Probabilistic Analysis of Learning in Artificial Neural Networks: The PAC Model and its Variants

Martin Anthony
Department of Mathematics, The London School of Economics and Political Science, Houghton Street, London WC2A 2AE, UK

Abstract

There are a number of mathematical approaches to the study of learning and generalization in artificial neural networks. Here we survey the 'probably approximately correct' (PAC) model of learning and some of its variants. These models provide a probabilistic framework for the discussion of generalization and learning. This survey concentrates on the sample complexity questions in these models; that is, the emphasis is on how many examples should be used for training. Computational complexity considerations are briefly discussed for the basic PAC model. Throughout, the importance of the Vapnik-Chervonenkis dimension is highlighted. Particular attention is devoted to describing how the probabilistic models apply in the context of neural network learning, both for networks with binary-valued output and for networks with real-valued output.

Contents

[0]This work is supported in part by the European Union through the ESPRIT 'Neurocolt' project.. Updates, corrections, and comments should be sent to Martin Anthony at m.anthony@lse.ac.uk.

Neural Computing Surveys 1, 1-47, 1997, http ://www.icsi.berkeley.edu/~jagota/NCS

1 Introduction

Artificial neural networks have proved to be useful in a diverse range of applications, such as financial prediction, robotics and pattern classification. Given this, it is important to be able to provide performance guarantees. One has to know how well a neural network will generalize from a certain amount of training data and how long its learning algorithm will take if trained on a large sample. We are therefore led to discuss issues of complexity. There are two important aspects of complexity in neural network learning. First, there is the issue of *sample complexity*: in many learning problems, training data is expensive and we should hope not to need too much of it. Secondly, there is *computational complexity*: a neural network which takes an hour to train may be of no practical use in complex financial prediction problems. It is important that both the amount of training data required for a prescribed level of performance and the running time of the learning algorithm in learning from this data do not increase too dramatically as the 'difficulty' of the learning problem increases. Such issues have been formalised and investigated over the past decade within the field of 'computational learning theory'. In this article, I shall describe one popular framework for discussing such problems. This is the probabilistic framework which has become known as the 'probably approximately correct', or PAC, model of learning. Of course, there are many other mathematical frameworks in which to discuss learning and generalisation (see, for instance [118]), and I make no claim that the framework discussed here is superior to others discussed elsewhere.

I do not survey here the whole area of the PAC model and its important variants. I have placed emphasis on those topics I find to be of most interest and, consequently, there is more discussion of sample complexity than of computational complexity. There are now a number of books dealing with probabilistic models of learning: the interested reader might consult [115, 114, 95, 73, 90, 40, 10] for further information.

The first part of this work concerns the basic PAC model, applicable for classification problems. The second part concerns extensions of the basic PAC model, such as those relevant to neural networks with real-valued outputs. The PAC theory is useful for a number of non-neural learning problems, such as the inference of boolean functions. Therefore, while aiming to keep the discussion pertinent to neural networks, I have tried also to retain the generality of the theory.

PART 1: THE BASIC PAC MODEL

2 The Basic PAC Model of Learning

In this section, we describe the basic 'probably approximately correct' (PAC) model of learning, which arises from the work of Valiant [113, 112] and Vapnik and Chervonenkis [116, 114]. This model is applicable to neural networks with one output unit which outputs either the value 0 or 1. Thus, it applies to neural network *classification* problems. In the PAC model as it applies to neural networks, it is assumed that the neural network receives a stream of *examples*, each labeled with the value of a particular *target function* (or *target concept*) $t : X \to \{0, 1\}$ on that example. The set X here is the set of all possible inputs to the neural network, which may, for instance be $\{0, 1\}^n$ or $\mathbf{R}^n$ if the network has n input units. The target function is to be thought of as the function which is being 'learned' and it is assumed that this is one of a set C of possible target functions. Often, one would make the assumption that the target function is indeed computable by the neural network, but we need not make this assumption in the general model to be presented. A fundamental assumption of the PAC model is that these training examples are presented independently and at random according to some probability distribution on the set of all examples. For instance, it may be that $X = \{0, 1\}^n$ and each example is equally likely to be presented.

How should generalization be quantified? Suppose that the set of all possible examples is $X = \mathbf{R}^n$ or $X = \{0, 1\}^n$ where n is the number of inputs to the network, and suppose that the neural network is trying to 'learn' a target function t. A *training sample for* t of length m is a vector $\mathbf{x} = (x_1, x_2, \ldots, x_m) \in X^m$ of m examples together with the values $t(x_1), \ldots, t(x_m)$. Thus, more formally, the training sample for t determined by $\mathbf{x}$ is $\mathbf{s} \in (X \times \{0, 1\})^m$ given by

$$\mathbf{s} = ((x_1, t(x_1)), (x_2, t(x_2)), \ldots, (x_m, t(x_m))) .$$

An example $x \in X$ is said to be a *positive* (*negative*) example of t if $t(x) = 1$ ($t(x) = 0$). Thus a training sample is a list of examples, each classified as positive or negative examples of the target function. It is often convenient to refer to a sequence $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ of (unlabeled) examples as a *sample*. We shall denote by $S(m, t)$ the set of all training samples of length m for t. The learning algorithm accepts the training sample $\mathbf{s}$ and alters the state of the network in some way in response to the information provided by the sample. It is to be hoped that, in the resulting state, the function computed by the network is a better approximation to the target concept than the function computed beforehand.

Having set the scene for the PAC model in the context of neural network learning, we shall now take a slightly more general approach. In this general approach, there are two key sets of $\{0, 1\}$-valued functions defined on a set X of all possible examples. These are the *concept space* C, from which the target function comes, and the *hypothesis space* H, from which the learning algorithm chooses its output $L(\mathbf{s})$. In the context of a neural network learning problem, the hypothesis space H would be the set of all functions computable by the given neural network architecture. The relationship between C and H is crucial, as we shall see. Recall that $S(m, t)$ is the set of all training samples of length m for t. In our general learning framework, a (C, H)-*learning algorithm* is a function[1] L from the set $\bigcup_{t \in C, m \geq 1} S(m, t)$ of all possible training samples for functions in C, to the set H of all possible output hypotheses. Given as input a training sample $\mathbf{s} \in S(m, t)$ for some positive integer m and some target $t \in C$, the hypothesis $L(\mathbf{s})$ output by the learning algorithm is in H.

In order to measure the success of a learning algorithm, it is first of all important to be able to measure how close a given hypothesis is to the target function. Since there is assumed to be some probability distribution, μ, on the set of all examples[2], we may define the *error*, $\mathrm{er}_\mu(h, t)$, of a function h (with respect to t) to be the μ-probability that a further randomly chosen example is classified incorrectly by h. In other

[1] At this stage it is convenient to regard a learning 'algorithm' as a function rather than a proper algorithm. We shall address algorithmic questions later.

[2] Strictly speaking, there is some fixed σ-algebra Σ on the set X of examples and (X, Σ, μ) is a probability space. When X is finite, we take Σ to be the power set of X and when X is a subset of $\mathbf{R}^n$, we take Σ to be the Borel σ-algebra. It is assumed that all functions in C and H are Σ-measurable.

words,

$$\mathrm{er}_\mu(h,t) = \mu(\{x \in X : h(x) \neq t(x)\}).$$

When t is clear from the context, we often use the simpler notation $\mathrm{er}_\mu(h)$ for $\mathrm{er}_\mu(h,t)$. Now, we should hope that the error of $L(\mathbf{s})$ is 'usually' 'small'. Since each of the m examples in the training sample is drawn randomly and independently according to μ, the sample vector $\mathbf{x}$ is drawn randomly from X^m according to the product probability distribution μ^m. Thus, more formally, we want it to be true that with high μ^m-probability the sample $\mathbf{s}$ arising from $\mathbf{x}$ is such that the function $L(\mathbf{s})$ computed after training has small error with respect to t. This leads us to the following formal definition of PAC learning, first given by Valiant in 1984 [113][3]:

Definition 2.1 (PAC learning algorithm) *Let L be a (C,H)-learning algorithm. Then L is* probably approximately correct [4]*, or* PAC *if* for any *given ϵ, δ with $0 < \epsilon, \delta < 1$, there is a sample length $m_*(\delta,\epsilon)$ such that* for all $t \in C$ *and* for all *probability distributions μ on the set of examples, we have*

$$m \geq m_*(\delta,\epsilon) \Rightarrow \mu^m\left(\{\mathbf{s} \in S(m,t) : \mathrm{er}_\mu(L(\mathbf{s}),t) > \epsilon\}\right) < \delta.$$

In other words, the algorithm is PAC if for each *accuracy parameter* ϵ and each *confidence parameter* δ, there is $m_*(\delta,\epsilon)$ such that if a sample has length at least $m_*(\delta,\epsilon)$ then it is 'probably' the case that after training on that sample, the function output by the learning algorithm is 'approximately' correct (We should note that the product probability distribution μ^m is really defined not on subsets of $S(m,t)$ but on sets of underlying vectors $\mathbf{x} \in X^m$. However, this abuse of notation is convenient and is unambiguous: for a fixed t, there is a clear one-to-one correspondence between vectors $\mathbf{x} \in X^m$ and training samples $\mathbf{s} \in S(m,t)$.) Note that the probability distribution μ occurs twice in the definition: explicitly in the requirement that the μ^m-probability of a misleading sample be small and implicitly through the fact that the error of $L(\mathbf{s})$ is measured with reference to μ. The crucial feature of the definition is that we require the sufficient sample length $m_*(\delta,\epsilon)$ to be independent of μ and of t, depending only on δ and ϵ.

If L is a PAC (C,H)-learning algorithm, then the *sample complexity*, m_L, of L, is the function such that for $\delta, \epsilon \in (0,1)$, $m_L(\delta,\epsilon)$ is the least integer which suffices as a suitable $m_*(\delta,\epsilon)$ in the definition of PAC learning.

It is fairly easy to show that if C and H are the same set and this set is finite, then there is a PAC (H,H)-learning algorithm. (Henceforth, we shall refer to a (H,H)-learning algorithm simply as a learning algorithm (for H) when it is clear that the concept space is also H. Often, the terminology *proper learning algorithm* is used for such learning algorithms.) We say that the learning algorithm L is *consistent* if, given any training sample $\mathbf{s} = ((x_1,t(x_1)),(x_2,t(x_2)),\ldots,(x_m,t(x_m)))$, the functions $L(\mathbf{s})$ and t agree on x_i, for each i between 1 and m. Let $t \in H$ and suppose $h \in H$ has error $\epsilon_h \geq \epsilon$ with respect to t and the distribution μ. Then the probability (with respect to the product distribution μ^m) that h agrees with t on a random sample of length m is clearly at most $(1-\epsilon)^m$. This is at most $\exp(-\epsilon m)$, using a standard approximation. Thus, since there are certainly at most $|H|$ such functions h, the probability that *some* function in H has error at least ϵ *and* is consistent with a randomly chosen sample $\mathbf{s}$ is at most $|H|\exp(-\epsilon m)$. For any fixed positive δ, this probability is less than δ provided

$$m \geq m_*(\delta,\epsilon) = \frac{1}{\epsilon}\log\left(\frac{|H|}{\delta}\right),$$

a bound independent of both the distribution and the target function. This analysis shows that *any* consistent (H,H)-learning algorithm is PAC.

The above analysis shows that if H is finite then any consistent (H,H)-learning algorithm is PAC. However, note that the argument fails completely unless H is finite. It is not immediately clear that PAC learning is possible in such circumstances. In the next few sections, we present a theory which shows that, in many such cases, it is possible. However, for the moment, consider the following informal arguments.

[3] This is not exactly the definition given by Valiant.

[4] The terminology 'probably approximately correct' is due to Angluin [2].

If a particular example has not been seen in a large sample, the chances are that this example has low probability (with respect to μ) and therefore misclassification of that example contributes little to the error of the function computed after training. In other words, the penalty paid for misclassification of a particular example is its probability, and, very loosely speaking, the two occurrences of μ in the definition can therefore 'balance' or 'cancel' each other.

3 VC-Dimension and Growth Function

We now begin to present the mathematical theory which will enable us to ensure PAC learning is possible in cases where the concept and hypothesis spaces are not finite.

Suppose F is a set of $\{0,1\}$-valued functions defined on a set X, and let $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ be a sample of length m of examples from X. We define $\Pi_F(\mathbf{x})$, the *number of classifications of* $\mathbf{x}$ *by* F, to be the number of distinct vectors of the form

$$\mathbf{x}^*(f) = (f(x_1), f(x_2), \ldots, f(x_m)) ,$$

as f runs through all functions of F. Although F may be infinite, $F|_{\mathbf{x}}$, the set of functions obtained by restricting the functions of F to domain $E_{\mathbf{x}} = \{x_1, x_2, \ldots, x_m\}$, is finite and is of cardinality $\Pi_F(\mathbf{x})$. Note that for any sample $\mathbf{x}$ of length m, $\Pi_F(\mathbf{x}) \leq 2^m$. An important quantity, and one which turns out to be crucial in learning theory, is the maximum possible number of classifications by F of a sample of a given length. We define the *growth function* Π_F by

$$\Pi_F(m) = \max \{\Pi_F(\mathbf{x}) : \mathbf{x} \in X^m\} .$$

We have used the notation Π_F for both the number of classifications and the growth function, but this should cause no confusion.

We noted above that the number of possible classifications by F of a sample of length m is at most 2^m, this being the number of binary vectors of length m. We say that a sample $\mathbf{x}$ of length m is *shattered* by F, or that F *shatters* $\mathbf{x}$, if this maximum possible value is attained; that is, if F gives all possible classifications of $\mathbf{x}$. (We shall also find it useful to talk of a set of points, rather than a sample, being shattered. The notion is the same: the set is shattered if and only if a sample with those entries is shattered.) Note that if the examples in $\mathbf{x}$ are not distinct then $\mathbf{x}$ cannot be shattered by any F. When the examples are distinct, $\mathbf{x}$ is shattered by F if and only if for each subset S of $E_{\mathbf{x}}$, there is some function f_S in F such that for $1 \leq i \leq m$, $f_S(x_i) = 1 \Longleftrightarrow x_i \in S$.

Based on the intuitive notion that a set F of functions has high expressive power if it can achieve all possible classifications of a large set of examples, we use as a measure of this power the *Vapnik-Chervonenkis dimension*, or *VC-dimension*, of F, defined as follows. The VC-dimension of F is the maximum length of a sample shattered by F; if there is no such maximum, we say that the VC-dimension of F is infinite. Using the notation introduced above, we can say that the VC-dimension of F, denoted $\mathrm{VCdim}(F)$, is given by

$$\mathrm{VCdim}(F) = \max \{m : \Pi_F(m) = 2^m\} ,$$

where we take the maximum to be infinite if the set is unbounded. We state this definition formally, and in a slightly different form, for future reference.

Definition 3.1 (VC-dimension) *Let F be a set of functions from a set X to $\{0,1\}$. The* VC-dimension *of F is (infinite, or) the maximal size of a subset E of X such that for each $S \subseteq E$, there is $f_S \in F$ with $f_S(x) = 1$ if $x \in S$ and $f_S(x) = 0$ if $x \in E \setminus S$.*

When we have a binary-output neural network $\mathcal{N}$ with set of all possible inputs X, there is a corresponding hypothesis space, H, defined on example space X. This consists of all functions from X to $\{0,1\}$ which can be computed by the network in some state. As a set of $\{0,1\}$-valued functions, this hypothesis space has a VC-dimension. When X is clear from the context, we refer to this as the *VC-dimension of the neural network* and denote it simply by $\mathrm{VCdim}(\mathcal{N})$. When X is not clear, we shall use the notation $\mathrm{VCdim}(\mathcal{N}, X)$,

and refer to this quantity as the *VC-dimension of $\mathcal{N}$ on example set X*. (For example, we may wish to consider the VC-dimension of a particular network first on real inputs and then on only binary inputs, and these quantities might well be different.) We state this definition formally.

Definition 3.2 (VC-dimension of neural network) *Let $\mathcal{N}$ be a binary-output neural network capable of taking on a number of states*[5]*, and let Ω denote the set of all possible states. Suppose that X is the set of all possible inputs. Let $\mathcal{N}_\omega$ be the function from X to $\{0,1\}$ computed by $\mathcal{N}$ when its state is ω, and let*

$$H_{\mathcal{N}} = \{\mathcal{N}_\omega : \omega \in \Omega\}$$

be the set of functions computable by $\mathcal{N}$. Then the VC-dimension of $\mathcal{N}$ on example set X*, denoted* $\mathrm{VCdim}(\mathcal{N}, X)$ *(or* $\mathrm{VCdim}(\mathcal{N})$ *if X is clear), is defined to be the VC-dimension of $H_{\mathcal{N}}$.*

A result which is often useful is that if F is a finite set of $\{0,1\}$-valued functions then F has VC-dimension at most $\log |F|$. This follows from the observation that if d is the VC-dimension of F and $\mathbf{x} \in X^d$ is shattered by F, then $|F| \geq |F|_{\mathbf{x}}| = 2^d$. (Here, and throughout, log denotes logarithm to base 2 and ln denotes natural logarithm.)

The growth function $\Pi_F(m)$ of a space of finite VC-dimension is a measure of how many different classifications of an m-sample into positive and negative examples can be achieved by the functions of F, while the VC-dimension of F is the maximum value of m for which $\Pi_F(m) = 2^m$. Clearly these two quantities are related, because the VC-dimension is defined in terms of the growth function. But there is another, less obvious, relationship: the growth function $\Pi_F(m)$ can be bounded by a polynomial function of m, and the degree of the polynomial is the VC-dimension d of F. Explicitly, we have the following theorem [99, 107, 116], usually known as *Sauer's Lemma*. The second inequality is elementary—a proof was given by Blumer *et al.* [37].

Theorem 3.3 (Sauer's Lemma) *Let $d \geq 0$ and $m \geq 1$ be given integers and let F be a set of $\{0,1\}$-valued functions with* $\mathrm{VCdim}(F) = d \geq 1$*. Then*

$$\Pi_F(m) \leq \sum_{i=0}^{d} \binom{m}{i} < \left(\frac{em}{d}\right)^d,$$

where the second inequality holds for $m \geq d$. (Here, e is the base of natural logarithms.)

The first inequality of this theorem is tight; there are F of VC-dimension d having growth function $\Pi_F(m) = \sum_{i=0}^{d} \binom{m}{i}$.

We have motivated our discussion of VC-dimension by describing it as a measure of the expressive power of a set of functions. We shall see that the VC-dimensions of the concept space and hypothesis space turn out to be key parameters for quantifying the difficulty of PAC learning.

4 VC-Dimension and Linear Dimension

In this section, by way of example, and because it will prove useful, we present a result of Dudley which relates linear (vector-space) dimension to the VC-dimension.

A *homogeneous halfspace* of $\mathbf{R}^n$ is a subset of the form $\{x \in \mathbf{R}^n : \sum_{i=1}^{n} w_i x_i > 0\}$, for some constants w_i, $(1 \leq i \leq m)$. A *homogeneous linear threshold function* is the indicator, or characteristic, function of a homogeneous half-space. Note that such functions are precisely those computable by a simple perceptron—a single linear threshold neuron—with n inputs and weight vector w. We have the following characterisation of the sets which are shattered by homogeneous threshold functions.

[5] By a state of the network, we mean an assignment of values to the variable parameters of the network (that is, the weights and thresholds, and so on). Thus, given a neural network and a state, there is a corresponding function, namely the function computed by that neural net architecture when the state is fixed to be the given one.

Proposition 4.1 *A set $E = \{x_1, x_2, \ldots, x_k\}$ of points of $\mathbf{R}^n$ can be shattered by the set of homogeneous linear threshold functions if and only if the points of E are linearly independent.*

Proof Suppose that the points are linearly dependent. Then at least one is a linear combination of the others. Without loss of generality, suppose $x_1 = \sum_{i=2}^{k} \lambda_i x_i$ for some constants λ_i, $(1 \leq i \leq k)$, not all zero. Suppose the vector w is such that for $2 \leq j \leq k$, the inner product $\langle w, x_j \rangle$ is positive if $\lambda_j > 0$ and non-positive if $\lambda_j \leq 0$. Then $\langle w, x_1 \rangle = \sum_{i=2}^{k} \lambda_i \langle w, x_i \rangle > 0$. It follows that there is no homogeneous linear threshold function such that the following holds: x_1 is a negative example and, for $2 \leq j \leq k$, x_j is a positive example if and only if $\lambda_j > 0$. Thus the set E is not shattered.

For the converse, it suffices to prove the result when $k = n$. Let A be the matrix whose rows are the (linearly independent) row vectors $x_1, x_2, \ldots, x_n$ and let $v \in \{-1, 1\}^n$. Then A is nonsingular and so the matrix equation $Aw = v$ has a solution. The homogeneous linear threshold function t defined by this solution weight-vector w satisfies $t(x_j) = 1$ if and only if entry j of v is 1. Thus all possible classifications of the set of vectors can be realized, and the set is shattered. □

Recall that a set $\{f_1, f_2, \ldots, f_k\}$ of functions defined on a set X is *linearly dependent* if there are constants λ_i $(1 \leq i \leq k)$, not all zero, such that, *for all* $x \in X$,

$$\lambda_1 f_1(x) + \lambda_2 f_2(x) + \ldots + \lambda_k f_k(x) = 0;$$

that is, some non-trivial linear combination of the functions is the zero function on X. The following result is due to Dudley [42]; we present here a proof from [6].

