression with equation (15) we obtain the final result,

$$\epsilon_g(P,T) = \epsilon_0 + \frac{1}{2\alpha\beta}. \tag{18}$$

For the case of a RBFN learning a linear target rule, a calculation exactly analogous to that given for nonlinear target rules leads again to equation (18).

5 Discussion

Until section 4 our analysis applied to completely general RBFNs for which all available parameters are adapted during training. The results obtained are therefore potentially useful in any further analysis of RBFNs using this formalism. In order to obtain actual learning curves it was necessary in section 4 to restrict our area of interest to more limited RBFNs, for which only the parameters of the vector $\boldsymbol{\alpha}$ are adapted during training. An area for future research is therefore the extension of the analysis of the previous section to fully general RBFNs.

A further area for further research is the analysis of RBFNs using more sophisticated and general techniques such as the *annealed approximation* and *replica method* (see Seung *et al.* [11]).

6 Conclusion

Statistical physics has made significant contributions to the study of learning curves for feed-forward neural networks, however most of the research to date applies to networks that are not used in practice to any significant extent. We have used the formalism provided by statistical physics to investigate the learning curves for Gaussian radial basis function networks, which have been used in many practical applications with excellent results. We have derived general expressions for the generalization error of a network with a given set of parameters learning nonlinear and linear, realizable and unrealizable rules in the absence of noise. We then obtained learning curves for a more restricted class of networks trained using a stochastic algorithm in the high-temperature limit. Two further possible areas for future research are the analysis of fully general networks in the high-temperature limit, and the analysis of radial basis function networks using more sophisticated techniques such as the annealed approximation and the replica method.

7 Acknowledgements

This research was supported by Science and Engineering Research Council research grant #GR/H16759. We thank Marc Mézard and Sebastian Seung for their useful comments on this work.

References

[1] Daniel J. Amit. *Modeling Brain Function. The World of Attractor Neural Networks.* Cambridge University Press, 1989.

[2] Timothy L. H. Watkin, Albrecht Rau, and Michael Biehl. The statistical mechanics of learning a rule. *Rev. Mod. Phys.*, 65(2):499–556, April 1993.

[3] H. Schwarze, M. Opper, and W. Kinzel. Generalization in a two-layer neural network. *Physical Review A*, 46(10):R6185–R6188, 1992.

[4] E. Barkai, D. Hansel, and H. Sompolinsky. Broken symmetries in multilayered perceptrons. *Physical Review A*, 45(6):4146–4161, March 1992.

[5] S. Renals and R. Rohwer. Phoneme classification experiments using radial basis functions. In *Proceedings of the International Joint Conference on Neural Networks*, pages I–461–I–467, June 1989.

[6] M. Niranjan and F. Fallside. Neural networks and radial basis functions in classifying static speech patterns. *Computer Speech and Language*, 4(3):275–289, July 1990. Also available as Cambridge University Engineering Department Report number CUED/F-INFENG/TR 22 (1988).

[7] Martin Anthony and Norman Biggs. *Computational Learning Theory.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1992.

[8] Sean B. Holden and Peter J. W. Rayner. Generalization and PAC learning: Some new results for the class of generalized single layer networks. *IEEE Transactions on Neural Networks*, 6(2):368–380, March 1995.

[9] Martin Anthony and Sean B. Holden. Quantifying generalization in linearly weighted neural networks. *Complex Systems*, 8:91–114, 1994.

[10] Wee Sun Lee, Peter L. Bartlett, and Robert C. Williamson. Lower bounds on the VC dimension of smoothly parameterized function classes. In *Proceedings of the 7th Annual Workshop on Computational Learning Theory*, pages 362–367. ACM Press, 1994.

[11] H. S. Seung, H. Sompolinsky, and N. Tishby. Statistical mechanics of learning from examples. *Physical Review A*, 45(8):6056–6091, April 1992.

[12] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation.* Santa Fe Institute Studies in the Sciences of Complexity. Addison-Wesley, 1991.

[13] L. D. Landau and E. M. Lifshitz. *Statistical Physics.* Course of Theoretical Physics. Pergamon Press, third edition, 1980.

[14] D. S. Broomhead and David Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[15] M. J. D. Powell. Radial basis functions for multivariable interpolation: A review. Technical Report #DAMTP 1985/NA12, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, October 1985.

[16] M. J. D. Powell. The theory of radial basis function approximation in 1990. Technical Report #DAMTP 1990/NA11, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, December 1990.

[17] John Moody and Christian J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294, 1989.

[18] Esther Levin, Naftali Tishby, and Sara A. Solla. A statistical approach to learning and generalization in layered neural networks. *Proceedings of the IEEE*, 78(10):1568–1574, October 1990.

[19] H. Sompolinsky, N. Tishby, and H. S. Seung. Learning from examples in large neural networks. *Physical Review Letters*, 65(13):1683–1686, 1990.

[20] S. Godsill. *The Restoration of Degraded Audio Signals.* PhD thesis, Cambridge University Engineering Department, Trumpington Street, Cambridge CB2 1PZ, U.K., 1993.

[21] Wilfred Kaplan. *Advanced Mathematics for Engineers.* Addison-Wesley, 1981.

[22] J. J. K. Ó Ruanaidh and W. J. Fitzgerald. The restoration of digital audio recordings using the gibbs sampler. Technical Report CUED/F-INFENG/TR.153 (1993), Cambridge University Engineering Department, Trumpington Street, Cambridge CB2 1PZ, U.K., 1993.