Theorem 4.2 *Let $\mathcal{V}$ be a real vector space of real-valued functions defined on a set X. Suppose that $\mathcal{V}$ has linear dimension d. For any $f \in \mathcal{V}$, define the $\{0,1\}$-valued function f_+ on X by*

$$f_+(x) = \begin{cases} 1 & \text{if } f(x) > 0 \\ 0 & \text{if } f(x) \leq 0, \end{cases}$$

and let $\mathrm{pos}\mathcal{V} = \{f_+ : f \in \mathcal{V}\}$. *Then the VC-dimension of* $\mathrm{pos}\mathcal{V}$ *is d.*

Proof Suppose that $\{f_1, f_2, \ldots, f_d\}$ is a basis for $\mathcal{V}$ and, for $x \in X$, let

$$x^{\mathcal{V}} = (f_1(x), f_2(x), \ldots, f_d(x)).$$

Suppose the subset E of X is shattered by $\mathrm{pos}\mathcal{V}$. Then for each $S \subseteq E$ there is $f_S \in \mathcal{V}$ such that $f_S(x) > 0$ if $x \in S$ and $f_S(x) \leq 0$ if $x \in E \setminus S$. Now, since $\{f_1, \ldots, f_d\}$ is a basis of $\mathcal{V}$, there are constants w_i $(1 \leq i \leq d)$ such that $f_S = \sum_{i=1}^{d} w_i f_i$. Then the condition $f_S(x) > 0$ for $x \in S$ and $f(x) \leq 0$ for $x \in E \setminus S$ is equivalent to $\sum_{i=1}^{d} w_i f_i(x) > 0$ if $x \in S$ and $\sum_{i=1}^{d} w_i f_i(x) \leq 0$ if $x \in E \setminus S$. But this is true if and only if there is a homogeneous linear threshold function (given by the vector whose entries are the w_i) such that $t(x^{\mathcal{V}}) = 1$ if $x \in S$ and $t(x^{\mathcal{V}}) = 0$ if $x \in E \setminus S$. It follows, first, that the VC-dimension of $\mathrm{pos}\mathcal{V}$ is at most d. Secondly, it is equal to d if and only if there is a set $\{x_1^{\mathcal{V}}, \ldots, x_d^{\mathcal{V}}\}$ of linearly independent 'extended' vectors. Suppose that this is not so. Then the vector subspace spanned by the set $\{x^{\mathcal{V}} : x \in X\}$ is of dimension at most $d-1$ and therefore is contained in some hyperplane. Hence there are constants $\lambda_1, \lambda_2, \ldots, \lambda_d$, not all zero, such that for every $x \in X$, $\sum_{i=1}^{d} \lambda_i (x^{\mathcal{V}})_i = 0$; that is, $\sum_{i=1}^{d} \lambda_i f_i(x) = 0$ for all x. But this contradicts the fact that the functions $f_1, \ldots, f_d$ are linearly independent. It follows that the VC-dimension of $\mathrm{pos}\mathcal{V}$ is d. □

For a positive integer n, let P_n be the simple perceptron, or single linear threshold neuron, having n inputs and a single computation unit. The arcs carrying the inputs have real-valued weights $w_1, w_2, \ldots, w_n$ and there is a real threshold value θ at the active unit. As will be familiar, the weighted sum of the inputs is applied to the active unit and this unit outputs 1 if and only if the weighted sum exceeds the threshold value θ. We have the following useful result.

Theorem 4.3 *Let P_n be the simple perceptron on n inputs. Then*

$$\mathrm{VCdim}(P_n, \mathbf{R}^n) = n+1 \text{ and } \mathrm{VCdim}(P_n, \{0,1\}^n) = n+1.$$

Proof These results follow either by slightly adapting the proof of Proposition 4.1 or by taking the space $\mathcal{V}$ of Theorem 4.2 to be, respectively, the set of all affine functions from $\mathbf{R}^n$ to $\mathbf{R}$, or $\{0,1\}^n \to \{0,1\}$. Each of these has basis $\{f_0, f_1, \ldots, f_n\}$, where f_0 is the identically-1 function and $f_i(x) = x_i$. □

5 A Useful Probability Theorem

In this section, we prove a result of Vapnik [114]. This result and related ones have been central to the mathematical development of PAC learning theory. The paper of Vapnik and Chervonenkis [116] gave the first such results. Results of a similar form, but specifically for learning theory applications, were given by Blumer *et al.* [37], who were the first to highlight the importance of this area of probability for the theory of PAC learning. Their results were subsequently improved in [12, 105].

The precise result presented here is a slight improvement of a special case of a result of Vapnik, and the proof is based on one from [5].

Suppose that H is a set of functions from X to $\{0,1\}$, and let S be the cartesian product $X \times \{0,1\}$. For $\mathbf{s} = ((x_1,b_1),(x_2,b_2),\ldots,(x_m,b_m)) \in S^m$ and $h \in H$, the *observed error* of h on $\mathbf{s}$ is defined to be

$$\mathrm{er}_{\mathbf{s}}(h) = \frac{1}{m}\left|\{i : h(x_i) \neq b_i\}\right|.$$

Note that if $\mathbf{s} \in S(m,t)$ for some function t, so that $b_i = t(x_i)$ for each i, then this quantity is simply the proportion of training examples which h misclassifies. For $h \in H$, $E_h \subseteq S$ is defined as $E_h = \{(x,b) \in S : h(x) \neq b\}$. The following result is due to Vapnik.

Theorem 5.1 (Vapnik) *Let P be any[6] probability distribution on S. With the above notation, for $\eta > 0$,*

$$P^m\left\{\mathbf{s} \in S^m : \exists h \in H \text{ with } \frac{P(E_h) - \mathrm{er}_{\mathbf{s}}(h)}{\sqrt{P(E_h)}} > \eta\right\} \leq 4\,\Pi_H(2m)\exp\left(-\frac{1}{4}\eta^2 m\right).$$

Proof Let

$$Q = \left\{\mathbf{s} \in S^m : \exists h \in H \text{ with } \frac{P(E_h) - \mathrm{er}_{\mathbf{s}}(h)}{\sqrt{P(E_h)}} > \eta\right\},$$

$$R = \left\{\mathbf{rs} \in S^{2m} : \exists h \in H \text{ with } \mathrm{er}_{\mathbf{s}}(h) - \mathrm{er}_{\mathbf{r}}(h) > \eta\sqrt{\mathrm{er}_{\mathbf{rs}}(h)}\right\}.$$

Then it can be shown, as in the original proof of Vapnik, that $P^m(Q) \leq 4P^{2m}(R)$ for $m > 2/\eta^2$. This technique, relating the required probability to the probability of a 'sample-based' event, is known as 'symmetrization' [95]. The next part of the proof is to bound the probability of this latter event, using a technique known as 'combinatorial bounding'. Let Λ be the 'swapping' subgroup of the symmetric group of degree $2m$. This is the group generated by the transpositions $(i, m+i)$ for $1 \leq i \leq m$. Thus, Λ consists of those permutations of $\{1,2,\ldots,2m\}$ which swap i with $m+i$ for some numbers i in the range 1 to m. Λ has a natural group action on S^{2m}; given $\mathbf{z} = (z_1,\ldots,z_{2m})$ and $\tau \in \Lambda$, we define

$$\tau\mathbf{z} = (z_{\tau(1)},\ldots,z_{\tau(2m)}).$$

[6] As mentioned earlier, there is a fixed σ-algebra Σ on X, which is 2^X if X is finite, and the Borel algebra if X is an infinite subset of some $\mathbf{R}^n$. The probability measure P is defined on the product σ-algebra $\Sigma \times 2^{\{0,1\}}$. To ensure the measurability of the sets occuring in the result and its proof, one assumes that H satisfies certain conditions. These conditions, which may be found in [95], will hold in all cases considered here.

Denote by $\Theta(\mathbf{z})$ the number of permutations τ in Λ for which $\tau\mathbf{z}$ belongs to R, and define $\Theta(2m)$ to be the maximum over all $\mathbf{z} \in S^{2m}$ of this parameter. Because the action of Λ is measure preserving (with respect to the product measure P^{2m}), it can easily be shown (see, for example, [37, 116]) that

$$P^{2m}(R) \leq \frac{\Theta(2m)}{|\Lambda|}.$$

Fix $\mathbf{z} = ((x_1, b_1), (x_2, b_2), \ldots, (x_{2m}, b_{2m})) \in S^{2m}$ and let $\mathbf{x} = (x_1, x_2, \ldots, x_{2m})$. Suppose $h_1, h_2, \ldots, h_t \in H$ are such that $t = \Pi_H(\mathbf{x})$ and

$$\{\mathbf{x}^*(h_1), \mathbf{x}^*(h_2), \ldots, \mathbf{x}^*(h_t)\} = \{\mathbf{x}^*(h) : h \in H\}.$$

For each i between 1 and t, let $R^i \subseteq S^{2m}$ be given by

$$R^i = \left\{ \mathbf{rs} \in S^{2m} : \mathrm{er}_{\mathbf{s}}(h_i) - \mathrm{er}_{\mathbf{r}}(h_i) > \eta \sqrt{\mathrm{er}_{\mathbf{rs}}(h_i)} \right\}.$$

In order to bound $\Theta(\mathbf{z})$, we observe that if $\tau \in \Lambda$ and $\tau\mathbf{z} \in R$ then, for some i between 1 and t, $\tau\mathbf{z} \in R^i$. Hence $\Theta(\mathbf{z}) \leq \sum_{i=1}^{t} \Theta^i(\mathbf{z})$ where, for each i, $\Theta^i(\mathbf{z})$ is the number of permutations in Λ taking $\mathbf{z}$ into R^i. We now bound $\Theta^i(\mathbf{z})$. Following [55], for each $1 \leq j \leq 2m$, let $X_j = 1$ if $z_j \in E_{h_i}$ (that is, if $h_i(x_j) \neq b_j$) and $X_j = 0$ otherwise and, for $1 \leq j \leq m$, let Y_j be the random variable which equals $X_j - X_{m+j}$ with probability $1/2$ and $X_{m+j} - X_j$ with probability $1/2$. Let U be the uniform distribution on Λ. Then,

$$\frac{\Theta^i(\mathbf{z})}{|\Lambda|} = U \left\{ \tau \in \Lambda : \sum_{j=1}^{m} \left(X_{\tau^{-1}(m+j)} - X_{\tau^{-1}(j)} \right) > \eta \left(\frac{m}{2} \sum_{j=1}^{2m} X_j \right)^{1/2} \right\}$$

$$= \mathrm{Prob} \left\{ \sum_{j=1}^{m} Y_j > \eta \left(\frac{m}{2} \sum_{j=1}^{2m} X_j \right)^{1/2} \right\}.$$

By Hoeffding's inequality [62], this probability is bounded by

$$\exp\left(- \frac{\eta^2 m \sum_{j=1}^{2m} X_j}{4 \sum_{j=1}^{m} (X_j - X_{m+j})^2} \right) \leq \exp\left(-\frac{1}{4}\eta^2 m \right).$$

This holds for each $1 \leq i \leq t$. Therefore

$$\frac{\Theta(\mathbf{z})}{|\Lambda|} \leq \sum_{i=1}^{t} \frac{\Theta^i(\mathbf{z})}{|\Lambda|} \leq t \exp\left(-\frac{1}{4}\eta^2 m \right).$$

Since $t \leq \Pi_H(2m)$ and since $\mathbf{z}$ was arbitrary, we therefore obtain

$$P^{2m}(R) \leq \max_{\mathbf{z}} \frac{\Theta(\mathbf{z})}{|\Lambda|} \leq \Pi_H(2m) \exp\left(-\frac{1}{4}\eta^2 m \right).$$

Since $P^m(Q) \leq 4\, P^{2m}(R)$, this proves the theorem for $m > 2/\eta^2$. The bound of the theorem holds trivially for $m \leq 2/\eta^2$. □

The above result concerns a probability distribution P on $S = X \times \{0, 1\}$. For a function $h : X \to \{0, 1\}$, we define the *error of h with respect to P* to be $\mathrm{er}_P(h) = P(\{(x, b) : h(x) \neq b\})$; that is, $\mathrm{er}_P(h) = P(E_h)$. Up to now, we have discussed only the error of a function with respect to a probability measure μ on X and a target function t on X, but this notion is subsumed by the definition of error with respect to a distribution

on $S = X \times \{0,1\}$. For, consider any (measurable) function t from X to $\{0,1\}$ and any probability measure μ on X. Then it is fairly easy to construct a probability measure[7] P on S such that for any (measurable) subset A of X,

$$P\left(\{(x,t(x)) : x \in A\}\right) = \mu(A) \quad \text{and} \quad P\left(\{(x,y) : x \in A,\ y \neq t(x)\}\right) = 0.$$

The details may be found in [5, 19]. We shall see later that this more general framework of a probability distribution on $X \times \{0,1\}$ is useful. The following result is a consequence of Theorem 5.1, together with Sauer's Lemma. A better sample size bound, with smaller constants, improving a previous bound from [37], has been obtained in [5, 19]. However, the following result is adequate for our purposes.

Theorem 5.2 *Let H be a hypothesis space of $\{0,1\}$-valued functions defined on an input space X. Let P be any probability measure on $S = X \times \{0,1\}$, let $0 < \epsilon < 1$ and let $0 < \gamma \leq 1$. Then the probability (with respect to the product measure P^m) that, for $\mathbf{s} \in S^m$, there is some hypothesis from H such that*

$$\mathrm{er}_P(h) > \epsilon \quad \text{and} \quad \mathrm{er}_{\mathbf{s}}(h) \leq (1-\gamma)\mathrm{er}_P(h)$$

is at most

$$4\,\Pi_H(2m)\exp\left(-\frac{1}{4}\gamma^2\epsilon m\right).$$

Suppose H has finite VC-dimension d and that $0 < \gamma \leq 1$. Then, for any $\delta, \epsilon \in (0,1)$, there is $m_0 = m_0(\delta,\epsilon)$ such that if $m > m_0$ then, for $\mathbf{s} \in S^m$, with probability at least $1-\delta$ (with respect to the product measure P^m),

$$\mathrm{er}_{\mathbf{s}}(h) \leq (1-\gamma)\epsilon \Longrightarrow \mathrm{er}_P(h) \leq \epsilon.$$

A suitable value of m_0 is

$$m_0 = \frac{8}{\gamma^2\epsilon}\left(\ln\left(\frac{4}{\delta}\right) + d\ln\left(\frac{48}{\gamma^2\epsilon}\right)\right).$$

Proof Noting that $\mathrm{er}_P(h) = P(E_h)$, the first part of the result follows easily from Theorem 5.1 on taking $\eta = \gamma\sqrt{\epsilon}$. To obtain the sample length bound when H has finite VC-dimension d, we use Sauer's inequality, which tells us that for $2m \geq d$,

$$\Pi_H(2m) < \left(\frac{2em}{d}\right)^d.$$

With this, one can show that if $m > m_0$, then

$$4\left(\frac{2em}{d}\right)^d \exp\left(-\frac{1}{4}\gamma^2\epsilon m\right) \leq \delta,$$

from which the result follows. The details, which are omitted here, may be found in [5, 19]. □

6 PAC Learning and the VC-Dimension

In this section we show that basic PAC learning is characterised by the VC-dimension. More precisely, suppose that L is a (C,H)-learning algorithm and that $C \subseteq H$. We shall see that if L is consistent and H has finite VC-dimension then L is PAC. On the other hand, we shall also see that if L is PAC then C must have finite VC-dimension. We give bounds on the sample complexity of PAC learning algorithms in terms of these VC-dimensions. As a result of these, if one knows the VC-dimension of a neural network, one can give fairly precise bounds on the sample complexity of PAC learning on the network. For the moment, we proceed with the general theory. In the next section we deal specifically with neural networks.

[7] For $A \in \Sigma$ and $y \in \{0,1\}$, one defines $P(A \times \{0,1\}) = \mu\left(A \cap t^{-1}\left(\{y\}\right)\right)$. This defines a probability measure P on $\Sigma \times 2^{\{0,1\}}$ having the required properties.

Theorem 5.2 is a 'uniform convergence' result. We mentioned that Theorem 5.1 is a special case of a result of Vapnik. The most general form of that result is a statement about the rate of convergence of relative frequencies of events to their probabilities. In the form presented here, the events are the error sets E_h, the relative frequencies are the observed errors, and the true probabilities are the error of the hypotheses with respect to P. We shall find the result in the form stated in Theorem 5.2 useful later. For the moment, it suffices to note the following corollary. Recall that, for a PAC learning algorithm L, the sample complexity function m_L is such that $m_L(\delta, \epsilon)$ is the least integer which may be taken as $m_*(\delta, \epsilon)$ in the definition of PAC learning.

Theorem 6.1 *Suppose that the concept space C is contained in the hypothesis space*[8] *H. Suppose that H has finite VC-dimension. Let L be any* consistent *(C, H)-learning algorithm. Then L is PAC, and its sample complexity m_L satisfies the inequality*

$$m_L(\delta, \epsilon) \leq \frac{8}{\epsilon}\left(\ln\left(\frac{4}{\delta}\right) + \mathrm{VCdim}(H)\ln\left(\frac{48}{\epsilon}\right)\right).$$

Proof This follows immediately from the above results. We take $\gamma = 1$ in Theorem 5.2. For any fixed $t \in C$ and any probability measure μ on X, we take P to be the distribution on S corresponding to t and P, as described above. The only other observation needed is that, in this case, $P(E_h)$ is precisely $\mathrm{er}_\mu(h, t)$. □

The constants in the sample complexity bound of the above theorem can be improved; see [12, 105].

We now present a lower bound result, part of which is due to Ehrenfeucht *et al.* [43] and part of which is due to Blumer *et al.* [37]. This provides a lower bound on the sample complexity of *any* PAC (C, H)-learning algorithm when C has finite VC-dimension and is 'non-trivial'. (A result of this strength is not needed simply to show that finite VC-dimension of the concept space is necessary for PAC learning: a simpler result proving this may be found in [37]. But we wish also to demonstrate that there are lower bounds and upper bounds on the sample complexity of consistent PAC algorithms which do not differ too greatly.)

Theorem 6.2 *Let C be a concept space of VC-dimension at least 1 and consisting of at least three distinct concepts. Suppose that H is some hypothesis space, and that L is any PAC (C, H)-learning algorithm. Then the sample complexity of L satisfies*

$$m_L(\delta, \epsilon) > \max\left(\frac{\mathrm{VCdim}(C) - 1}{32\epsilon}, \frac{1}{\epsilon}\ln\left(\frac{1}{\delta}\right)\right),$$

for all $\epsilon \leq 1/8$ and $\delta \leq 1/100$.

Proof This proof is from [10]. In order to prove the first part of the result, we shall prove that there is some probability distribution μ and some $t \in C$ such that for any $\epsilon \leq 1/8$ and for any positive integer $m \leq (d-1)/32\epsilon$,

$$\mu^m\left\{\mathbf{s} \in S(m, t) : \mathrm{er}_\mu\left(L\left(\mathbf{s}, t\right)\right) \geq \epsilon\right\} \geq \frac{1}{100}.$$

Since C has VC-dimension d, there is a sample $\mathbf{z} = (z_0, z_1, \ldots, z_{d-1})$ of length d shattered by C. Let $E_{\mathbf{z}} = \{z_0, z_1, z_2, \ldots, z_r\}$, where $r = d - 1$. Define a probability distribution μ on X by

$$\mu\left(z_0\right) = 1 - 8\epsilon, \quad \mu\left(z_i\right) = \frac{8\epsilon}{r}\ (1 \leq i \leq r).$$

Then a randomly chosen sample $\mathbf{x}$ is, with probability one, a sample of examples from E. We therefore need consider only samples drawn from E, and we can regard the concept space C to be simply the (finite) space of all $\{0, 1\}$-valued functions defined on domain E: we make this assumption throughout the rest of this proof.

[8] What is really required is that for every $t \in C$, for every positive integer m, if $\mathbf{x} \in X^m$, then there is some $h \in H$ which agrees with t on $\mathbf{x}$.

Suppose that L is a PAC (C,H)-learning algorithm. For $\mathbf{x} \in X^m$ and $t \in C$, if $\mathbf{s} \in S(m,t)$ is the corresponding training sample, it shall be convenient to denote $L(\mathbf{s})$ by $L(\mathbf{x},t)$. Let C_0 be the set of functions c in C for which $c(z_0) = 0$ and let F be the set $\{z_1, z_2, \ldots, z_r\}$. Fix a particular sample $\mathbf{y} \in E^m$, and let l be the number of distinct elements of F appearing as examples in $\mathbf{y}$. Let $c \in C_0$ and let x be any one of the $(r-l)$ examples in F not appearing in $\mathbf{y}$. Now, C_0 shatters F, since C shatters $E = F \cup \{z_0\}$. Hence precisely half of the functions c' in C_0 satisfy $c'(x) = 1$ (and half of them satisfy $c'(x) = 0$).

For $x \in F$, let us define $\Delta_x(\mathbf{y}, c)$ to be 1 if $L(\mathbf{y}, c)$ and c disagree on x and 0 otherwise. Then $D(\mathbf{y}, c) = \sum_{x \in F} \Delta_x(\mathbf{y}, c)$ is the number of x in F for which $L(\mathbf{y}, c)$ and c disagree. By the above remarks,

$$\sum_{c \in C_0} D(\mathbf{y}, c) = \sum_{c \in C_0} \sum_{x \in F} \Delta_x(\mathbf{y}, c) = \sum_{x \in F} \sum_{c \in C_0} \Delta_x(\mathbf{y}, c) \geq \sum_{x \in F \setminus E_{\mathbf{y}}} \frac{1}{2}|C_0| = \frac{1}{2}(r-l)|C_0|.$$

If $l < r/2$ then $(r-l) > r/2$. Hence, noting that $\mathbf{y}$ is arbitrary in the above analysis, if S is the set of samples which contain fewer than $r/2$ distinct elements of F, then we have

$$D = \sum_{\mathbf{x} \in S} \sum_{c \in C_0} D(\mathbf{x}, c) > \frac{r}{4}|C_0||S|.$$

We can interchange the order of summation to obtain

$$D = \sum_{c \in C_0} \sum_{\mathbf{x} \in S} D(\mathbf{x}, c) > \frac{r}{4}|S||C_0|,$$

from which it follows that for some $t \in C_0$, $\sum_{\mathbf{x} \in S} D(\mathbf{x}, t) > \frac{r}{4}|S|$.

For any $\mathbf{x} \in S$, $D(\mathbf{x}, t) \leq r$, and so if N is the number of samples $\mathbf{x}$ in S such that $D(\mathbf{x}, t) > r/8$, then

$$\frac{r}{4}|S| < \sum_{\mathbf{x} \in S} D(\mathbf{x}, t) \leq Nr + (|S| - N)\,\frac{r}{8},$$

yielding $N \geq |S|/7$. Now, if $D(\mathbf{x}, t) \geq r/8$ then $L(\mathbf{x}, t)$ has error at least $(8\epsilon/r)(r/8) = \epsilon$. Hence (observing that each element of S has equal probability according to μ^m),

$$\mu^m \{\mathbf{s} \in S(m,t) : \mathrm{er}_\mu(L(\mathbf{s})) \geq \epsilon\} \geq \frac{N}{|S|}\mu^m(S) \geq \frac{1}{7}\mu^m(S).$$

The probability that a point chosen according to the distribution μ lies in $F = \{z_1, z_2, \ldots, z_r\}$ is 8ϵ. If $m \leq r/32\epsilon$, the probability that a sample of length m has at least $d/2$ entries from F is therefore, by a standard Chernoff bound (see [85], for example), at most 93/100. Therefore if $m \leq r/(32\epsilon) = (d-1)/(32\epsilon)$, we have

$$\mu^m \{\mathbf{s} \in S(m,t) : \mathrm{er}_\mu(L(\mathbf{s})) \geq \epsilon\} \geq \frac{1}{7}\frac{7}{100} = \frac{1}{100},$$

as required.

Now we prove the second bound. Since C has at least three distinct functions, it contains c_1 and c_2 such that for some $a, b \in X$, $c_1(a) = c_2(a)$ and $c_1(b) = 1, c_2(b) = 0$. Without loss of generality, we assume $c_1(a) = c_2(a) = 1$. Let $0 < \delta, \epsilon < 1$ and let μ be the probability distribution for which $\mu(a) = 1 - \epsilon$ and $\mu(b) = \epsilon$ (and μ is zero elsewhere on X). The probability that a sample of length m has all its entries equal to a is $(1-\epsilon)^m$. Thus, if

$$m \leq \frac{(1-\epsilon)}{\epsilon} \ln\left(\frac{1}{\delta}\right),$$

then, with probability greater than δ, a sample $\mathbf{x}$ of length m has all its entries equal to a. Let $\mathbf{a}^1$ denote the training sample $\mathbf{a}^1 = ((a,1), \ldots, (a,1))$ of length m. Then $\mathbf{a}^1$ is a training sample for both c_1 and c_2. Suppose that L is a PAC (C,H)-learning algorithm. If b is a positive example of $L(\mathbf{a}^1)$ then $L(\mathbf{a}^1)$ has

error at least ϵ (the probability of b) with respect to c_1, while if b is a negative example of $L(\mathbf{a}^1)$ then this hypothesis has error at least ϵ with respect to c_2. It follows that there is $t \in C$, either c_1 or c_2 as above, such that

$$m \leq \frac{(1-\epsilon)}{\epsilon} \ln\left(\frac{1}{\delta}\right) \Longrightarrow \mu^m \left\{\mathbf{s} \in S(m,t) : \mathrm{er}_\mu\left(L(\mathbf{s})\right) > \epsilon\right\} > \delta.$$

The result follows. □

An immediate corollary of this result is that if a concept space C has infinite VC-dimension then there is no PAC learning algorithm for (C, H) for *any* hypothesis space H.

7 VC-Dimension of Binary-Output Networks

7.1 Introduction

Recall that the VC-dimension, $\mathrm{VCdim}(\mathcal{N}, X)$ of a neural network $\mathcal{N}$ with a single binary output is the VC-dimension of the set of functions it can compute (over the set X of examples). The following two results follow directly from the theory presented earlier.

Theorem 7.1 *Suppose that the binary-output neural network $\mathcal{N}$ can take on any state in a set Ω of possible states. Suppose that the network receives inputs from an example set X and that $\mathrm{VCdim}(\mathcal{N}, X)$ is finite. Let $t : X \to \{0,1\}$ and suppose that μ is any probability distribution on X. For $0 < \delta, \epsilon < 1$, let*

$$m_*(\delta, \epsilon) = \frac{8}{\epsilon}\left(\ln\left(\frac{4}{\delta}\right) + \mathrm{VCdim}(\mathcal{N}, X) \ln\left(\frac{192}{\epsilon}\right)\right).$$

Suppose a training sample for t of length $m \geq m_*(\delta, \epsilon)$ is drawn randomly according to μ^m (in other words, each example in the training sample is randomly drawn according to μ and is then presented, together with its correct classification, to the network). Suppose that the state of the network is adjusted on receiving the training sample. Then the following holds with probability [9] at least $1 - \delta$: if the state $\omega \in \Omega$ of the network after training on the sample is such that the function $\mathcal{N}_\omega$ then computed has observed error at most $\epsilon/2$ on the training sample, then the network will classify a further randomly chosen example correctly with probability at least $1 - \epsilon$.

In the statement of the following theorem, a learning algorithm for $\mathcal{N}$ simply means an $(H_\mathcal{N}, H_\mathcal{N})$-learning algorithm, where $H_\mathcal{N}$ is the set of functions computable by $\mathcal{N}$ on X.

Theorem 7.2 *Suppose that $\mathcal{N}$ is a binary-output neural network. Suppose that the network receives inputs from an example set X and that $\mathrm{VCdim}(\mathcal{N}, X)$ is finite. Suppose that L is a consistent learning algorithm for $\mathcal{N}$. Then L is PAC and its sample complexity satisfies*

$$m_L(\delta, \epsilon) \leq \frac{8}{\epsilon}\left(\ln\left(\frac{4}{\delta}\right) + \mathrm{VCdim}(\mathcal{N}, X) \ln\left(\frac{48}{\epsilon}\right)\right).$$

Furthermore, the lower bound result, Theorem 6.2, shows that (for fixed δ and ϵ), one cannot guarantee the conclusions of either of the above two theorems unless a sample of length at least proportional to the VC-dimension of the network is used. In this sense, the VC-dimension fairly precisely quantifies the sample complexity of PAC learning. It is worth remarking that if an explicit bound is known on the growth function, then significantly better sample complexity upper bounds than those presented in the above theorem can often be derived. However, often one only has some bound on the VC-dimension. (Of course, using Sauer's Lemma, this gives a bound on the growth function; but it is precisely this that is used in deriving the above bound.)

[9] The probability referred to here is the μ^m probability that a training sample of length m has the property stated.

The *size* of a neural network of a particular type may be defined to be the number W of variable parameters (the number of weights and thresholds). Suppose we want to design an *efficient* PAC learning algorithm for a class of neural networks. We shall discuss computational complexity later, but for the moment we simply observe that if the PAC algorithm is to run in time polynomial in the size of the network, then, certainly, it must have polynomial sample complexity. The theory already presented shows that this is true if and only if the VC-dimension of a network in the class is polynomial in the number of weights. We now present some results on the VC-dimensions of particular types of neural network.

7.2 Linearly weighted neural networks

We start with a fairly simple, but quite general, class of neural networks. These are the *linearly weighted neural networks*. This type of network was first introduced in the 1960s and it includes simple 'radial basis function networks' and 'polynomial discriminators'. A *linearly-weighted* neural network, with n real inputs and a single binary output, is defined by a fixed set $\phi_1, \phi_2, \ldots, \phi_p$ of *basis* functions, each of which maps $\mathbf{R}^n$ to $\mathbf{R}$. The state of the network is determined by a variable weight vector $w = (w_1, w_2, \ldots, w_p)$. The output corresponding to example $x \in \mathbf{R}^n$ and state w is 1 or 0 according as the weighted sum $\sum_{i=1}^{p} w_i \phi_i(x)$ is positive or not.

An example of a linearly weighted neural network is the *polynomial discriminator*, obtained when all the functions ϕ_i are monomials, by which is meant products of the components of x, such as $x_1^2 x_3 x_n^3$. The *degree* of a polynomial discriminator is the maximum total degree of any ϕ_i, in the usual sense. We have the following result [6, 16, 15], which is proved using Dudley's result, Theorem 4.2.

Theorem 7.3 *Suppose that $\mathcal{N}$ is a polynomial discriminator of degree at most k, on n inputs. Then*

$$\mathrm{VCdim}(\mathcal{N}, \mathbf{R}^n) \leq \binom{n+k}{k}, \quad \mathrm{VCdim}(\mathcal{N}, \{0,1\}^n) \leq \sum_{i=0}^{k} \binom{n}{i},$$

and these bounds are tight in the sense that there are such $\mathcal{N}$ with VC-dimensions meeting these bounds.

Proof For all n and k, let $B(n,k)$ be the set of all functions $f : \mathbf{R}^n \to \mathbf{R}$ of the form $f(x) = x_1^{r_1} x_2^{r_2} \ldots x_n^{r_n}$, where $r_1 + r_2 + \ldots + r_n \leq k$. It is fairly easy to show that $B(n,k)$ is a linearly independent set of functions on $\mathbf{R}^n$. For $k \leq n$, let $C(n,k)$ be the set of all functions $g : \{0,1\}^n \to \mathbf{R}$ of the form $g(x) = x_1^{b_1} x_2^{b_2} \ldots x_n^{b_n}$, where $b_i = 0$ or 1 and at most k of the b_i are 1. Then it can be shown that $C(n,k)$ is a linearly independent set of functions on $\{0,1\}^n$. Let $\langle B(n,k) \rangle$ be the real vector space of functions spanned by the set $B(n,k)$. This has linear dimension $|B(n,k)|$, which is easily seen to be $\binom{n+k}{k}$. Clearly, if $\mathcal{N}$ is a polynomial discriminator of degree at most k on n inputs then $H_{\mathcal{N}}$, the set of functions computed by $\mathcal{N}$, is contained in $\mathrm{pos}\langle B(n,k) \rangle$. It follows that

$$\mathrm{VCdim}(\mathcal{N}, \mathbf{R}^n) \leq \mathrm{VCdim}\left((\mathrm{pos}\langle B(n,k) \rangle\right) = \binom{n+k}{k},$$

where the equality follows from Theorem 4.2. If we consider only binary inputs, it is clear that no basis function which contains powers of any x_i greater than 1 is necessary; this is simply because, if $x = 0$ or 1 and r is any positive integer, then $x^r = x$. It follows that the space $(H_{\mathcal{N}})|_{\{0,1\}^n}$ of functions computable by $\mathcal{N}$ on $\{0,1\}^n$ is contained in $\mathrm{pos}\langle C(n,k) \rangle$, where $\langle C(n,k) \rangle$ is the real vector space generated by $C(n,k)$. Using Dudley's result, we obtain

$$\mathrm{VCdim}(\mathcal{N}, \{0,1\}^n) \leq \mathrm{VCdim}\left((\mathrm{pos}\langle C(n,k) \rangle\right) = |C(n,k)| = \sum_{i=0}^{k} \binom{n}{i}.$$

The bounds are met when $\mathcal{N}$ uses as basis functions all functions in $B(n,k)$ (respectively, $C(n,k)$). □

The *radial basis function networks* (with fixed centers) are another important class of linearly-weighted networks. Here each ϕ_i is defined in terms of the distance of the example from a fixed *center* y^i: $\phi_i(x) =$

$\phi(\|x - y^i\|)$, where ϕ is a fixed function and $\|z\|$ is the usual Euclidean norm of z. We have the following result, which is obtained using interpolation results of Micchelli [87]. Details of the proof may be found in [16, 15, 64], where further results along the same lines are presented.

Theorem 7.4 *Let $\mathcal{N}$ be a linearly weighted neural network based on p basis functions $\phi_1, \phi_2, \ldots, \phi_p$ where $\phi_i(x) = \phi(\|x - y^i\|)$ and $\phi(r)$ takes any one of the forms r, $\exp(-cr^2)$, $(r^2 + c^2)^{\alpha}$, $(r^2 + c^2)^{-\beta}$, (with α and β positive constants, with $\beta < 1$). Then the VC-dimension of this fixed-center radial basis function network on example set $\mathbf{R}^n$ is exactly p.*

7.3 Linear threshold networks

We now present upper bounds on the VC-dimension of linear threshold neural networks. In these networks, each neuron—or computation unit—is a linear threshold neuron. It forms the weighted sum of its inputs and it outputs 1 if this sum exceeds its threshold and 0 otherwise.

The following result, a bound on the VC-dimension of such networks on binary inputs, is due to Natarajan [89].

Theorem 7.5 *Let $\mathcal{N}$ be a linear threshold network with N neurons (including the input nodes), n input nodes and a total of W variable weights and thresholds. Then*

$$\mathrm{VCdim}(\mathcal{N}, \{0,1\}^n) \leq WN \log N.$$

Proof The proof uses a result due to Hong (see [90] for details), which states that a linear threshold network with N neurons and $\{0,1\}$-inputs need only use integer weights which can be encoded using $N \log N$ binary bits. It follows that the number of distinct functions computable by the network is at most $\left(2^{N \log N}\right)^W$. Since, for any finite space H, $\mathrm{VCdim}(H) \leq \log |H|$, we have $\mathrm{VCdim}(\mathcal{N}, \{0,1\}^n) \leq WN \log N$. □

This result is limited in that it concerns only networks with binary-valued inputs. The following result of Baum and Haussler [27] applies to *feedforward* linear threshold networks. These are linear threshold networks with no feedback; in other words, networks in which the underlying directed graph is acyclic. This result applies for real inputs, as well as for binary inputs. Moreover, in the case of feedforward networks with binary inputs, it is an improvement of Natarajan's result. (However, it should be noted that Natarajan's result applies to certain types of threshold network which are not feedforward networks, namely those which compute a function of their inputs in a finite time, with no infinite 'cycling'.) The result presented here is a slightly weaker version of Baum and Haussler's original result, with a simpler proof that makes use, as in [82], of a form of Sauer's inequality.

Theorem 7.6 *Suppose that $\mathcal{N}$ is a feedforward linear threshold network having a total of W variable weights and thresholds, and n inputs. Then*

$$\mathrm{VCdim}\left(\mathcal{N}, \{0,1\}^n\right) \leq \mathrm{VCdim}\left(\mathcal{N}, \mathbf{R}^n\right) < 6W \log W.$$

Proof Let $X = \mathbf{R}^n$ and suppose that $\mathbf{x} \in X^m$ is a sample of m points from X. We bound the growth function of $H = H_{\mathcal{N}}$ by bounding $\Pi_H(\mathbf{x})$ independently of $\mathbf{x}$. Denote by N the number of computation units (that is, the number of linear threshold neurons) in the network. Since the network is feedforward, the computation units may be labeled with the integers $1, 2, \ldots, N$ so that if the output of computation unit i is fed into unit j then $i < j$. Consider any particular computation unit, i. Denote the in-degree of i by d_i. (This is the number of units, including the input units, whose output is fed into unit i.) Since the computation unit is a linear threshold unit, the set of functions computable by that unit (in isolation) has VC-dimension $d_i + 1$, by Theorem 4.3. It follows, by a simple consequence of Sauer's Lemma, that if $\mathbf{y}$ is any sample of length m of points in $\{0,1\}^{d_i}$, then the number of ways in which unit i can classify $\mathbf{y}$ is at most

m^{d_i+2} for $m \geq d_i + 1$. It follows that, (if $m > \max_i d_i + 1$) the number of classifications $\Pi_H(\mathbf{x})$ of $\mathbf{x} \in X^m$ by the network is bounded by the quantity

$$m^{d_1+2} m^{d_2+2} \ldots m^{d_N+2},$$

which, since $W = d_1 + d_2 + \ldots + d_N + N$, the total number of weights and thresholds, is at most m^{W+N}. Since $W \geq N$, this is at most m^{2W}. Now, $m^{2W} < 2^m$ if $m = 6W \log W$, from which it follows that the VC-dimension of the network is less than $6W \log W$. □

This bound on the VC-dimension immediately yields an upper bound on the sample complexity of a consistent learning algorithm in PAC learning with accuracy parameter ϵ and confidence parameter δ. This bound is of order $\epsilon^{-1}(W \log W \log(1/\epsilon) + \log(1/\delta))$. (See Theorem 6.1.) However, as we noted earlier, it may be possible to obtain a better bound using the explicit estimate we obtained for the growth function, namely $\Pi_H(m) < m^{2W}$ for $m > W$. If we do this, we obtain a sample complexity bound of order $\epsilon^{-1}W(\log(W/\epsilon) + \log(1/\delta))$, as in [27].

The VC-dimension bound presented by Baum and Haussler, which is obtained in the same way, but by using Sauer's Lemma a little more carefully, is $2W \log(eN)$, where N is the number of computation units and e is the base of the natural logarithm. This result has subsequently been improved by Sakurai who, in [98], announces an upper bound of $W(\log(N-1) + o(\log N))$, as $N \to \infty$. The constants in the bound are not of any great theoretical significance. The main thing to notice is that the upper bounds on the VC-dimension of feedforward linear threshold networks are of order $W \log W$ where W is the total number of weights and thresholds; that is, the total number of variable parameters determining the state of the network. We have already seen that the simple perceptron on n inputs has VC-dimension $n + 1$, which is exactly the number of variable parameters in this network. Furthermore, for the more general case of linearly weighted neural networks, the theory presented above, together with Dudley's result, shows that the VC-dimension is at most W, no matter what the basis functions are (and that it is exactly W if the basis functions are linearly independent). It is natural to ask whether the bound of Theorem 7.6 is of the best possible order or whether one should be able to prove that in this case the VC-dimension is of order W. This leads us to a discussion of results providing lower bounds on the VC-dimension of linear threshold networks. In their paper, Baum and Haussler [27] proved that there are threshold networks with one hidden layer (that is, of depth two) and VC-dimension $\Omega(W)$, where W is the number of variable weights. Bartlett [22] showed that this holds for all such networks and he showed also, by means of results such as the following, that, for large classes of depth-three networks, the VC-dimension is $\Omega(W)$.

Theorem 7.7 *Suppose that $\mathcal{N}$ is a depth-three feedforward linear threshold network having one output unit, n inputs, k_1 units in the first hidden layer and k_2 in the second. Suppose also that $\mathcal{N}$ is fully connected between layers. If $k_1 > n > 1$ and $k_2 \leq k_1$, then*

$$\mathrm{VCdim}(\mathcal{N}, \mathbf{R}^n) \geq nk_1 + \frac{k_1(k_2 - 1)}{2} + 1.$$

The lower bounds discussed so far are linear in the number of weights. We still, therefore, have the following question: can it be true that some feedforward linear threshold networks have VC-dimension *significantly* larger than W, the number of variable parameters? The answer is 'yes', and this was first shown by Maass [80]. The following result may be found in [81].

Theorem 7.8 *Assume that $\{\mathcal{N}_n\}$ is any sequence of feedforward linear threshold networks of depth at least three (that is, having at least two hidden layers), fully connected between successive layers, and such that: $\mathcal{N}_n$ has n inputs; $\mathcal{N}_n$ has $\Omega(n)$ threshold units in the first hidden layer; $\mathcal{N}_n$ has at least $4 \log n$ units in the second hidden layer. Then*

$$\mathrm{VCdim}(\mathcal{N}_n, \{0,1\}^n) = \Theta(n^2 \log n).$$

Note that, with $\mathcal{N}_n$ as in this theorem, $n^2 \log n = \Theta(W \log W)$ since the network is fully connected. Theorem 7.8 refers to networks taking binary inputs. It is perhaps surprising that, even with this restriction, a network may have a 'superlinear' VC-dimension. The result shows that no upper bound better than order $W \log W$ can be given: to within a constant, the bound of Theorem 7.6 is tight.

The networks of Theorem 7.8 have depth at least three. The following result of Sakurai [98] shows that there are feedforward linear threshold networks of depth two (with just one hidden layer) having superlinear VC-dimension on *real* inputs. These networks are smaller than those of Theorem 7.8, but the result concerns real inputs, not binary inputs and hence is not immediately comparable with Theorem 7.8. We also state a fairly tight upper bound Sakurai obtained on the VC-dimension.

Theorem 7.9 *Let $\mathcal{N}$ be a fully-connected depth-two feedforward linear threshold network having n inputs and h units in the hidden layer. Then*

$$\mathrm{VCdim}(\mathcal{N}, \mathbf{R}^n) \geq \frac{1}{2} nh \left(\log h + o(\log h) + O\left(\frac{(\log h)^2}{n} \right) \right)$$

and

$$\mathrm{VCdim}(\mathcal{N}, \mathbf{R}^n) \leq nh \left(\log h + o(\log h) \right),$$

as $h, n \to \infty$.

7.4 Other activation functions

We now move on to discuss more complex types of neural networks, in which the computation units can perform operations more sophisticated than simple linear thresholding of the weighted sum of their inputs. The bounds obtained so far on the VC-dimensions of particular neural network architectures have been fairly precise. In this section, we shall be content with rather looser upper bounds on the VC-dimension. In particular, we should like to obtain upper bounds polynomial in the number of weights.

We first need some notation. Let us take $i \to j$ to mean that the output of unit i feeds into unit j. The linear threshold networks we have discussed compute the output o_j of unit j as follows:

$$o_j = \mathcal{H}\left(\sum_{i \to j} w_{ij} o_i - \theta_j \right),$$

where w_{ij} is the weight on the connection from i to j, θ_j is the threshold on unit j, and $\mathcal{H}$ is the *Heaviside step function*: $\mathcal{H}(x) = 1$ if $x > 0$ and $\mathcal{H}(x) = 0$ if $x \leq 0$.

The Heaviside function $\mathcal{H}$ is just one example of an activation function and in practice it is often replaced by a sigmoid function, f. This is some 'smooth' function from $\mathbf{R}$ to $[0, 1]$, with $f(x) \to 0$ as $x \to -\infty$ and $f(x) \to 1$ as $x \to \infty$. The best-known example is the *standard sigmoid* function, given by $\sigma(x) = 1/(1+e^{-x})$. We now consider networks with one linear threshold output unit and sigmoid activation functions at all hidden units. (The linear threshold output unit ensures that the network computes $\{0, 1\}$-valued functions, so that it is legitimate to discuss the VC dimension.)

Sontag has shown (see [111, 83]) that there is a neural network $\mathcal{N}$ of *infinite* VC-dimension, having two real inputs, two hidden units with sigmoid activation function

$$f(y) = \frac{1}{\pi} \tan^{-1} y + \cos y / (7 + 7y^2) + 1/2,$$

and a linear threshold output unit. In view of this, it is not possible to obtain a general VC-dimension bound for sigmoid neural networks and one must consider particular sigmoids or particular types of sigmoid separately.

First, we present the following, quite general, result of Goldberg and Jerrum [48].

Theorem 7.10 *Suppose that $\{\mathcal{N}_{W,n} : W, n \in \mathbf{N}\}$ is a family of binary-output neural networks, where $\mathcal{N}_{W,n}$ has n inputs and W variable real parameters (weights and thresholds). Suppose there is an algorithm $\mathcal{A}_{W,n}$, for each W and n, which takes as input $w \in \mathbf{R}^W$ and an example $x \in \mathbf{R}^n$ and decides whether $\mathcal{N}_{W,n}$ outputs 1 on input x when in state w. Suppose that this algorithm uses only conditional jumps (conditional on the equality and inequality of real numbers) and the standard arithmetic operations—addition, subtraction, multiplication and division—on real numbers. Assume that each such operation is performed in constant time. If the running time of $\mathcal{A}_{W,n}$ is bounded by $t = t(W, n)$, then* $\mathrm{VCdim}(\mathcal{N}_{W,n}, \mathbf{R}^n) = O(Wt)$.

This result is useful for a number of different types of network. It can be applied by identifying the computation of the algorithm $\mathcal{A}_{W,n}$ on input (w, x) with the action of the network $\mathcal{N}_{W,n}$ on input x when it is in state w. The 'conditional jumps' of the algorithm could correspond to defining the activation function in a piecewise manner, and to thresholding. The arithmetic operations of the algorithm correspond to calculating the input to a computation unit and the value of the activation function on this input. In order for the theorem to apply usefully in this way, the activation functions must be based on thresholding and on the simple arithmetical operations of addition, multiplication, subtraction and division. If this is so, then the theorem implies that the VC-dimension of the network $\mathcal{N}_{W,n}$ is of order Wt where t is a bound on the time taken by the network to calculate its output. As a very simple example from [48], suppose that each $\mathcal{N}_{W,n}$ is a feedforward linear threshold network. The number of operations required for this network to produce an output, given an input in $\mathbf{R}^n$, is $O(W)$ and hence the theorem yields an upper bound of $O(W^2)$ for the VC-dimension of $\mathcal{N}_{W,n}$. Of course, we know that there is a better upper bound of $O(W \log W)$, but the $O(W^2)$ bound is very easily obtained from Theorem 7.10. The theorem is very useful in obtaining polynomial upper bounds on the VC-dimension of neural networks. However, it does *not* apply to the standard sigmoid function or any other activation function involving exponentiation: it only applies if the computations in the network use conditioning and *standard* arithmetic operations only. Nevertheless, there are sigmoid functions which do not involve exponentiation. For example, Goldberg and Jerrum applied their result to networks in which each hidden unit has activation function

$$f(x) = \begin{cases} 1 - 1/(2x+2) & \text{if } x > 0; \\ 1/(2-2x) & \text{if } x \leq 0, \end{cases}$$

and in which the output unit is a threshold unit. Theorem 7.10 implies that the VC-dimension of such a network is polynomial in the number of weights.

Another application of Theorem 7.10 is to networks in which each activation function f_j is a piecewise polynomial function. Maass [80] proved that the VC-dimension of such networks can be polynomially bounded provided the networks have constant depth. Applying Theorem 7.10 gives the following result [48, 83], in which no bound on the depth is necessary.

Theorem 7.11 *Let $\{\mathcal{N}_W\}$ be a family of feedforward neural networks with binary outputs, having piecewise polynomial functions of bounded degree and bounded number of pieces as activation functions on the hidden units. Suppose the network $\mathcal{N}_W$ has W variable parameters. Then* $\mathrm{VCdim}(\mathcal{N}_W, \mathbf{R}^n) = O(W^2)$.

Elaborations on this result have been obtained by Maass [79]. Whether this bound can be improved to $O(W \log W)$ is an open problem.

Returning to sigmoid functions, the following result was obtained by Macintyre and Sontag [84], using deep results from logic.

Theorem 7.12 *Let $\mathcal{N}$ be a feedforward network with binary output and the standard sigmoid function as activation function on the hidden units. Then $\mathcal{N}$ has finite VC-dimension.*

In view of Sontag's example of a sigmoid network with infinite VC-dimension, this is an important result. Macintyre and Sontag did not give an explicit upper bound on the VC-dimension of such networks, but the bound one obtains by close examination of their proof is doubly-exponential in W. (This observation was communicated to me by Wolfgang Maass.)

In practice, computers work to finite accuracy and therefore in any computer simulation of a neural network, the inputs (and weights) are discrete and the following theorem of Bartlett and Williamson [24] is applicable.

Theorem 7.13 *Let D be a positive integer and $X_D = \{-D, -D+1, \ldots, D-1, D\}^n$. Suppose $\mathcal{N}$ is a depth-two binary-output neural network having standard sigmoid activation functions on the hidden units. Then* $\mathrm{VCdim}(\mathcal{N}, X_D) \leq 8W \log(11WD)$, *where W is the number of weights and thresholds.*

Karpinski and Macintyre [67], using some fairly sophisticated mathematical machinery, obtained a polynomial upper bound on the VC-dimension of standard sigmoid networks. Specifically, they proved the following.

Theorem 7.14 *Let $\mathcal{N}$ be a feedforward network with binary output and the standard sigmoid function as activation function on the hidden units. Suppose that the total number of adjustable weights and thresholds is W and that there are k computational units. Then $\mathcal{N}$ has finite VC-dimension at most*

$$Wk(Wk-1) + \text{lower order terms},$$

which is $O(W^4)$.

Koiran and Sontag [75] showed that the VC-dimension of (unbounded depth) standard sigmoid nets is $\Omega(W^2)$, so there is a strict separation between the VC-dimension of threshold nets and sigmoid nets.

We have presented a number of results on the VC-dimensions of neural networks. Similar analyses [80, 67] can be carried out if the activation functions do not take as input the *linear* weighted sum of the outputs, o_i, feeding in, but rather take as their arguments some *polynomial* function of these. Such units are often known as *sigma-pi* units. The polynomial discriminators discussed earlier may be thought of as single units of this type.

7.5 The effect of weight restrictions

In this subsection we briefly discuss how restriction of the weights in some way can change the VC-dimension of a network. First, we have a result like one due to Baum and Haussler [27], which concerns the VC-dimension of a feedforward linear threshold network in which only a certain number of weights can be non-zero. These relevant connections are not specified in advance of training. All that is asserted is that, of a total of W weights and thresholds, at most W' can be non-zero after network training. The hypothesis space in this case is the set of all functions computable by the network in a state with at most W' non-zero weights and thresholds.

Theorem 7.15 *Suppose that $\mathcal{N}$ is a feedforward linear threshold network having a total of W variable weights and thresholds and n inputs. Let H' be the set of functions computable by $\mathcal{N}$ when at most W' of the weights and thresholds are non-zero. Then*

$$\mathrm{VCdim}\,(H') < 6W' \log W.$$

Proof We proceed in a manner similar to the proof of Theorem 7.6. First, note that there are $\binom{W}{W'}$ possible selections for the non-zero weights and thresholds. Given a particular such choice, the network computes a set of functions having growth function bounded by $m^{2W'}$. This is proved as in Theorem 7.6. It follows that

$$\Pi_{H'}(m) \leq \binom{W}{W'} m^{2W'} < W^{W'} m^{2W'}.$$

This quantity is less than 2^m (for W greater than some fixed constant) if $m \geq 6W' \log W$ and this is, therefore, a bound on the VC-dimension. □

The theorem demonstrates that the power of a neural network decreases significantly when it is forced to set some number of weights to zero. Using the bound on the growth function obtained in the proof, as in [27], one obtains a bound on the sample complexity of any consistent learning algorithm for $\mathcal{N}$ which produces states with at most W' non-zero weights and thresholds. (Additionally, one obtains a sample length bound for the property described in Theorem 7.1.) As earlier, the bounds obtained using the explicit bound on the growth function are better than those obtained simply by using the VC-dimension bound.

Suppose that there are known symmetries inherent in the class of concepts being learned. These symmetries may be reflected in learning algorithms for the neural network, by constraining groups of weights to have the same value. Such techniques have been discussed in [102], for example. Shawe-Taylor [103] has bounded the sample complexity of such learning algorithms by bounding the growth function in a manner similar to that in [27]. His results, following [19, 104], also apply more generally to networks having more than one output node, but we shall not discuss this aspect here. Let $\mathcal{N}$ be a feedforward linear threshold network. Suppose that there are W^* *classes* of weights and thresholds, so that all weights (and thresholds) within one such equivalence class are forced to have the same value. Shawe-Taylor shows that the VC-dimension of the set of functions computable by $\mathcal{N}$, subject to the weight restrictions, is of order $W^* \log N$ where N is the number of computation units. Recalling that the upper bound of Baum and Haussler on the VC-dimension of the unconstrained network is of order $W \log N$, we see that W has been replaced by the effective number of weights and thresholds, W^*.

In both the above cases, restricting the weights reduces the VC dimension. A common technique in neural network learning is to keep the magnitude of the weights as small as possible; see [61], for example. It is natural to ask whether such a restriction significantly decreases the VC-dimension. Lee, Bartlett and Williamson [76] have investigated this problem for depth two feedforward networks with certain activation functions on the hidden units and a linear threshold unit as the output unit. In particular, they have the following result.

Theorem 7.16 *Let $\mathcal{N}$ be a feedforward network with n inputs, one linear threshold output unit, and one hidden layer of k units having activation function* $\tanh x$. *Let $W = k(n+2)+1$ be the number of weights and thresholds. Let U be* any *open subset of $\mathbf{R}^W$ containing the origin and, for $M > 0$, let $H_{M,U}$ be the set of functions on $(-M, M)^n$ computable by $\mathcal{N}$ using states in U. Then, for any $M > 0$,*

$$\mathrm{VCdim}(H_{M,U}) \geq (k-1)(n+1)+1.$$

This shows that if we place a bound on the absolute values of the weights, and if we similarly restrict the examples, then the VC-dimension is still at least $\Omega(W)$.

8 Computational Complexity of Learning

Thus far, a learning algorithm has been defined as a function mapping training samples into hypotheses. We have noted that if learning is to take place in time polynomial in the size of the network, then a polynomial bound on the VC-dimension is necessary. However, this condition is not sufficient. There is sometimes an inherent computational intractability in producing a probably approximately correct hypothesis, as we see in this section. The results of the previous section show that polynomial VC-dimension bounds hold for many types of network. For these networks, then, the sample complexity for PAC learning is polynomial and hence if there is not an efficient PAC learning algorithm, it is for reasons of *computational complexity*.

In this section, we shall be more specific about the *algorithmics*. We shall concentrate on the case $C = H$, where the hypothesis space and the concept space coincide. In the context of neural networks, this means that all target functions are computable by the network we are training. If PAC learning by a learning algorithm is to be of practical value, it must, first, be possible to implement the learning algorithm on a computer; that is, it must be *computable* and therefore, in a real sense, an *algorithm*, not just a function. Further, it should be possible to implement the algorithm 'quickly'.

We shall consider neural networks on binary inputs. If $\mathcal{N}$ is a network with n inputs, then the example space is $X = \{0,1\}^n$. To keep the argument general at this stage, we shall phrase our discussion in terms of

general hypothesis spaces defined on $\{0,1\}^n$ for some n, rather than deal specifically with neural networks. It is convenient to make the following definitions. We say that a union of hypothesis spaces $H = \bigcup H_n$ is *graded* by example size n, when H_n denotes the space of hypotheses defined on examples of size n. For example, H_n may be the space P_n of functions computable by the perceptron, defined on real vectors of length n. By a *learning algorithm for* $H = \bigcup H_n$ we mean a function L from the set of training samples for hypotheses in H to the space H, such that when $\mathbf{s}$ is a training sample for $h \in H_n$ it follows that $L(\mathbf{s}) \in H_n$.

Consider a learning algorithm L for a hypothesis space $H = \bigcup H_n$, graded by example size. An input to L is a training sample, which consists of m examples of size n together with the m single-bit labels. The total size of the input is therefore $m(n+1)$, and it would be possible to use this single number as the measure of input size. However, there is some advantage in keeping track of m and n separately, and so we shall use the notation $R_L(m,n)$ to denote the worst-case running time of L on a training sample of m examples of size n.

A learning algorithm L for $\bigcup H_n$ is said to be a PAC learning algorithm if L acts as a PAC learning algorithm for each H_n. The sample complexity provides the link between the running time $R_L(m,n)$ of a learning algorithm (that is, the number of operations required to produce its output on a sample of length m when the examples have size n) and its running time as a PAC learning algorithm (that is, the number of operations required to produce an output which is probably approximately correct with given parameters). For a general hypothesis space G and for δ, ϵ between 0 and 1, let

$$m_0(G,\delta,\epsilon) = \frac{8}{\epsilon}\left(\ln\left(\frac{4}{\delta}\right) + \mathrm{VCdim}(G)\ln\left(\frac{48}{\epsilon}\right)\right).$$

This quantity appeared in Theorem 6.1 as an upper bound on the sample complexity of a consistent (G,G)-learning algorithm. Since a sample of length $m_0(H_n,\delta,\epsilon)$ is sufficient for PAC learning, the number of operations required of L is at most $R_L(m_0(H_n,\delta,\epsilon),n)$.

Until now, we have regarded the accuracy parameter ϵ as fixed but arbitrary. It is clear that decreasing this parameter makes the learning task more difficult, and therefore the running time of an efficient PAC learning algorithm should be constrained in some appropriate way as ϵ^{-1} increases. We say that a learning algorithm L for $H = \bigcup H_n$ is *efficient with respect to accuracy and example size* if its running time is polynomial in m and n and the sample complexity $m_L(H_n,\delta,\epsilon)$ depends polynomially on n and ϵ^{-1}.

We are now ready to consider the implications for learning of the theory of NP-hard problems. Let $H = \bigcup H_n$ be a hypothesis space of functions, graded by the example size n. The *consistency problem* for H may be stated as follows.

$H-CONSISTENCY$
Instance A training sample $\mathbf{s}$ of labeled examples of size n.
Question Is there a hypothesis in H_n consistent with $\mathbf{s}$?

In practice, we wish to produce a consistent hypothesis, rather than simply know whether or not one exists. In other words, we have to solve a search problem, rather than a decision problem. But these problems are directly related: if the search problem can be solved in polynomial time, so too can the decision problem.

If there is a consistent learning algorithm L for a graded hypothesis space $H = \bigcup H_n$ such that $\mathrm{VCdim}(H_n)$ is polynomial in n and the algorithm runs in time polynomial in the sample length m, and in n, then the results presented earlier show that L PAC learns H_n with running time polynomial in n and ϵ^{-1}, and so is efficient with respect to accuracy and example size. Roughly speaking we may say that an efficient 'consistent-hypothesis-finder' is an efficient 'PAC learner'. It is natural to ask to what extent the converse is true. It turns out that efficient PAC learning does imply efficient consistent-hypothesis-finding, provided we are prepared to accept a *randomised algorithm*. For our purposes, a randomised algorithm A has access to a random number generator and is allowed to use these random numbers as part of its input. (See [39].) The computation carried out by the algorithm is determined by its input, so that it depends on the particular sequence produced by the random number generator. It follows that we can speak of the probability that A has a given outcome.

We say that a randomised algorithm A 'solves' a search problem Π if it behaves in the following way. The algorithm always halts and produces an output. If A has failed to find a solution to Π then the output is simply *no*. If there is no solution to the search problem Π, the algorithm always outputs *no*, whereas if there *is* a solution then, with probability at least $1/2$, A outputs a solution. The probability that the algorithm fails in k attempts is at most $(1/2)^k$, which approaches zero very rapidly with increasing k.

The following result is from [93]. (See also [89, 56].)

Theorem 8.1 *Let $H = \bigcup H_n$ be a hypothesis space and suppose that there is a PAC learning algorithm for H which is efficient with respect to accuracy and example size. Then there is a randomised algorithm which solves the problem of finding a hypothesis in H_n consistent with a given training sample of a hypothesis in H_n, and which has running time polynomial in n and m (the length of the training sample).*

Proof Suppose that $\mathbf{s}^*$ is a training sample for a target hypothesis $t \in H_n$, and that $\mathbf{s}^*$ contains m^* distinct labeled examples. We shall show that it is possible to find a hypothesis consistent with $\mathbf{s}^*$ by running the given PAC learning algorithm L on a related training sample. Define a probability distribution μ on the example space X by $\mu(x) = 1/m^*$ if x occurs in $\mathbf{s}^*$ and $\mu(x) = 0$ otherwise. We can use a random number generator with output values i in the range 1 to m^* to select an example from X according to this distribution. Thus the selection of a training sample of length m for t, according to the probability distribution μ, can be simulated by generating a sequence of m random numbers in the required range.

Let L be a PAC learning algorithm as postulated in the statement of the Theorem. Then, when δ, ϵ, are given, we can find an integer $m_L(n, \delta, \epsilon)$ for which the probability (with respect to training samples $\mathbf{s} \in S(m_L, t)$) that the error of $L(\mathbf{s})$ is less than ϵ is greater than $1 - \delta$. Suppose we specify the confidence and accuracy parameters to be $\delta = 1/2$ and $\epsilon = 1/m^*$. Then if we run the given algorithm L on a training sample $\mathbf{s}$ of length $m_L(n, 1/2, 1/m^*)$, drawn randomly according to the distribution μ, the PAC property of L ensures that the probability that the error of the output is less than $1/m^*$ is greater than $1 - 1/2 = 1/2$. Since there are no examples with probability strictly between 0 and $1/m^*$, this implies that the probability that the output agrees exactly with the training sample is greater than $1/2$.

The procedure described in the previous paragraph is the basis for a randomised algorithm L^* for finding a hypothesis which agrees with the given training sample $\mathbf{s}^*$. In summary, L^* consists of the following steps.

- Evaluate $m_L = m_L(n, 1/2, 1/m^*)$.
- Construct, as described, a sample $\mathbf{s}$ of length m_L, according to μ.
- Run the given PAC learning algorithm L on $\mathbf{s}$.
- Check $L(\mathbf{s})$ explicitly to determine whether or not it agrees with $\mathbf{s}^*$.
- If $L(\mathbf{s})$ does not agree with $\mathbf{s}^*$, output *no*. If it does, output $L(\mathbf{s})$.

As we noted, the PAC property of L ensures that L^* succeeds with probability greater than $1/2$. Finally, it is clear that, since the running time of L is polynomial in m and its sample complexity $m_L(n, 1/2, 1/m^*)$ is polynomial in n and $m^* = 1/\epsilon$, the running time of L^* is polynomial in n and m^*. □

This result enables us to move from hardness results for the consistency problem to hardness results for PAC learning. Recall that RP denotes the class of problems which can be solved by polynomial-time randomised algorithms.

Theorem 8.2 *Suppose $H = \bigcup H_n$ and the H-*CONSISTENCY *problem is NP-hard. Then, unless RP equals NP, there is no PAC learning algorithm for H which runs in time polynomial in ϵ^{-1} and n.*

The fact that computational complexity-theoretic hardness results hold for neural networks was first shown by Judd [66]. In this section we shall prove a simple hardness result from [10, 11] along the lines of one due to Blum and Rivest [36].

The network has n inputs and $k+1$ computation units ($k \geq 1$). The first k computation units are 'in parallel' and each of them is connected to all the inputs. The last computation unit is the output unit; it is connected by arcs with fixed weight 1 to the other computation units, and it has fixed threshold k. The effect of this arrangement is that the output unit acts as a multiple AND gate for the outputs of the other computation units. We shall refer to this network as $\mathcal{N}_n^k$.

A state ω of $\mathcal{N}_n^k$ is described by the thresholds θ_l ($1 \leq l \leq k$) of the first k computation units and the weights w_{il} on the arcs (i,l) linking the inputs to the computation units. We shall use the notation $w^{(l)}$ for the n-vector of weights on the arcs terminating at l, so that $w_i^{(l)} = w_{il}$.

We shall prove that the consistency problem for $\mathcal{N}^k = \bigcup \mathcal{N}_n^k$ is NP-hard (provided $k \geq 3$), by a reduction from *GRAPH k-COLORING*. Let G be a graph with vertex-set $V = \{1, 2, \ldots, n\}$ and edge-set E. We construct a training sample $\mathbf{s}(G)$ as follows. For each vertex $i \in V$ we take as a negative example the vector v_i which has 1 in the ith coordinate position, and 0's elsewhere. For each edge $ij \in E$ we take as a positive example the vector $v_i + v_j$. We also take the zero vector $o = 00\ldots0$ to be a positive example.

Theorem 8.3 *Let G be a graph. There is a function in $\mathcal{N}_n^k$ which is consistent with $\mathbf{s}(G)$ if and only if G is k-colorable.*

Proof Suppose h is computable by $\mathcal{N}_n^k$ and is consistent with the training sample. By the construction of the network, h is a conjunction $h = h_1 \wedge h_2 \wedge \ldots \wedge h_k$ of linear threshold functions. (That is, $h(x) = 1$ if and only if $h_i(x) = 1$ for all i between 1 and k.) Specifically, there are weight-vectors $w^{(1)}, w^{(2)}, \ldots, w^{(k)}$ and thresholds $\theta_1, \theta_2, \ldots, \theta_k$ such that

$$h_l(y) = 1 \iff \langle w^{(l)}, y \rangle \geq \theta_l \quad (1 \leq l \leq k).$$

Note that, since o is a positive example, we have $0 = \langle w^{(l)}, o \rangle \geq \theta_l$ for each l between 1 and k. For each vertex i, $h(v_i) = 0$, and so there is at least one function h_f ($1 \leq f \leq k$) for which $h_f(v_i) = 0$. Thus we may define $\chi : V \to \{1, 2, \ldots, k\}$ by

$$\chi(i) = \min\{f : h_f(v_i) = 0\}.$$

It remains to prove that χ is a coloring of G. Suppose that $\chi(i) = \chi(j) = f$, so that $h_f(v_i) = h_f(v_j) = 0$. In other words,

$$\langle w^{(f)}, v_i \rangle < \theta_f, \quad \langle w^{(f)}, v_j \rangle < \theta_f.$$

Then, recalling that $\theta_f \leq 0$, we have $\langle w^{(f)}, v_i + v_j \rangle < \theta_f + \theta_f \leq \theta_f$. It follows that $h_f(v_i + v_j) = 0$ and $h(v_i + v_j) = 0$. Now if ij were an edge of G, then we should have $h(v_i + v_j) = 1$, because we assumed that h is consistent with the training sample. Thus ij is not an edge of G, and χ is a coloring, as claimed.

Conversely, suppose we are given a coloring $\chi : V \to \{1, 2, \ldots, k\}$. For $1 \leq l \leq k$ define the weight-vector $w^{(l)}$ as follows: $w_i^{(l)} = -1$ if $\chi(i) = l$ and $w_i^{(l)} = 1$ otherwise. Define the threshold θ_l to be $-1/2$. Let $h_1, h_2, \ldots, h_k$ be the corresponding linear threshold functions, and let h be their conjunction. We claim that h is consistent with $\mathbf{s}(G)$. Since $0 \geq \theta_l = -1/2$ it follows that $h_l(o) = 1$ for each l, and so $h(o) = 1$. In order to evaluate $h(v_i)$, note that if $\chi(i) = f$ then $\langle w^{(f)}, v_i \rangle = w_i^{(f)} = -1 < -1/2$, so $h_f(v_i) = 0$ and $h(v_i) = 0$, as required. Finally, for any color l and edge ij we know that at least one of $\chi(i)$ and $\chi(j)$ is not l. Hence $\langle w^{(l)}, v_i + v_j \rangle = w_i^{(l)} + w_j^{(l)}$, where either both of the terms on the right-hand side are 1, or one is 1 and the other is -1. In any case the sum exceeds the threshold $-1/2$, and $h_l(v_i + v_j) = 1$. Thus $h(v_i + v_j) = 1$. □

The proof that the decision problem for consistency in $\mathcal{N}^k$ is NP-hard for $k \geq 3$ follows directly from this result. We have shown that if there is a polynomial time algorithm for $\mathcal{N}^k$*−CONSISTENCY*, then there is one for *GRAPH k-COLORING*. But *GRAPH k-COLORING* is NP-complete [47], and hence it follows that the $\mathcal{N}^k$*−CONSISTENCY* problem is NP-hard if $k \geq 3$. (In fact, the same is true if $k = 2$. This follows from work of Blum and Rivest [36].)

Thus, fixing k, we have a very simple family of feedforward linear threshold networks, each consisting of $k+1$ computation units (one of which is 'hard-wired' and acts simply as an AND gate) for which the

problem of 'loading' a training sample is computationally intractable. Theorem 8.2 enables us to move from this hardness result for the consistency problem to a hardness result for PAC learning. The theorem tells us that, since $\mathcal{N}^k - CONSISTENCY$ is NP-hard, unless RP equals NP, there can be no computationally efficient PAC learning algorithm for the graded space $\mathcal{N}^k = \bigcup \mathcal{N}_n^k$ when $k \geq 2$.

We have only considered here the complexity of learning when the concept space and the hypothesis space are the same (that is, when $C = H$). Although the above results shows that it is intractable to learn $\mathcal{N}^k$ if the hypothesis produced must also be in $\mathcal{N}^k$, it is still unknown whether or not the class can be learned using a larger hypothesis space (for example, a family of larger neural networks, such that the nth network can compute all the functions that $\mathcal{N}^k$ can). In this sense, the hardness result just given is 'representation-dependent' [93]. Hardness results of a 'representation-independent' nature, in which the only specification on the hypothesis space is the reasonable one that it be 'polynomially evaluatable', have been obtained using assumptions about the intractability of certain problems arising in cryptography; see, for example [72].

PART 2: VARIANTS AND EXTENSIONS

There are a number of limitations to the applicability of the basic PAC model and many variants have been studied in recent years in an attempt to generalize and extend the theory. In this part of the article, we discuss some of these. A fuller treatment of these and other variants will be found in the forthcoming book [8].

9 Stochastic Concepts

The results presented so far have nothing to say if there is some form of 'noise' present during the learning procedure. Further, the basic model applies only to the learning of functions: each example is either a positive example or a negative example of the given target concept, not both. But one can envisage situations in which the 'teacher' has difficulty classifying some examples, so that the labeled examples presented to the 'learner' are not labeled by a function, the same example being on occasion classified by the 'teacher' as a positive example and on other occasions (possibly within the same training sample) as a negative example. For example, in the context of machine vision, if the concept is a geometrical figure then points close to the boundary of the figure may be difficult for the teacher to classify, sometimes being classified as positive and sometimes as negative. Alternatively, the problem may not lie with the teacher, but with the 'concept' itself. This may be ill-formed and may not be a function at all.

To deal with these situations, we have the notion of a *stochastic concept* [70, 119, 37]. A stochastic concept on X is simply a probability distribution P on $X \times \{0,1\}$. Informally, for finite or countable X, one interprets $P((x,b))$ to be the probability that x will be given classification b. This can be specialised to give the standard PAC model, as we saw in section 5. Suppose we have a probability distribution μ on X and a target concept t. Then recall that there is a probability distribution P on $X \times \{0,1\}$ such that for all measurable subsets A of X,

$$P\left(\{(x,t(x)) : x \in A\}\right) = \mu(A); \quad P\left(\{(x,b) : x \in A, b \neq t(x)\}\right) = 0.$$

In this case, we say that the stochastic concept P *corresponds* to t and μ. What can be said about 'learning' a stochastic concept by means of a hypothesis space H of $\{0,1\}$-valued functions? The error of $h \in H$ with respect to the target stochastic concept is the probability

$$\mathrm{er}_P(h) = P\left(\{(x,b) \in X \times \{0,1\} : h(x) \neq b\}\right)$$

of misclassification by h of a further randomly drawn training example. This is, as earlier, precisely $P(E_h)$, where $E_h = \{(x,b) : h(x) \neq b\}$ is the *error set* of h. If P is truly stochastic (and not merely the stochastic

representation of a function, as described above) it is unlikely that this error can be made arbitrarily small. As earlier, the *observed error* of h on a training sample $\mathbf{s} = ((x_1, b_1), (x_2, b_2), \ldots, (x_m, b_m))$ is defined to be

$$\mathrm{er}_{\mathbf{s}}(h) = \frac{1}{m} |\{i : h(x_i) \neq b_i\}|.$$

Clearly this may be non-zero for all $h \in H$ (particularly if the same example occurs twice in the sample, but with different labels). To discuss learning in this context, one minor modification of the definition of a learning algorithm is required. In this context, a learning algorithm (using hypothesis space H) is a mapping from the set $\bigcup_{m \geq 1} (X \times \{0,1\})^m$ to H. (Note that, previously, with a concept space C, L needed only to map from training samples for functions in C.) What we should like, informally speaking, is that there is some sample length m_L, independent of the stochastic concept P, such that if a hypothesis has 'small' observed error with respect to a random sample of length at least m_L then, with high probability, it has 'small' error with respect to P.

The following result was presented in section 5 as part of Theorem 5.2. We present it here again for convenience. This result is a statistical result, in the sense that it does not concern the performance of a particular learning algorithm. However, it implies, roughly speaking, that if a leaning algorithm manages to produce an output hypothesis with small error, then that hypothesis is likely to be correct on most other inputs.

Theorem 9.1 *Let H be a hypothesis space of $\{0,1\}$-valued functions defined on an input space X and suppose that H has finite VC-dimension. Let P be any stochastic concept on X. Suppose that $0 < \delta, \epsilon < 1$ and $0 < \gamma \leq 1$. If*

$$m \geq \frac{8}{\gamma^2 \epsilon} \left(\ln \left(\frac{4}{\delta} \right) + \mathrm{VCdim}(H) \ln \left(\frac{48}{\gamma^2 \epsilon} \right) \right)$$

then, with P^m-probability at least $1 - \delta$, a random sample $\mathbf{s}$ from $(X \times \{0,1\})^m$ satisfies

$$\mathrm{er}_{\mathbf{s}}(h) \leq (1 - \gamma)\epsilon \Longrightarrow \mathrm{er}_P(h) \leq \epsilon.$$

The notion of a stochastic concept can be applied to a number of situations. As already indicated, it can be useful when the target 'concept' is not a function. It can also be useful when there is 'classification noise' (see [4]), that is, where there is an underlying target function, but the randomly chosen examples have their labels 'flipped' occasionally. This corrupts the training data and results in a stochastic concept. Additionally, in the standard PAC model, we have assumed that the concept space is a subset of the hypothesis space. Suppose this is not so and $t \in C \setminus H$. Then there can be no $h \in H$ such that the error of h with respect to t is 0 for all probability distributions μ on X. However, Theorem 9.1 is applicable. Suppose μ is a distribution on X and take P to be the stochastic concept corresponding to t and μ. Since the sample length given in Theorem 9.1 is independent of the stochastic concept P, we obtain a type of learnability result when H has finite VC-dimension: there is a sample length $m_0(\delta, \epsilon)$ such that if a randomly drawn training sample of t of length m_0 is presented, then with probability at least $1 - \delta$, if $h \in H$ is correct on a fraction of at least $1 - \epsilon/2$ of the sample, then h has error at most ϵ. (Here, we have taken $\gamma = 1/2$ for simplicity.)

In general, suppose that P is a stochastic concept on X and that H is a hypothesis space on X. Let

$$\mathrm{opt}_H(P) = \inf_{h \in H} \mathrm{er}_P(h),$$

which is a measure of how well the stochastic concept can be approximated by functions in H. We make the following definitions.

Definition 9.2 (Probably approximately optimal algorithm) *Suppose that H is a hypothesis space on X. We say that a learning algorithm L is a* probably approximately optimal algorithm for H *if for any $0 < \delta, \epsilon < 1$, there is $m_L(\delta, \epsilon)$ such that for $m \geq m_L(\delta, \epsilon)$, the following holds for any stochastic concept P on X: if $\mathbf{s} \in (X \times \{0,1\})^m$ is randomly drawn then with probability at least $1 - \delta$,*

$$\mathrm{er}_P(L(\mathbf{s})) < \mathrm{opt}_H(P) + \epsilon.$$

Definition 9.3 (UCE property) *A set H of functions from X to $\{0,1\}$ has the* uniform convergence of errors *(*UCE*) property if the following holds. Given real numbers δ and ϵ ($0 < \delta, \epsilon < 1$), there is a positive integer $m_0(\delta, \epsilon)$ such that, for any probability distribution P on $S = X \times \{0,1\}$, if $m \geq m_0(\delta, \epsilon)$,*

$$P^m\left(\{\mathbf{s} \in S^m : \text{ for all } h \in H, |\mathrm{er}_P(h) - \mathrm{er}_{\mathbf{s}}(h)| < \epsilon\}\right) > 1 - \delta.$$

Thus, roughly speaking, H has the UCE property if one can guarantee with high confidence that the observed errors of functions in H on a sample of large enough length are close to their actual errors, for all stochastic concepts. Results of Vapnik and Chervonenkis [116] on the uniform convergence of relative frequencies to probabilities show that H has the UCE property if and only if H has finite VC-dimension. (The fact that the VC-dimension is sufficient follows from Theorem 5.1.)

It is easy to see that if H has the UCE property then there is a probably approximately optimal learning algorithm for H—the algorithm which returns the hypothesis which minimises the observed error.

Theorem 9.4 *Suppose that hypothesis space H has the UCE property. Let the learning algorithm $L_{\min}$ be defined as follows: On input $\mathbf{s} \in (X \times \{0,1\})^m$, $L_{\min}$ returns a hypothesis $L_{\min}(\mathbf{s}) \in H$ which minimises the observed error $\mathrm{er}_{\mathbf{s}}(h)$. Then $L_{\min}$ is a probably approximately optimal learning algorithm for H.*

Proof Let P be any stochastic concept. Since $\mathrm{opt}_H(P) = \inf_{h \in H} \mathrm{er}_P(h)$, there is $h^* \in H$ such that $\mathrm{er}_P(h^*) < \mathrm{opt}_H(P) + \epsilon/3$. Let δ, ϵ be given. Since H has the UCE property, provided $m \geq m_0(\delta, \epsilon/3)$, for $\mathbf{s} \in (X \times \{0,1\})^m$, with probability at least $1-\delta$, $\mathrm{er}_P(h) < \mathrm{er}_{\mathbf{s}}(h) + \epsilon/3$ for all $h \in H$. Suppose that $h_{\min}$ is a hypothesis with minimal observed error on $\mathbf{s}$. Then, with probability at least $1-\delta$, $\mathrm{er}_P(h_{\min}) < \mathrm{er}_{\mathbf{s}}(h_{\min}) + \epsilon/3$ and $\mathrm{er}_P(h^*) < \mathrm{er}_{\mathbf{s}}(h^*) + \epsilon/3$. Using the fact that $\mathrm{er}_{\mathbf{s}}(h_{\min}) \leq \mathrm{er}_{\mathbf{s}}(h^*)$, we therefore have

$$\mathrm{er}_P(h_{\min}) < \mathrm{er}_{\mathbf{s}}(h_{\min}) + \epsilon/3 \leq \mathrm{er}_{\mathbf{s}}(h^*) + \epsilon/3 < \mathrm{er}_P(h^*) + 2\epsilon/3 < \mathrm{opt}_H(P) + \epsilon,$$

with probability at least $1-\delta$. It follows that $L_{\min}$ is probably approximately optimal, with sample complexity at most $m_0(\delta, \epsilon/3)$. □

Of course, finding a hypothesis in H with minimal observed error may be a computationally intractable problem. In order to overcome the computational complexity problem, it is sometimes convenient to seek to produce a hypothesis from a *larger* class H', which does not necessarily have near-optimal error among hypotheses from H', but performs well with respect to H. This model of learning, in which the aim is to produce, from a class H' with $H' \supseteq H$, an output hypothesis only slightly worse than the best approximation in H, was introduced by Kearns, Schapire and Sellie [71], who called it *agnostic learning*.

10 Distribution-Specific Learning

Perhaps the main attraction of the definition of PAC learning is the 'distribution-free' criterion: the sample complexity is independent of the probability distribution. The proofs of the standard computational hardness results for PAC learning, and the lower bounds on sample complexity, involve the use of very particular probability distributions, so the theory presented earlier is very dependent on this criterion. If we know in advance what the distribution on the examples is, or if we know that it is one of a particular set of distributions, then the full strength of the PAC definition is not needed. Let us suppose that the concept space and hypothesis space are both equal to H and that $\mathcal{P}$ is a class of probability distributions on X. We may say that a learning algorithm L for H *learns with respect to* $\mathcal{P}$ if there is a sample length function $m_L(\delta, \epsilon)$ such that for any $t \in H$ and *any* $\mu \in \mathcal{P}$, if $m \geq m_L(\delta, \epsilon)$, then with μ^m-probability at least $1-\delta$, if a training sample $\mathbf{s}$ is presented, $\mathrm{er}_\mu(L(\mathbf{s})) < \epsilon$. This is a weakening of the PAC criterion, in that m_L need exist and be uniform only over $\mathcal{P}$. In general, finite VC-dimension is not necessary for such 'distribution-specific' learning. (Note, however, that the theory of previous sections shows that it *is* necessary when $\mathcal{P}$ consists of all distributions.) Thus, it may be possible to learn with respect to a particular class of distributions even when PAC learning is not possible. Furthermore, learning with respect to a particular distribution or class of distributions may be computationally much easier than PAC learning; see [45, 117, 44, 68, 92, 50, 78, 46].

The original statistical work of Vapnik and Chervonenkis [116] has something to say about learning with respect to particular distributions. Let us consider the case $\mathcal{P} = \{\mu\}$, in which $\mathcal{P}$ consists of just one distribution. The results of Vapnik and Chervonenkis (see also [88]) imply that H is learnable with respect to μ, by *any* consistent learning algorithm, if

$$\frac{\mathbf{E}_{\mu^m}(\log \Pi(\mathbf{x}))}{m} \to 0 \text{ as } m \to \infty,$$

where $\mathbf{E}_{\mu^m}(.)$ denotes expected value with respect to the distribution μ^m.

Learnability with respect to a particular distribution has also been studied by Benedek and Itai [33, 32]. They show that if μ is discrete—that is, if the support of μ is a countable subset of X—then any hypothesis space H is learnable with respect to μ. For general distributions, they develop a theory involving 'ϵ-covers'. In order to describe this, we first need the notion of an *ϵ-cover* of a subset of a pseudo-metric space. A *pseudo-metric* ∂ on a set A is a function from $A \times A$ to $\mathbf{R}$ such that

$$\partial(a,b) = \partial(b,a) \geq 0, \quad \partial(a,a) = 0, \quad \partial(a,b) \leq \partial(a,c) + \partial(c,b)$$

for all $a, b, c \in A$. An *ϵ-cover* for a subset W of A is a subset S of A such that for every $w \in W$, there is some $s \in S$ such that $\partial(w,s) \leq \epsilon$. W is said to be *totally-bounded* if it has a finite ϵ-cover for all $\epsilon > 0$. When W is totally bounded, we denote by $N(\epsilon, W, \partial)$ the size of the smallest ϵ-cover for W, known as the *covering number*. Now, the probability measure μ induces a pseudo-metric ∂_μ on the hypothesis space H: we define

$$\partial_\mu(h,g) = \mu(\{x : h(x) \neq g(x)\}).$$

Benedek and Itai show that H is learnable with respect to μ if and only if the pseudo-metric space (H, ∂_μ) is totally bounded. (They use the term 'finitely coverable'.) For simplicity of notation, let us denote by $N_\mu(\epsilon)$ the covering number $N(\epsilon, H, \partial_\mu)$. Benedek and Itai show that any algorithm for learning H with respect to μ has sample complexity at least $\log((1-\delta)N_\mu(2\epsilon))$ and that, further, if (H, ∂_μ) is totally bounded, then there is an algorithm for learning H with respect to μ having sample complexity bounded by

$$\frac{54}{\gamma}\left(\log\left(\frac{1}{\gamma}\right) + \ln(N_\mu(\gamma/2))\right),$$

where γ is the smaller of δ, ϵ. (We remark that, unlike standard PAC learning, it is not the case that any consistent learning algorithm will do.)

The results of Benedek and Itai mentioned above show that a necessary and sufficient condition for H to be learnable with respect to a distribution μ is that $N_\mu(\epsilon) < \infty$ for all $\epsilon > 0$. It follows that if $\mathcal{P}$ is a class of probability distributions and H is learnable with respect to $\mathcal{P}$ then for any $\epsilon > 0$, the set $\{N_\mu(\epsilon) : \mu \in \mathcal{P}\}$ is bounded. Benedek and Itai conjectured that this condition is also sufficient for learnability with respect to $\mathcal{P}$. However, by means of a counterexample, Dudley *et al.* [41] proved that this is not so.

We discussed the efficiency of (distribution-independent) learning earlier, in which we required that a learning algorithm for a graded space run in time polynomial in n, the size of the examples, and in ϵ^{-1}. Let us concentrate on a single hypothesis space H, ungraded by example size. To be efficient, a learning algorithm for H should have running time which does not vary too dramatically as the parameters ϵ and δ are decreased. Therefore we should desire a learning algorithm to have sample complexity polynomial in $1/\epsilon$ and $1/\delta$. In the standard PAC model, this is always achievable: H is PAC learnable if and only if it has finite VC-dimension, in which case any consistent learning algorithm is PAC and has polynomial sample complexity, from the standard bounds. The corresponding situation in distribution-specific learning is not so easy. We now address the question of what conditions ensure that H is learnable with respect to μ with a sample complexity polynomial in $1/\epsilon$ and $1/\delta$. It follows immediately from the bounds of Benedek and Itai that a necessary and sufficient condition is that $N_\mu(\epsilon) < 2^{p(1/\epsilon)}$ for some polynomial p, but this is not a very manageable condition. Bertoni *et al.* [35] took a different approach, following on from the work of Vapnik

and Chervonenkis. For $\mathbf{x} = (x_1, \ldots, x_m) \in X^m$, let $C_m(\mathbf{x})$ be the size of the largest subset of $\{x_1, \ldots, x_m\}$ shattered by H. Bertoni *et al.* show that if there is a positive constant β such that

$$\mathbf{E}_{\mu^m}\left(\frac{C_m(\mathbf{x})}{m}\right) = O(m^{-\beta}),$$

then *any* consistent learning algorithm for H learns H with respect to μ and has polynomial sample complexity. (Note that this is a stronger conclusion than described above, since it says that *all consistent* learning algorithms learn and have sample complexity polynomial in the relevant parameters, rather than simply that there is *some* efficient learning algorithm.) Anthony and Shawe-Taylor [5, 17] found a further sufficient condition for polynomial sample complexity. Let us say that a sequence $\{S_k\}_{k=1}^{\infty}$ of subsets of X is *non-decreasing* if for each k, $S_k \subseteq S_{k+1}$. For $S \subseteq X$, let $H|S$ denote the set of functions in H restricted to domain S. Suppose that $X = \cup_{k=1}^{\infty} S_k$, where $\{S_k\}$ is non-decreasing and $\mathrm{VCdim}(H|S_k) \leq k$. Anthony and Shawe-Taylor [5, 17] proved that if the probability distribution μ satisfies

$$1 - \mu(S_k) = O(k^{-\beta})$$

for some $\beta > 0$, then any consistent learning algorithm for H learns with respect to μ and has polynomial sample complexity.

For further discussion of distribution-dependent learning, we refer the reader to the papers of Benedek and Itai [34], Ben-David, Benedek and Mansour [29], Bertoni *et al.* [35], Kharitonov [74], Li and Vitanyi [77], Linial, Mansour and Nisan [78]. For discussion specific to neural networks, see [52, 86].

11 Graph Dimension and Multiple-Output Nets

The basic PAC model concerns learning $\{0,1\}$-valued functions only; that is, it is concerned only with classification problems. A significant and important extension of the basic PAC model is to the learning of general function spaces.[10]

In this section we discuss the *graph dimension* and an application to artificial neural networks with more than one binary output unit. Thus far, the only networks we have been able to consider have a single binary output.

11.1 The graph dimension

We start with a very general framework. Suppose that C, H are sets of functions from an example space X into a set Y (not necessarily $\{0,1\}$) with $C \subseteq H$, and suppose that $t \in C$. Suppose also that there is a probability distribution μ on X. Generalising in the obvious way from our previous definitions, we may define the *error* of $h \in H$ with respect to t to be

$$\mathrm{er}_{\mu}(h,t) = \mu(\{x \in X : h(x) \neq t(x)\}).$$

That is, h is erroneous on example x if $h(x) \neq t(x)$. When $Y = \mathbf{R}$, for example, this may seem a little coarse; we shall later discuss an alternative approach. With this measure of error, we may define PAC learning as earlier.

For $h \in H$, let $\mathcal{G}h$ be the function from $X \times Y$ to $\{0,1\}$ defined by

$$\mathcal{G}h(x,y) = 1 \Longleftrightarrow h(x) = y.$$

We call $\mathcal{G}h$ the *graph* of h. Further, let $\mathcal{G}H = \{\mathcal{G}h : h \in H\}$, the *graph space* of H. We have the following definition [89].

[10] In what follows, certain technical measure-theoretic conditions have to be satisfied; we shall not discuss these, but the reader may find the details in the paper of Haussler [55] or the book by Pollard [95].

Definition 11.1 *For a set H of functions from X to Y, define the* graph dimension *of H, denoted* $\mathrm{gdim}(H)$, *is the VC-dimension of the set $\mathcal{G}H$ of $\{0,1\}$-valued functions on $X \times Y$.*

Observe that if $Y = \{0,1\}$, the graph dimension and the VC-dimension are equal. Now, it can be shown that if the hypothesis space $\mathcal{G}H$ is PAC learnable (in the usual sense), then so too is H (in the generalized sense), by any consistent learning algorithm. We have the following result, which has been phrased in terms of a probability distribution P on $X \times Y$ rather than in terms of a probability distribution μ on X and a target function $t \in C$. However, as mentioned earlier for the case when $Y = \{0,1\}$, the notion of a probability distribution on $X \times Y$—a 'stochastic function'—includes the situation where one has μ on X and $t \in C$. For $\mathbf{s} = ((x_1, y_1), \ldots, (x_m y_m)) \in (X \times Y)^m$, the *observed error* of $h \in H$ on $\mathbf{s}$ is defined to be

$$\mathrm{er}_{\mathbf{s}}(h) = \frac{1}{m} \left| \{ i : h(x_i) \neq y_i \} \right| .$$

Theorem 11.2 *Let $0 < \epsilon < 1$ and $0 < \gamma \leq 1$. Suppose H is a hypothesis space of functions from an input space X to a set Y, and let P be any probability measure on $S = X \times Y$. Then the probability (with respect to P^m) that, for $\mathbf{x} \in S^m$, there is some $h \in H$ such that*

$$\mathrm{er}_P(h) > \epsilon \quad \text{and} \quad \mathrm{er}_{\mathbf{x}}(h) \leq (1-\gamma)\mathrm{er}_P(h)$$

is at most

$$4\,\Pi_{\mathcal{G}H}(2m) \exp\left(-\frac{\gamma^2 \epsilon m}{4} \right) .$$

Furthermore, if H has finite graph dimension, this quantity is less than δ for

$$m = \frac{8}{\gamma^2 \epsilon} \left(\ln\left(\frac{4}{\delta} \right) + \mathrm{gdim}(H) \ln\left(\frac{48}{\gamma^2 \epsilon} \right) \right) .$$

Proof Define the error set E_h of $h \in H$ by

$$E_h = \{ (x, y) \in X \times Y : h(x) \neq y \} .$$

Observe that $E_h = (X \times Y) \setminus \mathcal{G}(h)$. The proof of Theorem 5.1 can easily be modified[11] to obtain the bound

$$P^m \left\{ \mathbf{s} \in S^m : \exists h \in H \text{ with } \frac{P(E_h) - \mathrm{er}_{\mathbf{s}}(h)}{\sqrt{P(E_h)}} > \eta \right\} \leq 4\,\Pi_{\mathcal{G}H}(2m) \exp\left(-\frac{1}{4}\eta^2 m \right) ,$$

for $\eta > 0$, where $S = X \times Y$. The result follows from this. □

In the same way as Theorem 5.2 implies Theorem 6.1, this result tells us that if H has finite graph dimension and $C \subseteq H$, then any consistent (C, H)-learning algorithm is PAC. The result also gives an upper bound on the sample complexity of any consistent algorithm.

We see from the above that if $\mathrm{gdim}(H)$ is finite then H is PAC learnable (by H) by any consistent algorithm. It is natural to ask whether finite graph dimension is a necessary condition for learnability in this generalized model. Natarajan showed that it is not: there are PAC learnable function spaces with infinite graph dimension (see Natarajan [90]). Natarajan finds a weaker necessary condition for learnability, showing that a certain measure, now known as the Natarajan dimension, must be finite for H to be PAC learnable. More recently, Ben-David, Cesa-Bianchi and Long [31, 30] have shown that when Y is *finite*, the finiteness of the graph dimension is a necessary *and* sufficient condition for H to be PAC learnable. Furthermore,

[11] In fact, the original result of Vapnik was quite a bit more general than our Theorem 5.1, concerning the relative uniform deviation of relative frequencies from probabilities over a class of events. This result follows easily from the more general form of Vapnik's result when one takes the events to be the error sets. Details may be found in [5, 19, 37].

they show that in this case, the Natarajan dimension is finite if and only if the graph dimension is finite, so that Natarajan's necessary and sufficient conditions are matching. For the case of finite Y, Ben-David *et al.* present a characterisation of the 'dimension' measures whose finiteness is equivalent to the learnability of H. An interesting consequence of their results may be described as follows. Suppose that H maps into the finite set Y and, for each $h \in H$ and each $y \in Y$, let $h^y : X \to \{0,1\}$ be defined by $h^y(x) = 1 \Longleftrightarrow h(x) = y$. Then, the function space H is learnable if and only if each of the spaces $H^y = \{h^y : h \in H\}$ is learnable; that is, if and only if each of these 'projection' spaces has finite VC-dimension.

11.2 Multiple-output feedforward threshold networks

We now briefly consider feedforward threshold networks with more than one binary output unit. We shall obtain a bound on the growth function of the graph space of such a network and thereby obtain an upper bound on the graph dimension. We do not explicitly present a bound on the sample complexity of a consistent learning algorithm as a PAC algorithm, but such a bound could be derived either from the graph dimension bound or (better) from the bound on the growth function (see [19, 104, 103]).

We first note that there is another way of describing the notion of graph dimension. For $\mathbf{y} = (y_1, \ldots, y_m) \in Y^m$, let $I_{\mathbf{y}} : Y^m \to \{0,1\}^m$ be defined by

$$I_{\mathbf{y}}((z_1, \ldots, z_m)) = (a_1, \ldots, a_m), \quad \text{where} \quad a_i = 1 \Longleftrightarrow y_i = z_i.$$

For $\mathbf{x} = (x_1, \ldots, x_m) \in X^m$ and $h \in H$, define $\mathbf{x}^*(h) = (h(x_1), \ldots, h(x_m))$. This defines a mapping $\mathbf{x}^*$ from H to Y^m. For each $\mathbf{y} \in Y^m$, the composition $I_{\mathbf{y}} \circ \mathbf{x}^*$ is a mapping from H to the finite set $\{0,1\}^m$. We define $\Delta_H(\mathbf{x})$ to be the maximum, as $\mathbf{y}$ ranges over Y^m, of $|I_{\mathbf{y}} \circ \mathbf{x}^*(H)|$, the cardinality of the image of H under $I_{\mathbf{y}} \circ \mathbf{x}^*$. Further, we let $\Delta_H(m)$ be the maximum of $\Delta_H(\mathbf{x})$ over all $\mathbf{x} \in X^m$. Then, clearly, $\Delta_H(m) = \Pi_{\mathcal{G}H}(m)$, and therefore the graph dimension of H (is either infinite, or) is the largest integer d such that $\Delta_H(d) = 2^d$. Notice that for finite Y,

$$\Delta_H(\mathbf{x}) \leq |\mathbf{x}^*(H)| \leq , _H(m),$$

where $, _H(m)$ is the maximum over all $\mathbf{x} \in X^m$ of $|\mathbf{x}^*(H)|$. It follows that if one can bound the quantity $, _H(m)$, then a bound on the growth function of the graph space, and hence on the graph dimension, can be obtained. This is the technique used in obtaining the following result. (See [104, 19, 103] for improvements on this.)

Theorem 11.3 *Suppose that $\mathcal{N}$ is a feedforward linear threshold network having W variable parameters (weights and thresholds) and any number of output units. Let H be the set $H_{\mathcal{N}}$ of functions computable by $\mathcal{N}$. Then, for $m > W$, $\Pi_{\mathcal{G}H}(m) < m^{2W}$ and the graph dimension of the network is less than $6W \log W$.*

Proof The bounds here are exactly the same as those obtained in Theorem 7.6. One uses a proof similar to the proof of that result, obtaining an upper bound on $, _H(m)$ of m^{2W} for $m > W$. The same proof technique works since each computational unit computes a $\{0,1\}$-valued function. One just has to be aware that counting in this manner does indeed provide a bound on $, _H(m)$. □

12 Pseudo-Dimension and Function Learning

12.1 The pseudo-dimension

We have seen that the graph dimension can be used to measure the expressive power of a hypothesis space of functions, in somewhat the same way as the VC-dimension is used for $\{0,1\}$-valued hypothesis spaces. But there are other such measures. In passing, we have already mentioned the Natarajan dimension. We now introduce a very useful dimension, known as the pseudo-dimension. This was introduced by Pollard [95] and is defined whenever the set of functions maps into $Y \subseteq \mathbf{R}$. (More generally, it may be defined when Y

is any totally ordered set, but this shall not concern us here.) Let H be a set of functions from X to $\mathbf{R}$. For any $\mathbf{x} = (x_1, x_2, \ldots, x_m) \in X^m$, and for $h \in H$, let

$$\mathbf{x}^*(h) = (h(x_1), h(x_2), \ldots, h(x_m))$$

and let $\mathbf{x}^*(H) = \{\mathbf{x}^*(h) : h \in H\}$. We say that $\mathbf{x}$ is pseudo-shattered by H if some translate $\mathbf{r} + \mathbf{x}^*(H)$ of $\mathbf{x}^*(H)$ intersects all orthants of $\mathbf{R}^m$. The *pseudo-dimension* of H, denoted pdim(H), is either infinite or is the largest length of a pseudo-shattered sample. The pseudo-dimension is often called the *combinatorial dimension* or *Pollard dimension*. We state the definition formally in a rather more explicit way.

Definition 12.1 (Pseudo-dimension) *Let H be a set of functions from X to $\mathbf{R}$ and let $\mathbf{x} \in X^m$. We say $\mathbf{x}$ is* pseudo-shattered *by H if there are $r_1, r_2, \ldots, r_m \in \mathbf{R}$ such that for any $\mathbf{b} \in \{0,1\}^m$, there is $h_\mathbf{b} \in H$ with*

$$h_\mathbf{b}(x_i) \geq r_i \iff b_i = 1.$$

The largest d such that some sample of length d is pseudo-shattered is the pseudo-dimension *of H, denoted* pdim(H). *(When this maximum does not exist, the pseudo-dimension is taken to be infinite.)*

Note that when $Y = \{0,1\}$, the definition of pseudo-dimension reduces to the VC-dimension. Furthermore, when H is a vector space of real functions, the pseudo-dimension of H is precisely the vector-space dimension of H; see [55]. (This generalizes Dudley's result, Theorem 4.2.)

12.2 Learning real-valued functions

When considering a space H of functions from X to $\mathbf{R}^k$, as is appropriate for applications to neural networks with k real outputs, it seems rather over-restrictive, and inappropriate, to say that a hypothesis h is erroneous with respect to a target t on example x unless $h(x)$ and $t(x)$ are precisely equal. Up to now, this is the definition of error we have used. There are other ways of measuring error, if one is prepared to ask not *is the output correct?* but *is the output close?* in some sense. Haussler [55] has developed a 'decision-theoretic' framework encompassing many ways of measuring error by means of *loss functions*. We shall describe this framework in a way which also subsumes the discussion on stochastic concepts.

First, we need some definitions. A *loss function* is, for our purposes, a non-negative bounded function $l : Y \times Y \to [0, M]$ (for some M). Informally, the loss $l(y, y')$ is a measure of how 'bad' the output y is, when the desired output is y'. An example of a loss function is the *discrete loss function*, defined by $l(y, y') = 1$ unless $y = y'$, in which case $l(y, y') = 0$. This is the loss function one uses to measure error in the standard PAC model and in the above discussion of graph dimension. It seems reasonable if the outputs are in $\{0,1\}^k$, for example, as occurs in a threshold network with k output units. However, other loss functions are more appropriate if the outputs are real numbers. One useful loss function is the *L^1-loss*, or *linear loss*, which is defined when $Y \subseteq \mathbf{R}^k$. This is given by

$$l(y, y') = \frac{1}{k} \sum_{i=1}^{k} |y_i - y'_i|.$$

In both of these examples, the loss function is actually a metric, but there is no need for this. For example, a loss function which is not a metric—and which is often used—is the *L^2-loss*, or *quadratic loss*, defined on $\mathbf{R}^k$ by

$$l(y, y') = \frac{1}{k} \sum_{i=1}^{k} (y_i - y'_i)^2.$$

There are many other useful loss functions, such as the *L^∞-loss*, the *logistic loss* and the *cross-entropy loss*. The reader is referred to the paper of Haussler [55] for a far more detailed discussion of the general decision-theoretic approach and its applications. As in our discussion of stochastic concepts, we consider probability

distributions P on $X \times Y$. Suppose that $l : Y \times Y \to [0, M]$ is a particular loss function. For $h \in H$, we define the *error* of h with respect to P (and l) to be

$$\mathrm{er}_{P,l}(h) = \mathbf{E}_P\left(l(h(x), y)\right) = \int_{X \times Y} l(h(x), y) dP(x, y),$$

the expected value of $l(h(x), y)$. When P is the stochastic concept corresponding to a target function t and distribution μ on X, then this error is $\mathbf{E}_\mu(l(h(x), t(x))$, the average loss in using h to approximate t. Note that if l is the discrete metric then this is simply the μ-probability that $h(x) \neq t(x)$, which is precisely the measure of error used in the standard PAC learning definition.

There is a slight difference between the function learning framework we shall describe and the standard learning framework for $\{0, 1\}$-valued functions as we have described it. We shall assume that a function learning algorithm is not simply given a training sample of adequate length, but is also given as part of its input the accuracy parameter ϵ. This is not really so different from the basic PAC learning framework: in the original PAC model as introduced by Valiant, a learning algorithm was given the accuracy and confidence parameters and had access to an 'oracle' generating random labeled examples. Subsequently, Haussler *et al.* [56] showed that this model is equivalent to the 'functional' model we described in earlier sections, in which the learning algorithm is given only a training sample as input.

Suppose that a sample $\mathbf{s} = ((x_1, y_1), \ldots, (x_m, y_m))$ of points from $X \times Y$ is given. The *observed error* (or *empirical loss*) of h on this sample is

$$\mathrm{er}_{\mathbf{s},l}(h) = \frac{1}{m} \sum_{i=1}^{m} l(h(x_i), y_i).$$

The aim of learning in this context is to find, on the basis of a 'large enough' sample $\mathbf{s}$, and the parameter ϵ, some $L(\epsilon, \mathbf{s}) \in H$ which has close to optimal error with respect to P. We arrive at the following definition.

Definition 12.2 (Probably approximately optimal learning of functions) *Suppose that H is a set of functions from X to Y and that l is a loss function defined on $Y \times Y$. A function*

$$L : (0, 1) \times \bigcup_{m \geq 1} (X \times Y)^m \to H$$

is said to be a probably approximately optimal learning algorithm *for H if for any $0 < \delta, \epsilon < 1$, there is $m_L(\delta, \epsilon)$ such that for $m \geq m_L(\delta, \epsilon)$, the following holds: For any probability distribution P on $X \times Y$, if $\mathbf{s} \in (X \times Y)^m$ is randomly drawn then, with probability at least $1 - \delta$,*

$$\mathrm{er}_{P,l}\left(L(\epsilon, \mathbf{s})\right) < \mathrm{opt}_H(P) + \epsilon,$$

where $\mathrm{opt}_H(P) = \inf_{h \in H} \mathrm{er}_{P,l}(h)$.

(As earlier, there is an 'agnostic learning' variant of this, in which the aim is to output a hypothesis from a larger hypothesis space which approximates to the target almost as well as the best approximation in H does; see [82].) As in the standard PAC model and the stochastic PAC model described earlier, this can be guaranteed provided we have a 'uniform convergence of errors' property. Extending the earlier definition, we say that a hypothesis space H of functions from X to Y has the *uniform convergence of errors* (*UCE*) property with respect to the loss function l if for $0 < \delta, \epsilon < 1$, there is a positive integer $m_0(\delta, \epsilon)$ such that, for any probability distribution P on $X \times Y$,

$$P^m\left(\{\mathbf{s} : \text{for all } h \in H, |\mathrm{er}_{P,l}(h) - \mathrm{er}_{\mathbf{s},l}(h)| < \epsilon\}\right) > 1 - \delta$$

for all $m \geq m_0$. If this is the case, then a learning algorithm which outputs a hypothesis with near-minimal observed error will be a probably approximately optimal learning algorithm. More precisely, we have the following result [55, 114].

Theorem 12.3 *Suppose that H is a hypothesis space of functions from X to Y and that l is a loss function defined on $Y \times Y$. Let $\hat{L}$ be any learning algorithm for H with the property that, given input consisting of ϵ and a sample $\mathbf{s}$,*

$$\mathrm{er}_{\mathbf{s},l}\left(\hat{L}(\epsilon, \mathbf{s})\right) \leq \inf_{h \in H} \mathrm{er}_{\mathbf{s},l}(h) + \epsilon/3.$$

If H has the UCE property with respect to l, then $\hat{L}$ is a probably approximately optimal learning algorithm for H. Furthermore, the sample complexity of $\hat{L}$ satisfies

$$m_{\hat{L}}(\delta, \epsilon) \leq m_0(\delta, \epsilon/3),$$

where m_0 is the sample length function in the definition of the UCE property.

Proof This is analogous to the proof of Theorem 9.4. Suppose that H has the UCE property and let $m > m_0(\delta, \epsilon/3)$ where $m_0(\delta, \epsilon)$ is the sample length function in the definition of the UCE property. Fix $\mathbf{s} \in (X \times Y)^m$ and let $\hat{h}$ denote $\hat{L}(\epsilon, \mathbf{s})$. Then, for all $h \in H$, we have

$$\mathrm{er}_{P,l}(\hat{h}) \leq \mathrm{er}_{\mathbf{s},l}(\hat{h}) + \epsilon/3 \leq \mathrm{er}_{\mathbf{s},l}(h) + 2\epsilon/3 \leq \mathrm{er}_{P,l}(h) + \epsilon,$$

so that

$$\mathrm{er}_{P,l}(\hat{h}) \leq \inf_{h \in H} \mathrm{er}_{P,l}(h) + \epsilon = \mathrm{opt}_H(P) + \epsilon.$$

The result follows. □

Note that, in this case, there need not be any function in H minimizing the observed error, since the errors can be real numbers and H may be infinite. (In other words, one has an *infimum* but not necessarily a *minimum*.) For this reason, we demand observed error within ϵ of the greatest lower bound of the observed errors.

Extending results of Pollard and others, Haussler proves results on the rate of convergence of empirical means of random variables to their expectations, uniformly over a class of (bounded) random variables. For $h \in H$, let the function $l_h : X \times Y \to [0, M]$ be given by $l_h(x, y) = l(h(x), y)$. Then the error of h with respect to distribution P on $X \times Y$ is simply the *expectation* $\mathbf{E}_P(l_h)$ of l_h, and the *observed error* of h on the sample $\mathbf{s}$ is the *empirical mean of* l_h based on $\mathbf{s}$,

$$\mathrm{er}_{\mathbf{s},l}(h) = \frac{1}{m} \sum_{i=1}^{m} l_h((x_i, y_i)).$$

Haussler shows that if the *loss space*, $l_H = \{l_h : h \in H\}$ has finite pseudo-dimension, then H has the UCE property (for loss function l).

Theorem 12.4 *Suppose that H is a set of functions from X to Y and that $l : Y \times Y \to [0, M]$ is a loss function. Let $l_H = \{l_h : h \in H\}$ be the loss space of H with respect to l. Suppose that the space l_H of functions from $X \times Y$ to $[0, M]$ has finite pseudo-dimension* $\mathrm{pdim}(l_H)$. *For $0 < \delta, \epsilon < 1$, let*

$$m_0(\delta, \epsilon) = \frac{64M^2}{\epsilon^2}\left(2\,\mathrm{pdim}(l_H) \ln\left(\frac{16eM}{\epsilon}\right) + \ln\left(\frac{8}{\delta}\right)\right).$$

Let P be any probability distribution on $S = X \times Y$. If $m \geq m_0(\delta, \epsilon)$, then with P^m-probability at least $1 - \delta$, $\mathbf{s} \in S^m$ satisfies $|\mathrm{er}_{\mathbf{s},l}(h) - \mathrm{er}_{P,l}(h)| < \epsilon$, for all $h \in H$.

As in our earlier discussion, this result shows that if the loss space has finite pseudo-dimension then H has the UCE property with respect to l. It follows, by an argument similar to that given in Theorem 9.4, that if $\hat{L}$ is, as in Theorem 12.3, a learning algorithm which outputs a hypothesis having near-minimal observed error, then $\hat{L}$ is a probably approximately optimal learning algorithm. A bound on the sample complexity

of this algorithm follows from the theorem. If H maps into $\mathbf{R}$, then the pseudo-dimension of l_H can be related to that of H itself if l has a certain 'monotonicity' property (see [55]). In this case, one obtains bounds on the sample length $m_0(\delta, \epsilon)$ for the UCE property, and the sample complexity of $\hat{L}$, which depend on $\mathrm{pdim}(H)$. Recently, Alon *et al.* [1] have determined a necessary and sufficient condition for H to have the UCE property (for loss function l). This condition is in many cases weaker than finite pseudo-dimension of l_H. Their result will be described later. For the moment, we present a result on the pseudo-dimension of a particular loss space relevant in the study of neural networks. The following result is a special case of one due to Maass [79]. It generalizes Theorem 7.11. (We state it only for the L^1-loss function and for less general types of activation function than those in the result of Maass.)

Theorem 12.5 *Let $\{\mathcal{N}_W\}$ be a family of feedforward neural networks with real outputs, W variable parameters, and piecewise polynomial functions of bounded degree and bounded number of pieces as activation functions on the hidden units. (The bounds on the number of pieces and degrees do not depend on W.) Let H_W be the set of functions computable by $\mathcal{N}_W$ and let l_W be the loss space of $\mathcal{N}_W$ relative to the L^1 loss function. Then $\mathrm{pdim}\,(l_W) = O(W^2)$ as $W \to \infty$.*

This theorem is proved by generalizing the techniques in [48]. The lower bound results of Theorem 7.8 and Theorem 7.9 show that a lower bound $\Omega(W \log W)$ holds in general. The upper bound of the above theorem is $O(W^2)$ and we know that for linear threshold networks the pseudo-dimension (which equals the VC-dimension) is $O(W \log W)$. It is unknown whether the power of analog neural networks can be of larger order than $W \log W$. This is an interesting open problem.

13 Capacity of a Function Space

13.1 Capacity and learning

We have seen that one way in which to ensure that a space of functions has the UCE property is to show that the pseudo-dimension of the corresponding loss space is finite. Another way, described in [55], is to use the notion of the *capacity* of a function space. For simplicity, we shall focus here only on the cases in which Y is a bounded subset of some $\mathbf{R}^k$, $Y \subseteq [0, M]^k$, and we shall use the L^1-loss function, which for the remainder of this section will be denoted simply by l. Other loss functions may be treated similarly. Observe that the loss function maps into $[0, M]$ in this case. Suppose that H is any set of functions mapping X into $[0, M]^k$ and that μ is a probability distribution on X. Extending the ideas developed earlier in the study of distribution-specific learning by Benedek and Itai, one can define a pseudo-metric ∂_μ on H by

$$\partial_\mu(f, g) = \mathbf{E}_\mu\left(l(f(x), g(x))\right).$$

The ϵ-*capacity* of H is defined to be

$$\mathcal{C}_H(\epsilon) = \sup_\mu N(\epsilon, H, \partial_\mu),$$

where the supremum is taken over all probability distributions μ on X. If there is no finite ϵ-cover for some μ, or if the supremum does not exist, we say that the ϵ-capacity is infinite.[12] Results of Haussler [55] and Pollard [95] provide the following uniform bound on the rate of convergence of observed errors to actual errors.

Theorem 13.1 *With the notation of this section, if P is any probability distribution on $S = X \times Y$, then*

$$P^m\left(\{\mathbf{s} \in S^m : \text{there is } h \in H \text{ with } |\mathrm{er}_{P,l}(h) - \mathrm{er}_{\mathbf{s},l}(h)| > \epsilon\}\right) < 4\,\mathcal{C}_H\left(\epsilon/16\right) e^{-\epsilon^2 m/64M^2}$$

for all $0 < \epsilon < 1$. Here, l denotes the L^1-loss function.

[12] The definition just given is not quite the same as the definition given by Haussler; here, we take a slightly more direct approach because we are not aiming for the full generality of Haussler's analysis.

When $k = 1$ and H maps into $[0, M]$, the capacity can be related to the pseudo-dimension of H. Haussler [55] (see also Pollard [95]) showed that if H has finite pseudo-dimension then

$$\mathcal{C}_H(\epsilon) < 2\left(\frac{2eM}{\epsilon}\ln\frac{2eM}{\epsilon}\right)^{\mathrm{pdim}(H)}.$$

This, combined with the above result, shows that, in this case, H has the UCE property and that a sufficient sample length $m_0(\delta, \epsilon)$ is

$$m_0(\delta, \epsilon) = \frac{64M^2}{\epsilon^2}\left(2\,\mathrm{pdim}(H)\log\left(\frac{16eM}{\epsilon}\right) + \log\left(\frac{8}{\delta}\right)\right).$$

(Note that, using the capacity approach, we deal directly with $\mathrm{pdim}(H)$ rather than with $\mathrm{pdim}(l_H)$.) Thus, if H is a space of real functions and $\mathrm{pdim}(H)$ is finite, then a learning algorithm $\hat{L}$ (as in Theorem 12.3) which outputs the hypothesis with near-minimal observed error is a probably approximately optimal learning algorithm (with respect to the L^1 loss function), having sample complexity $m_{\hat{L}}(\delta, \epsilon)$ bounded by $m_0(\delta, \epsilon/3)$.

13.2 Applications to sigmoid neural networks

We see from the above discussion that, for neural networks having one real output, the capacity approach yields a bound on the sample complexity of the probably approximately optimal learning algorithm $\hat{L}$. Furthermore, in such cases, the bound depends on the pseudo-dimension of the set of functions computed by the network, rather than on the pseudo-dimension of the associated loss space.

Macintyre and Sontag [84] proved the finiteness of the pseudo-dimension of feedforward neural networks having as activation the standard sigmoid function. However, no explicit bounds were given. Bartlett and Williamson [24] obtained the following result, relevant for discrete inputs.

Theorem 13.2 *Let $\mathcal{N}$ be a depth-two feedforward network having the standard sigmoid activation function on the hidden units and an output unit with linear activation function (so that it outputs the weighted sum of its inputs, without thresholding.) For a positive integer D, let $X_D = \{-D, -D+1, \ldots, D-1, D\}^n$, where n is the number of inputs to $\mathcal{N}$. Let H_D be the set of functions from X_D to $\mathbf{R}$ computable by the network on example set X_D. Then* $\mathrm{pdim}(H_D) < 8W\log(11WD)$.

We now describe an approach taken by Haussler, bounding directly the capacity of more general types of sigmoid network. In his paper, Haussler [55] shows how the general framework and results can, in addition, be applied to radial basis function networks and networks composed of product units. Suppose that each activation function f_i is a 'smooth' bounded monotone function, which need not be the standard sigmoid. In particular, suppose that f_i takes values in a bounded interval $[\alpha, \beta]$ and that it is differentiable on $\mathbf{R}$ with bounded derivative, $|f_i'(x)| \leq B$ for all x. By proving some 'composition' results on the capacity of function spaces and by making use of the pseudo-dimension and its relationship to capacity for real-valued function spaces, Haussler obtained bounds on the capacity of feedforward artificial neural networks with general sigmoid activation functions. We state the following special case of a result from [55].

Theorem 13.3 *Suppose that $\mathcal{N}$ is a feedforward sigmoid network of depth d, with any number of output nodes and W variable parameters (weights and thresholds). Let Δ be the maximum in-degree of a computation node. Suppose that each activation function maps into the interval $[\alpha, \beta]$. Let H be the set of functions computable by the network on inputs from $[\alpha, \beta]^n$ when the variable parameters are constrained to be at most V in absolute value. Then for $0 < \epsilon \leq \beta - \alpha$,*

$$\mathcal{C}_H(\epsilon) \leq \left(\frac{2e(\beta-\alpha)d(\Delta VB)^{d-1}}{\epsilon}\right)^{2W},$$

where B is a bound on the absolute values of the derivatives of the activation functions. Further, for fixed V, there is a constant K such that for any probability distribution P on $X \times \mathbf{R}^k$, the following holds: provided

$$m \geq \frac{K}{\epsilon^2}\left(W \log\left(\frac{B^d}{\epsilon}\right) + \log\left(\frac{1}{\delta}\right)\right),$$

then with P^m-probability at least $1-\delta$, a sample $\mathbf{s}$ from $(X \times \mathbf{R}^k)^m$ satisfies

$$|\mathrm{er}_{\mathbf{s},l}(h) - \mathrm{er}_{P,l}(h)| < \epsilon$$

for all $h \in H$. Here, l denotes the L^1-loss function.

This result shows that the space of functions computed by a certain type of sigmoid network has the UCE property. It provides an upper bound on the length of sample which should be used in order to be confident that the observed error is close to the actual error. It follows that any learning algorithm with the property described in Theorem 12.3, resulting in a state of the network with near-minimal observed error, is probably approximately optimal. The bound of the theorem provides a bound on the sample complexity of such an algorithm. The presence of the bound B on the absolute values of the derivatives of the activation functions means that this theorem does not apply to linear threshold networks, where the activation functions are not differentiable. Nonetheless, the sample length bounds are similar to those obtained for linear threshold networks. Although in this theorem there is assumed to be some uniform upper bound V on the maximum magnitude of the weights, the result of Macintyre and Sontag [84] mentioned earlier shows that if every activation function is the *standard* sigmoid function and if there is one output node, then such a bound is not necessary. They show that the set of functions computed by a standard sigmoid network with unrestricted weights (and on unrestricted real inputs) has finite pseudo-dimension.

14 SCALE-SENSITIVE DIMENSIONS

14.1 Learnability of p-concepts

An interesting variant of the PAC model which has received attention recently is that of 'p-concept learning', introduced by Kearns and Schapire [70]. Much attention has been on 'learning a good model of probability' of a p-concept and it is this problem we shall discuss here.

A *p-concept* (or *probabilistic concept*) is a function t from X to the interval $[0, 1]$. The value $t(x)$ is meant to represent a probability. A motivating example given by Kearns and Schapire is that in which the example space x encodes a set of meteorological measurements and $t(x)$ is the probability, given measurements x, that rain will follow. In such a 'learning' scenario, one of only two possible outcomes are observed; either it rains or it does not rain. The aim of a weather forecaster is, based on such observations, to determine a good approximation to the probability $t(x)$ that it will rain under the conditions encoded by x. Suppose that H is a class of p-concepts, and that $t \in H$. A training sample $\mathbf{s}$ for t in this context consists of a sequence of examples, *each labeled with* 0 *or* 1. The example x is labeled 1 with probability $t(x)$ and 0 with probability $1 - t(x)$. We shall call such a training sample a *p-concept training sample* to distinguish it from the usual notion of a training sample. A p-concept learning algorithm receives as input the accuracy parameter ϵ (as for function learning), the 'scale' parameter η, and a p-concept training sample $\mathbf{s}$. The aim of the learning algorithm is to produce a function $L(\eta, \epsilon, \mathbf{s}) \in H$ such that with high probability $L(\eta, \epsilon, \mathbf{s})$ is 'close to' t. Given η and ϵ between 0 and 1 and a probability distribution μ on X, we say that a p-concept h is *a (η, ϵ)-good model of probability* for the p-concept t if

$$\mu(\{x : |h(x) - t(x)| \geq \eta\}) < \epsilon.$$

The formal definition of p-concept learning with a good model of probability (henceforth, for brevity, simply 'learning') can now be given.

Definition 14.1 *The class H of p-concepts is* learnable by a p-concept learning algorithm L *if for all $\eta, \delta, \epsilon \in (0,1)$, there is a sample size $m_L(\eta, \delta, \epsilon)$ such that for any $t \in H$ and any probability distribution μ on X, if $m \geq m_L(\eta, \delta, \epsilon)$ then, given the scale parameter η, the accuracy parameter ϵ, and a p-concept training sample $\mathbf{s}$ for t of length m, the algorithm outputs a hypothesis $L(\eta, \epsilon, \mathbf{s})$ which, with probability at least $1-\delta$, is a (η, ϵ)-good model of probability of t.*

Note that any p-concept on X and any probability distribution μ on X can be realized by a probability distribution P on $X \times \{0,1\}$. (For X countable, for instance, P is given by $P((x,1)) = \mu(x)t(x)$ and $P((x,0)) = \mu(x)(1-t(x))$.) Thus it is possible to give a more general definition of p-concept learning, in which the aim is not to find a good model of probability but to find a hypothesis which almost minimizes the linear or quadratic loss with respect to a target distribution P on $X \times \{0,1\}$; see [1], for instance.

In their paper, Kearns and Schapire gave examples of efficient algorithms for learning particular classes of p-concepts. One example they gave is the class ND of all non-decreasing functions from $[0,1]$ to $[0,1]$. They show that if H has finite pseudo-dimension then H is learnable (by an algorithm which near-minimizes observed quadratic loss, as in Theorem 12.3). This result follows from the uniform convergence result, Theorem 12.4, and from the fact that the pseudo-dimension of the (appropriate restriction of the) loss space equals the pseudo-dimension of H. The fact that ND is learnable yet of infinite pseudo-dimension shows that finite pseudo-dimension is not necessary for p-concept learning. Kearns and Schapire also obtain a lower bound on the sample complexity of a p-concept learning algorithm, involving a 'scale-sensitive' version of the pseudo-dimension (to be described below). Alon *et al.* [1] proved a very general result which shows that the parameter involved in the lower bound of Kearns and Schapire is the crucial one for the analysis of p-concept learnability. This parameter, the γ-dimension, is formally defined as follows.

Definition 14.2 *For $\gamma > 0$, we say that a sample $(x_1, \ldots, x_d)$ is γ-shattered by H if the following holds: there are $r_1, r_2, \ldots, r_d$ in $[0,1]$ such that for each $\mathbf{b} \in \{0,1\}^d$ there is $h_\mathbf{b} \in H$ with*

$$h_\mathbf{b}(x_i) > r_i + \gamma \text{ if } b_i = 1, \quad h_\mathbf{b}(x_i) < r_i - \gamma \text{ if } b_i = 0.$$

The γ-dimension of H, denoted $\dim_\gamma(H)$ is (infinite, or) the maximal length of γ-shattered sample.

In other words, $\mathbf{x}$ is γ-shattered if it is shattered in the sense of pseudo-dimension, but with a 'width of shattering' of at least γ. This dimension is an example of a 'scale-sensitive dimension'. As an example, it is fairly easy to see that $\dim_\gamma(\mathrm{ND}) = \lfloor 1/2\gamma \rfloor$. Note that the pseudo-dimension is the limit of the γ-dimension: $\mathrm{pdim}(H) = \lim_{\gamma \to 0} \dim_\gamma(H)$. It is possible, as in the case of ND, for $\dim_\gamma(H)$ to be finite for all γ but for $\mathrm{pdim}(H)$ to be infinite. This scale-sensitive dimension has been bounded for classes of neural networks by Gurvits and Koiran [54] and Bartlett [23].

Alon *et al.* [1] proved the following result. (The necessity was proved earlier by Kearns and Schapire [70].)

Theorem 14.3 *Let H be a class of p-concepts. Then H is learnable (in the p-concept model) if and only if $\dim_\gamma(H)$ is finite for all $\gamma > 0$.*

The algorithm used to prove sufficiency in the above theorem is, like that of Theorem 12.3, one which outputs a hypothesis which has near-minimal observed error, with respect to the quadratic loss. (As indicated above, they use a slightly more general model of p-concept learning in which it is not assumed that the target p-concept belongs to H.) Given the results in their paper, it follows that the sample complexity, $m_L(\eta, \delta, \epsilon)$, of this algorithm is of order

$$\frac{1}{\epsilon^2 \eta^4} \left(d \left(\ln \left(\frac{d}{\epsilon \eta^2} \right) \right)^2 + \ln \left(\frac{1}{\delta} \right) \right),$$

where $d = \dim_{\epsilon\eta^2/96}(H)$. (The bound given in their paper is for the problem of finding a hypothesis with near-minimal quadratic error. The bound given here follows by an argument analogous to that given in the proof of Theorem 14.7 below.)

Note that the condition in this theorem—finite γ-dimension for all γ—is a weaker condition than that of having finite pseudo-dimension. In related work, Simon [109] has obtained general lower bounds on the sample complexity of p-concept learning algorithms in terms of a different scale-sensitive dimension.

14.2 Learnability of functions

In this subsection, we return to the model of learning real functions on X, emphasising the importance of scale-sensitive dimension, and briefly discussing learning in the presence of noise. Suppose we have a fixed loss function l. In the context of function learning with respect to a probability distribution P on $X \times \mathbf{R}$, a training sample, as in earlier sections, is an element of $(X \times \mathbf{R})^m$ for some m. Recall that a learning algorithm L is *probably approximately optimal* if for all $0 < \delta, \epsilon < 1$, there is $m_L(\delta, \epsilon)$ such that for $m \geq m_L$, and for any probability distribution P on $X \times \mathbf{R}$, with probability at least $1 - \delta$, a sample $\mathbf{s} \in (X \times \mathbf{R})^m$ is such that

$$\mathrm{er}_{P,l}\left(L(\epsilon, \mathbf{s})\right) < \inf_{h \in H} \mathrm{er}_{P,l}(h) + \epsilon.$$

Here, $\mathrm{er}_{P,l}(h)$ is the expectation $\mathbf{E}_P(l(h(x), y))$.

Alon *et al.* [1] prove some very general results. A corollary of their main result is the following characterisation of spaces having the UCE property with respect to a loss function l.

Theorem 14.4 *Let H be a set of functions from X to $\mathbf{R}$. Then H has the UCE property (with respect to loss function l) if and only if $\dim_\gamma(l_H)$ is finite for all $\gamma > 0$.*

We have the following corollary.

Corollary 14.5 *Let H be a set of functions from X to $[0, M]$, for some M, and let l be a loss function. Suppose that the loss space l_H has finite γ-dimension for all γ. Then the learning algorithm $\hat{L}$ of Theorem 12.3, which outputs a hypothesis having near-minimal observed error, is a probably approximately optimal learning algorithm.*

Now suppose that we have a concept space C of real functions and that $C \subseteq H$. Let us suppose also that the functions in H are uniformly bounded: without loss of generality, H maps into some interval of the form $[0, M]$. Suppose $t \in C$ and that μ is a probability distribution on X. Such a pair (t, μ) can, as usual, be represented by a probability distribution P on $X \times \mathbf{R}$. We now restrict attention to those distributions P on $X \times \mathbf{R}$ that correspond to an underlying distribution μ on X and a target function $t \in C$ in this way, and we take as loss function either the quadratic loss or the linear loss. In this situation, $\mathrm{opt}_H(P) = 0$ and thus a probably approximately optimal algorithm (restricted to such P) is a *probably approximately correct* (C, H)-learning algorithm. We have an alternative characterisation of such PAC algorithms, as follows.

Definition 14.6 *Let C and H be sets of functions from X to $[0, M]$ with $C \subseteq H$. For a positive integer m and $t \in C$, let $S(m, t)$ denote the set of all training samples of length m for t. A function*

$$\mathcal{A} : \mathbf{R}^+ \times (0, 1) \times \bigcup_{t \in C, m \geq 1} S(m, t) \to H$$

is said to learn C with a good model by H *if for every $0 < \delta, \epsilon, \eta < 1$ there is $m_{\mathcal{A}}(\eta, \delta, \epsilon)$ such that if $m \geq m_{\mathcal{A}}$, the following holds: if μ is any distribution on X, and $t \in C$ then, with probability at least $1 - \delta$, the function $h_{\mathcal{A}} = \mathcal{A}(\eta, \epsilon, \mathbf{s})$ satisfies*

$$\mu^m\left(\{x \in X : |h_{\mathcal{A}}(\mathbf{s})(x) - t(x)| \geq \eta\}\right) < \epsilon.$$

Theorem 14.7 *Let C and H be sets of functions from X to $[0, M]$ with $C \subseteq H$. Then there is a PAC (C, H)-learning algorithm (with respect to either the quadratic or linear loss function) if and only if there is an algorithm $\mathcal{A}$ which learns C with a good model (by H).*

Proof We prove equivalence for the quadratic loss. Suppose that L is a PAC algorithm with respect to the quadratic loss. Algorithm $\mathcal{A}$ acts as follows: given as input $\eta, \epsilon \in (0,1)$ and $\mathbf{s} \in S(m,t)$, $\mathcal{A}$ simulates L, giving output $L(\epsilon\eta^2, \mathbf{s})$. Let $m_L(\delta, \epsilon)$ be the sample complexity function of L. We claim that $\mathcal{A}$ learns with a good model and has sample complexity satisfying

$$m_{\mathcal{A}}(\eta, \delta, \epsilon) \leq m_L(\delta, \epsilon\eta^2).$$

To see this, simply observe that if a sample $\mathbf{s}$ has length at least $m_L(\delta, \epsilon\eta^2)$, then with probability at least $1-\delta$, $h_L = L(\epsilon\eta^2, \mathbf{s})$ satisfies

$$\mathrm{er}_{P,l}(h_L) = \mathbf{E}_\mu\left((h_L(x) - t(x))^2\right) < \epsilon\eta^2,$$

where P is the probability distribution on $X \times \mathbf{R}$ arising from μ on X and $t \in C$. But it then follows that

$$\mu\left(\{x \in X : |h_L(x) - t(x)| \geq \eta\}\right) < \epsilon,$$

by Markov's inequality. Thus such an algorithm $\mathcal{A}$ exists. Conversely, suppose that $\mathcal{A}$ learns with a good model. Then there is a PAC algorithm: this algorithm simulates $\mathcal{A}$, and has sample complexity m_L satisfying $m_L(\delta, \epsilon) \leq m_{\mathcal{A}}(\sqrt{\epsilon}/2, \delta, \epsilon/2M^2)$. This follows from the observation that if $\mu\left(\{x : |h(x) - t(x)| \geq \sqrt{\epsilon}/2\}\right) < \epsilon/2$, then

$$\mathbf{E}_\mu\left((h(x) - t(x))^2\right) < (\sqrt{\epsilon}/2)^2 + (\epsilon/2M^2)M^2 < \epsilon,$$

where we have used the fact that functions in H map into $[0, M]$. The result follows. □

Definition 14.6 is very similar to the definition of p-concept learning. If H is a set of functions from X to $[0,1]$, it is clear that the p-concept learning of H (by H) is more difficult than the learning of H (by H) with a good model: in learning with a good model, if x is an example in the training sample, then the learning algorithm receives the value $t(x)$, whereas, in a p-concept training sample, x carries a binary label stochastically determined by $t(x)$. It follows that we cannot infer from Theorem 14.3 that finite γ-dimension for all γ is necessary for function learning with a good model. (Indeed, it is not.) However, any upper bound on the sample complexity of p-concept learning a class H provides an upper bound for learning H with a good model.

In [9, 20, 18], the following result is given. This shows that if the hypothesis space has finite pseudo-dimension and a learning algorithm interpolates well enough on the training sample then it is a probably approximately correct algorithm. Further, it shows that if there is some noise in the classification of the training sample, then learning 'up to the level of the noise' is possible in some cases. (See [91, 26] for other results on function learning in the presence of noise.)

Theorem 14.8 *Let H be a set of functions from X to $[0, M]$, for some M. Suppose H has finite pseudo-dimension and that $C \subseteq H$. Let $0 < \eta < 1$ and suppose that L is a (C, H)-learning algorithm which receives as input parameters η, ϵ and samples $\mathbf{s}$. Suppose also that, for $t \in C$, for*

$$\mathbf{s} = ((x_1, t(x_1)), (x_2, t(x_2)), \ldots, (x_m, t(x_m))) \in S(m,t),$$

the hypothesis $h_L = L(\eta, \epsilon, \mathbf{s})$ has the property that $|h_L(x_i) - t(x_i)| < \eta$ for $1 \leq i \leq m$. Then, for $0 < \delta < 1$, there is $m_L(\eta, \delta, \epsilon)$, of order

$$\frac{1}{\epsilon}\left(\mathrm{pdim}(H)\ \ln\left(\frac{1}{\epsilon}\right) + \ln\left(\frac{1}{\delta}\right)\right),$$

such that if $m \geq m_L$, the following holds: for any probability distribution μ on X and any $t \in C$, with probability at least $1-\delta$, $\mu\left(\{x \in X : |h_L(x) - t(x)| \geq \eta\}\right) < \epsilon$.

The result presented in [9] is more general than this. In that paper, it is shown that to obtain accurate bounds on the sample complexity function m_L (and, indeed, bounds depending on η), it is more appropriate to use a scale-sensitive dimension termed the *band-dimension*, also used in [91]. In [9], it is also shown that finite pseudo-dimension of H is a *necessary* condition for the conclusion of Theorem 14.8 to hold. Thus the condition described in this theorem is *stronger* than learnability. However, Anthony and Bartlett [7] have shown that finite γ-dimension for all γ *is* sufficient for a weaker conclusion, as the following theorem shows.

Theorem 14.9 *Let H be a set of functions from X to $[0, M]$, for some M. Suppose H has finite γ-dimension for all γ and that $C \subseteq H$. Let $0 < \eta < 1$ and suppose that L is a (C, H)-learning algorithm which receives as input parameters η, ϵ and samples $\mathbf{s}$. Suppose also that, for $t \in C$, for*

$$\mathbf{s} = ((x_1, t(x_1)), (x_2, t(x_2)), \ldots, (x_m, t(x_m))) \in S(m, t),$$

the hypothesis $h_L = L(\eta, \epsilon, \mathbf{s})$ has the property that $|h_L(x_i) - t(x_i)| < \eta$ for $1 \leq i \leq m$. Then, for any $0 < \alpha, \delta < 1$, there is $m_L(\eta, \alpha, \delta, \epsilon)$, of order

$$\frac{1}{\epsilon}\left(\ln\left(\frac{1}{\delta}\right) + \dim_{\alpha/8}(H) \ln^2\left(\frac{\dim_{\alpha/8}(H)}{\alpha\epsilon}\right)\right).$$

such that if $m \geq m_L$, the following holds: for any probability distribution μ on X and any $t \in C$, with probability at least $1 - \delta$, $\mu(\{x \in X : |h_L(x) - t(x)| \geq \eta + \alpha\}) < \epsilon$.

Bartlett, Long and Williamson [26] have recently shown that even if there is some reasonable level of noise in the labeling of the training sample, then, provided $\dim_\gamma(H)$ is finite for all γ, learning with a good model is possible. It is not possible to obtain non-trivial general lower bounds on the sample complexity of (noiseless) function learning in terms of $\dim_\gamma(H)$. A simple 'coding' argument shows this. (See [26].) However, Bartlett *et al.* have obtained lower bounds on the sample complexity in the presence of noise. Their results show that if we demand learnability which is robust in the presence of noise, the finiteness of the γ-dimension of H for all γ is a necessary and sufficient condition for learnability. Using a different scale-sensitive dimension—which may be thought of as a scale-sensitive version of the Natarajan dimension—Simon [110] has obtained general lower bounds on the sample complexity of noiseless learning. He has shown that, to within a logarithmic factor, this bound is optimal for a number of spaces H. Anthony and Bartlett [7] have shown that finite γ-dimension for all γ *is necessary* as well as sufficient for the conclusion of Theorem 14.9 to hold.

15 Conclusions and further reading

There are many aspects of learning theory not discussed in this work. We have barely touched on the computational complexity of PAC learning and its variants. Further discussion of this may be found, for example, in [82, 21, 10, 73, 93, 69, 70, 101, 63, 79]. There are also many more variants of the PAC model which we have not mentioned here. Throughout the discussion here, it has been assumed that there is some fixed hypothesis space. This is natural in the context of neural networks where one has in mind a fixed architecture. However, there is a variant of the PAC model known as the *prediction* model, in which there is no fixed hypothesis space; in a sense, the output of a prediction algorithm is some program which classifies further examples. We refer the reader to [94, 57] for details. Another very important variant is that in which the learning algorithm can ask questions concerning, for example, the classification of a chosen example; see, for example [2, 28, 53]. Such 'query learning' is an active area of research, and the paper by Angluin [3] provides a good survey. In a similar vein, one may ask how much easier learning becomes when there is a 'helpful teacher' providing cleverly-chosen examples as training sample. (This is very different from the PAC model in that the training examples are no longer randomly chosen.) Such models of teaching have been studied in the case where the goal is to learn the target exactly [49, 13, 65, 108, 14] and in which the goal is to learn an approximation to the target in a probabilistic sense [97, 96]. Other useful variants not discussed here are those in which the distribution and target are permitted to change a little between observations, as in [25, 58, 59], models of *weak learning* in which the learner only has to do slightly better than random guessing [51, 100, 60], and variants in which the learning algorithm has access to the predictions of 'experts' [38]. Something which has not been discussed in any detail here is the use of real-output neural networks for classification. Recent work [106, 23] has shown that, here, the scale-sensitive γ-dimension is very useful.

Acknowledgements

I thank Peter Bartlett, Nicolò Cesa-Bianchi and Mark Jerrum for comments and suggestions on an earlier version of this article. I am grateful to three anonymous referees for their detailed and helpful comments on the journal version.

References

[1] N. Alon, S. Ben-David, N. Cesa-Bianchi, and D. Haussler. Scale-sensitive dimensions, uniform convergence, and learnability. In *Proceedings of the Symposium on Foundations of Computer Science*. IEEE Press, 1993. To appear, *Journal of the ACM*.

[2] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, Apr. 1988.

[3] D. Angluin. Computational learning theory: survey and selected bibliography. In *Proc. 24th Annu. ACM Sympos. Theory Comput.*, pages 351–369. ACM Press, New York, NY, 1992.

[4] D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[5] M. Anthony. *Uniform Convergence and Learnability.* PhD thesis, University of London (London School of Economics and Political Science), Feb. 1991. A revised version appears as London School of Economics Mathematics Preprint LSE-MPS-11, May 1991.

[6] M. Anthony. Classification by polynomial surfaces. Mathematics Preprint Series LSE-MPS-39, London School of Economics, Oct. 1992. Revised version appears *Discrete Applied Mathematics*, 61 (1995): 91–103.

[7] M. Anthony and P. Bartlett. Function learning from interpolation. Extended abstract in Proceedings EuroCOLT'95, Springer-Verlag, 1995: 211–221. Full version submitted.

[8] M. Anthony and P. Bartlett. *Theory of Learning in Neural Networks* (working title). In preparation. To be published by Cambridge University Press.

[9] M. Anthony, P. Bartlett, Y. Ishai, and J. Shawe-Taylor. Valid generalisation from approximate interpolation. *Combinatorics, Probability and Computing*, Volume 5 191–214 (1996).

[10] M. Anthony and N. Biggs. *Computational Learning Theory: An Introduction.* Cambridge Tracts in Theoretical Computer Science (30). Cambridge University Press, Cambridge, UK, 1992. (Reprinted, in paperback, with corrections, 1997.)

[11] M. Anthony and N. Biggs. Computational learning theory for artificial neural networks. In J. Taylor, editor, *Mathematical Approaches to Neural Networks*, North Holland Mathematical Library (51), pages 25–62. Elsevier Science Publishers B. V., Amsterdam, 1993.

[12] M. Anthony, N. Biggs, and J. Shawe-Taylor. The learnability of formal concepts. In *Proc. 3rd Annu. Workshop on Comput. Learning Theory*, pages 246–257. Morgan Kaufmann, San Mateo, CA, 1990.

[13] M. Anthony, G. Brightwell, D. Cohen, and J. Shawe-Taylor. On exact specification by examples. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 311–318. ACM Press, New York, NY, 1992.

[14] M. Anthony, G. Brightwell, and J. Shawe-Taylor. On specifying Boolean functions by labelled examples. *Discrete Applied Mathematics*, 61 (1995): 1–25.

[15] M. Anthony and S. B. Holden. Quantifying generalisation in linearly weighted neural networks. *Complex Systems* 8, (1994), 91–114.

[16] M. Anthony and S. B. Holden. On the power of polynomial discriminators and radial basis function networks. In *Proc. 6th Annu. Workshop on Comput. Learning Theory*, pages 158–164. ACM Press, New York, NY, 1993.

[17] M. Anthony and J. Shawe-Taylor. A sufficient condition for polynomial distribution-dependent learnability. To appear, *Discrete Applied Mathematics*..

[18] M. Anthony and J. Shawe-Taylor. Generalising from approximate interpolation. Mathematics Preprint Series LSE-MPS-47, London School of Economics, May 1993.

[19] M. Anthony and J. Shawe-Taylor. A result of Vapnik with applications. *Discrete Applied Mathematics*, 47:207–217, 1994. Also, technical report CSD-TR-628, Royal Holloway and Bedford New College, University of London, 1990.

[20] M. Anthony and J. Shawe-Taylor. Valid generalisation of functions from close approximations on a sample. In *Computational Learning Theory: Euro-COLT'93*, (ed. J. Shawe-Taylor and M. Anthony), Oxford University Press, 1994.

[21] P. Auer, P. Long, W. Maass, and G. Woeginger. On the complexity of function learning. In *Proc. 6th Annu. Workshop on Comput. Learning Theory*, pages 392–401. ACM Press, New York, NY, 1993.

[22] P. Bartlett. Vapnik-Chervonenkis dimension bounds for two- and three-layer networks. *Neural Computation* 5 (3): 371–373, 1993.

[23] P. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. Technical report, Department of Systems Engineering, Australian National University, May 1996.

[24] P. Bartlett and R. C. Williamson. The Vapnik-Chervonenkis dimension and pseudodimension of two-layer neural networks with discrete inputs. *Neural Computation* 8: 653–656, 1996.

[25] P. L. Bartlett. Learning with a slowly changing distribution. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 243–252. ACM Press, New York, NY, 1992.

[26] P. L. Bartlett, P. M. Long, and R. C. Williamson. Fat-shattering and the learnability of real-valued functions. *Journal of Computer and System Sciences*, 52(3): 434–452, 1996. Extended abstract in *Proceedings of COLT'94*.

[27] E. Baum and D. Haussler. What size net gives valid generalization? *Neural Computation*, 1(1):151–160, 1989.

[28] E. B. Baum. Polynomial time algorithms for learning neural nets. In *Proc. 3rd Annu. Workshop on Comput. Learning Theory*, pages 258–272. Morgan Kaufmann, San Mateo, CA, 1990.

[29] S. Ben-David, G. M. Benedek, and Y. Mansour. A parametrization scheme for classifying models of learnability. In *Proc. 2nd Annu. Workshop on Comput. Learning Theory*, pages 285–302. Morgan Kaufmann, San Mateo, CA, 1989. Journal version appears as A parameterization scheme for classifying models of PAC learnability, *Information and Computation* 120 (1): 11–21, 1995.

[30] S. Ben-David, N. Cesa-Bianchi, D. Haussler, and P. M. Long. Characterizations of learnability for classes of $\{0, \ldots, n\}$-valued functions. *J. of Comp. and Sys. Sci.* 50(1): 74–86, 1995.

[31] S. Ben-David, N. Cesa-Bianchi, and P. M. Long. Characterizations of learnability for classes of $\{0, ..., n\}$-valued functions. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 333–340. ACM Press, New York, NY, 1992.

[32] G. Benedek and A. Itai. Learnability with respect to fixed distributions. *Theoret. Comput. Sci.*, 86(2):377–389, 1991.

[33] G. M. Benedek and A. Itai. Learnability by fixed distributions. In *Proc. 1st Annu. Workshop on Comput. Learning Theory*, pages 80–90. Morgan Kaufmann, San Mateo, CA, 1988.

[34] G. M. Benedek and A. Itai. Dominating distributions and learnability. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 253–264. ACM Press, New York, NY, 1992.

[35] A. Bertoni, P. Campadelli, A. Morpurgo, and S. Panizza. Polynomial uniform convergence and polynomial-sample learnability. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 265–271. ACM Press, New York, NY, 1992.

[36] A. Blum and R. L. Rivest. Training a 3-node neural net is NP-Complete. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems I*, pages 494–501. Morgan Kaufmann, 1989.

[37] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989.

[38] N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, D. Haussler, R. E. Schapire, and M. K. Warmuth. How to use expert advice. In *Proc. 25th Annu. ACM Sympos. Theory Comput.*, pages 382–391. ACM Press, New York, NY, 1993.

[39] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA., 1990.

[40] L. Devroye, L. Gyorfi and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New York, 1996.

[41] R. Dudley, S. Kulkarni, T. Richardson, and O. Zeitouni. A metric entropy bound is not sufficient for learnability. *IEEE Transactions on Information Theory* 40(3): 883–885, 1994.

[42] R. M. Dudley. Central limit theorems for empirical measures. *Annals of Probability*, 6(6):899–929, 1978.

[43] A. Ehrenfeucht, D. Haussler, M. Kearns, and L. Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82:247–261, 1989.

[44] U. Faigle and W. Kern. On learnability of monotone DNF functions under uniform distribution. Manuscript, Department of Mathematics, University of Twente, Netherlands, 1990.

[45] M. Flammini, A. Marchetti-Spaccamela, and L. K. Cera. Learning DNF formulae under classes of probability distributions. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 85–92. ACM Press, New York, NY, 1992.

[46] M. L. Furst, J. C. Jackson, and S. W. Smith. Improved learning of AC^0 functions. In *Proc. 4th Annu. Workshop on Comput. Learning Theory*, pages 317–325. Morgan Kaufmann, San Mateo, CA, 1991.

[47] M. Garey and D. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. Freemans, San Francisco, 1979.

[48] P. Goldberg and M. Jerrum. Bounding the Vapnik-Chervonenkis dimension of concept classes parameterized by real numbers. *Machine Learning*, 18(2-3): 131–148, 1995. (Extended abstract appeared in *Proceedings of 6th Annual ACM Conference on Computational Learning Theory*, pages 361–369. ACM Press, 1993.)

[49] S. A. Goldman and M. J. Kearns. On the complexity of teaching. In *Proc. 4th Annu. Workshop on Comput. Learning Theory*, pages 303–314. Morgan Kaufmann, San Mateo, CA, 1991.

[50] S. A. Goldman, M. J. Kearns, and R. E. Schapire. Exact identification of circuits using fixed points of amplification functions. In *Proc. of the 31st Symposium on the Foundations of Comp. Sci.*, pages 193–202. IEEE Computer Society Press, Los Alamitos, CA, 1990.

[51] S. A. Goldman, M. J. Kearns, and R. E. Schapire. On the sample complexity of weak learning. In *Proc. 3rd Annu. Workshop on Comput. Learning Theory*, pages 217–231. Morgan Kaufmann, San Mateo, CA, 1990.

[52] M. Golea, M. Marchand and T. Hancock. On learning μ-perceptron networks on the uniform distribution. *Neural Networks* 9: 67–82, 1996.

[53] T. Hancock, M. Golea, and M. Marchand. Learning nonoverlapping perceptron networks from examples and membership queries. *Machine Learning* 16(3): 161–183, 1994.

[54] L. Gurvits, L. and P. Koiran. Approximation and learning of convex superpositions. In *Proceedings of Eurocolt'95*, Springer-Verlag Lecture Notes in Artificial Intelligence, pages 222–236.

[55] D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Inform. Comput.*, 100(1):78–150, Sept. 1992.

[56] D. Haussler, M. Kearns, N. Littlestone, and M. K. Warmuth. Equivalence of models for polynomial learnability. *Inform. Comput.*, 95(2):129–161, December 1991.

[57] D. Haussler, N. Littlestone, and M. K. Warmuth. Predicting {0,1} functions on randomly drawn points. *Information and Computation* 108(2): 212–261, 1994. (Extended abstract in *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 100–109. IEEE Computer Society Press, 1988.)

[58] D. P. Helmbold and P. M. Long. Tracking drifting concepts by minimizing disagreements. *Machine Learning* 14 (1): 27–45, 1994.

[59] D. P. Helmbold and P. M. Long. Tracking drifting concepts using random examples. In *Proc. 4th Annu. Workshop on Comput. Learning Theory*, pages 13–23. Morgan Kaufmann, San Mateo, CA, 1991.

[60] D. P. Helmbold and M. K. Warmuth. Some weak learning results. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 399–412. ACM Press, New York, NY, 1992.

[61] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, 1991.

[62] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, Mar. 1963.

[63] K. Höffgen and H. Simon. Robust trainability of single neurons. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 428–439. ACM Press, New York, NY, 1992.

[64] S. B. Holden. *On the theory of generalization and self-structuring in linearly weighted connectionist networks.* PhD thesis, University of Cambridge, Sept. 1993. Also appears as Cambridge University Engineering Department technical report CUED/F-INFENG/TR.161, January 1994.

[65] J. Jackson and A. Tomkins. A computational model of teaching. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 319–326. ACM Press, New York, NY, 1992.

[66] S. Judd. Learning in neural networks. In *Proc. 1st Annu. Workshop on Comput. Learning Theory*, pages 2–8. Morgan Kaufmann, San Mateo, CA, 1988.

[67] M. Karpinski and A. Macintyre. Polynomial bounds for VC Dimension of Sigmoidal Neural Networks. In *Proceedings of the 27th annual ACM Symposium on the Theory of Computing*, 200–208, 1995.

[68] M. Kearns, M. Li, L. Pitt, and L. Valiant. On the learnability of Boolean formulae. In *Proc. 19th Annu. ACM Sympos. Theory Comput.*, pages 285–294. ACM Press, New York, NY, 1987.

[69] M. J. Kearns. *The Computational Complexity of Machine Learning*. ACM Distinguished Dissertation Series. The MIT Press, Cambridge, MA., 1989.

[70] M. J. Kearns and R. E. Schapire. Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences* 48: 464–497, 1994. (Extended abstract in *Proc. of the 31st Symposium on the Foundations of Comp. Sci.*, pages 382–391. IEEE Computer Society Press, Los Alamitos, CA, 1990.)

[71] M. J. Kearns, R. E. Schapire, and L. M. Sellie. Toward efficient agnostic learning. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 341–352. ACM Press, New York, NY, 1992.

[72] M.J. Kearns and L.G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM* 41(1): 67–95, 1994.

[73] M.J. Kearns and U. Vazirani (1995). *Introduction to Computational Learning Theory*, MIT Press 1995.

[74] M. Kharitonov. Cryptographic hardness of distribution-specific learning. In *Proc. 25th Annu. ACM Sympos. Theory Comput.*, pages 372–381. ACM Press, New York, NY, 1993.

[75] P. Koiran and E.D. Sontag, Neural Networks with Quadratic VC Dimension. *Journal of Computer and System Sciences* 54(1): 190–198, 1997

[76] W. S. Lee, P. L. Bartlett, and R. C. Williamson. Lower bounds on the VC-dimension of smoothly parametrized function classes. *Neural Computation* 7: 990–1002, 1995.

[77] M. Li and P. M. B. Vitanyi. Learning simple concepts under simple distributions. *SIAM J. Computing* 20(5): 911–935, 1991.

[78] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform, and learnability. In *Proc. of the 31st Symposium on the Foundations of Comp. Sci.*, pages 574–579. IEEE Computer Society Press, 1989.

[79] W. Maass. Agnostic PAC-learning of functions on analog neural nets (extended abstract) In *Advances in Neural Information Processing Systems, 6*. Morgan Kaufmann, 1993. Full version: *Neural Computation* 7(5): 1054–1078, 1995.

[80] W. Maass. Bounds on the computational power and learning complexity of analog neural nets (extended abstract). In *Proceedings of 25th Annual ACM Symposium on the Theory of Computing*, pages 335–344. ACM Press, 1993.

[81] W. Maass. Neural nets with superlinear VC-dimension. *Neural Computation*, 6(5): 877–884, 1994.

[82] W. Maass. On the complexity of learning on feedforward neural nets. Manuscript, Institute for Theoretical Computer Science, Technische Universitaet Graz., 1993.

[83] W. Maass. Vapnik-Chervonenkis dimension of neural nets. In *The Handbook of Brain Theory and Neural Networks* (ed. M.A. Arbib), Bradford Books/MIT Press, 1995, pp. 1000–1003.

[84] A. Macintyre and E. D. Sontag. Finiteness results for sigmoidal "neural" networks. In *Proceedings of 25th Annual ACM Symposium on the Theory of Computing*, pages 325–334. ACM Press, 1993.

[85] C. McDiarmid. On the method of bounded differences. In J. Siemons, editor, *Surveys in Combinatorics, 1989*, London Mathematical Society Lecture Note Series (141). Cambridge University Press, Cambridge, UK, 1989.

[86] M. Marchand and S. Hadjifaradji. Strong unimodality and exact learning of constant depth μ-perceptron networks. in D.S. Touretzky, M.C. Moxer and M.E. Hasselmo, eds., *Advances in Neural Information Processing Systems 8*, 288–294, MIT Press, Cambridge MA, 1996.

[87] C. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986.

[88] B. Natarajan. Probably approximate learning over classes of distributions. *SIAM J. Computing*, 21(3):438–449, 1992.

[89] B. K. Natarajan. On learning sets and functions. *Machine Learning*, 4(1), 1989.

[90] B. K. Natarajan. *Machine Learning: A Theoretical Approach.* Morgan Kaufmann, San Mateo, California, 1991.

[91] B. K. Natarajan. Occam's razor for functions. In *Proc. 6th Annu. Workshop on Comput. Learning Theory*, pages 370–376. ACM Press, New York, NY, 1993.

[92] G. Pagallo and D. Haussler. A greedy method for learning μ-DNF functions under the uniforn distribution. Technical report, University of California at Santa Cruz, UCSC-CRL-89-12, 1989.

[93] L. Pitt and L. Valiant. Computational limitations on learning from examples. *J. ACM*, 35:965–984, 1988.

[94] L. Pitt and M. K. Warmuth. Prediction preserving reducibility. *J. of Comput. Syst. Sci.*, 41(3):430–467, December 1990. Special issue on the *Third Annual Conference of Structure in Complexity Theory* (Washington, DC., June 88).

[95] D. Pollard. *Convergence of Stochastic Processes.* Springer-Verlag, 1984.

[96] K. Romanik. Testing as a dual to learning. Technical Report UMIACS-TR-91.93, CS-TR-2704, Dept. of Computer Science, University of Maryland, June 1991.

[97] K. Romanik. Approximate testing and learnability. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 327–332. ACM Press, New York, NY, 1992.

[98] A. Sakurai. Tighter bounds of the VC-dimension of three-layer networks. In *Proceedings of World Congress on Neural Networks*, pages 540–543, 1993.

[99] N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory (A)*, 13:145–147, 1972.

[100] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[101] R. E. Schapire. *The Design and Analysis of Efficient Learning Algorithms.* ACM Distinguished Dissertation Awards Series. The MIT Press, Cambridge, MA., 1991.

[102] J. Shawe-Taylor. Building symmetries into feedforward network architectures. In *Proceedings of First IEE Conference on Artificial Neural Networks*, pages 158–162, 1989.

[103] J. Shawe-Taylor. Sample sizes for threshold networks with equivalences. *Information and Computation*, 118(1): 65–72, 1995.

[104] J. Shawe-Taylor and M. Anthony. Sample sizes for multiple output threshold networks. *Network: Computation in Neural Systems*, 2:107–117, 1991.

[105] J. Shawe-Taylor, M. Anthony, and N. Biggs. Bounding sample-size with the Vapnik-Chervonenkis dimension. *Discrete Applied Mathematics*, 42:65–73, 1993.

[106] J. Shawe-Taylor, P. Bartlett, R. Williamson, M. Anthony. Structural risk minimisation over data-dependent hierarchies. submitted.

[107] S. Shelah. A combinatorial problem: Stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41:247–261, 1972.

[108] A. Shinohara and S. Miyano. Teachability in computational learning. *New Generation Computing*, 8:337–347, 1991.

[109] H. U. Simon. General bounds on the number of examples needed for learning probabilistic concepts. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 402–411. ACM Press, New York, NY, 1993. Full version: *J. of Comp. and Sys. Sci.* 52(2): 239–254, 1996.

[110] H. U. Simon. Bounds on the number of examples needed for learning functions. In *Computational Learning Theory: Eurocolt'93*. Oxford University Press, 1994.

[111] E. Sontag. Feedforward nets for interpolation and classification. *J. Comp. Syst. Sci*, 45:20–48, 1992.

[112] L.G. Valiant. Deductive learning. *Phil. Trans. Roy. Soc. Lond. A*, 312:441–446, 1984.

[113] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, Nov. 1984.

[114] V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York, 1982.

[115] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[116] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probab. and its Applications*, 16(2):264–280, 1971.

[117] K. Verbeurgt. Learning DNF under the uniform distribution in quasi-polynomial time. In *Proc. 3rd Annu. Workshop on Comput. Learning Theory*, pages 314–326. Morgan Kaufmann, San Mateo, CA, 1990.

[118] D. Wolpert (editor). *The Mathematics of Generalization: the Proceedings of the SFI/CNLS Workshop on Formal Approaches to Supervised Learning, Santa Fe, 1992*. Addison-Wesley, Reading, MA, 1995.

[119] K. Yamanishi. A learning criterion for stochastic rules. *Machine Learning* 9: 165–203, 1992.