Network 2 (1991) 107–117. Printed in the UK

Sample sizes for multiple-output threshold networks

John Shawe-Taylor† and Martin Anthony‡

† Department of Computer Science, Royal Holloway and Bedford New College, University of London, Egham, Surrey TW20 0EX, UK

‡ Department of Mathematics, London School of Economics, University of London, Houghton Street, London WC2A 2AE, UK

Received 22 October 1990

Abstract. This paper applies the theory of probably approximately correct (PAC) learning to multiple-output feedforward threshold networks. It is shown that the sample size for reliable learning can be bounded above by a quantity independent of the number of outputs of the network.

1. Introduction

This paper is inspired by the results of Baum and Haussler [3] bounding the sample sizes required for reliable generalization of a single-output feedforward threshold network. They prove their result using the theory of probably approximately correct (PAC) learning introduced by Valiant [12]. They show that for $0 \leq \epsilon \leq 1/8$, if a sample of size $m \geq m_0 = O((W/\epsilon)\log_2(N/\epsilon))$ is loaded into a feedforward network of linear threshold units with N nodes and W weights, so that a fraction $1 - \epsilon/2$ of the examples are correctly classified, then with confidence approaching certainty the network will correctly classify a fraction $1 - \epsilon$ of future examples drawn according to the same distribution.

More generally the theory of PAC learning is concerned with learning from examples. In order to have predictive power there must be a relation between the training and testing examples. In PAC learning this relation is taken to be an underlying probability distribution which governs how both the training and testing examples are drawn. The strength of the results is that they are independent of the particular distribution which occurs in practice.

In this context PAC learning requires that there is a sample size depending only on a given accuracy parameter ϵ and confidence parameter δ, such that if a hypothesis can be found which agrees with the target on a training sample of at least this size, then with probability $1 - \delta$ the hypothesis found will correctly classify future test examples with probability $1 - \epsilon$. The sample size is required to be polynomially dependent on $1/\epsilon$ and $1/\delta$.

Standard PAC theory applies only to learning Boolean-valued functions or classifications of the input space. Hence the set of hypotheses can be viewed simply as subsets of the input space, being the sets of inputs which give output 1. In many practical applications, however, researchers are interested in training threshold networks with multiple outputs. These might be classifications of inputs under different criteria, or

0954-898X/91/010107+11$03.50

simply the representation of a mapping from one multidimensional space to another. The theory of PAC learning does not, however, readily generalize to functions mapping to ranges other than the Boolean domain.

Recent results by Haussler [8] have shown how the theory can be generalized to neural networks with sigmoid activation functions. This very powerful work gives sample size bounds in these cases, but does not apply to threshold networks as the results rely on Lipschitz bounds for the activation functions. If naively translated for the case which we study in this paper, they also give bounds which are a factor t larger, where t is the number of output units. This factor arises from the fact that the measure of error used by Haussler is the L_1 norm, while we are interested in completely accurate outputs, and so use the discrete metric. In order to guarantee complete agreement on all components of the output we must choose an error bound a factor t smaller in the L_1 norm. This increases the sample size bound by a factor of t.

A parallel study to that of generalization of multilayer perceptrons is into their capabilities. Recently Baum [2] has obtained a result indicating that the Vapnik–Chervonenkis dimension of a multilayer perceptron with d input units and one hidden layer of $\lceil N/d \rceil$ units is at least N, since any set of N inputs in general position can be arbitrarily partitioned by a suitable choice of weights. This indicates that the Vapnik–Chervonenkis dimension is $\Omega(W)$ where W is the number of weights in the network. A paper by Mitchison and Durbin [9] investigates the capability or capacity of multiple-output feedforward networks, extending ideas and results of Cover [5]. Their definition of capacity is the number of inputs at which only half of all the possible functions on those inputs can be implemented by the network. They show that for a network with n inputs, a single layer of h hidden units and an output layer of s units, $s \leq h \leq n$, the capacity m satisfies $2n \leq m \leq nt \log_2 t$, where $t = 1 + h/s$. When $s = 1$ this simplifies to a similar bound $m = O(W \log_2 N)$, where W is the number of variable weights and N the number of computational nodes. In the case of multiple outputs ($s > 1$) the bound indicates that a network will only be able to implement a small fraction of the very large number of possible functions when the value of s is of the same order as the number of hidden units. This, however, does not indicate how large a training set is required to give good generalization with high probability. It is this question which the current paper addresses. The difference between the answers obtained to these two questions in the case of multiple-output networks seems to indicate that the close relationship, between the number of samples that a single hidden-layer network can always classify (subject to general position conditions) and the number required for reliable training, does not carry over from the single-output to the multiple-output case.

Our first expectation for the sample size required in the multiple-output case might be that, since each input/output pair is giving t times as much information in a t-output network (as compared with a single-output network), the sample size would be reduced by a factor of t:

$$m \geq m_0 = O\left(\frac{W}{t\epsilon'} \log_2 \frac{N}{\epsilon'}\right)$$

for a network with W variable weights and N computational nodes. However, by analogy this would be the sample size to guarantee an individual output i is correct with probability $1 - \epsilon'$. Hence the probability that all outputs are correct (i.e. that

the multiple-output is correct in the discrete metric) is

$$(1-\epsilon')^t \geq 1-\epsilon' t.$$

To ensure this is greater than $1-\epsilon$, we must choose ϵ' so that $1-\epsilon' t \geq 1-\epsilon$ or

$$\epsilon' \leq \epsilon/t.$$

This implies a sample size of

$$m \geq m_0 = O\left(\frac{W}{\epsilon}\log_2\frac{Nt}{\epsilon}\right).$$

The main result of this paper will show that in fact

$$m \geq m_0 = O\left(\frac{W}{\epsilon}\log_2\frac{N}{\epsilon}\right)$$

is sufficient.

The paper is organized as follows. Section 2 introduces PAC learning and quotes results of Vapnik and Blumer *et al* relating the Vapnik–Chervonenkis dimension to the sample sizes required for PAC learning, while section 3 discusses the generalization of PAC learning to functions with a finite range. In section 4 we turn to multilayer networks and compute the relevant growth function in this case. This allows us to bound the sample sizes required for reliably training a multiple-output network. In a final section we discuss conclusions and open questions.

2. PAC learning

The theory of Valiant's probably aproximately correct (PAC) learning is concerned with learning from examples of a *target function* (or *concept*), by choosing from a set of functions (the *hypothesis space*) a function meant to be a good approximation to the target. In this framework, we are given a set of inputs and a hypothesis space of functions from the inputs to $\{0,1\}$. There is assumed to be a (usually fixed but unknown) probability distribution on the inputs, and the aim is to find a good approximation to a particular target concept from the hypothesis space, given only a random sample of training examples and the value of the target concept on these examples.

Formally, the input space is a probability space (X,Σ,μ) and the hypothesis space H is a set of measurable functions from X to $\{0,1\}$. The target concept c is assumed to be one of the functions from H. In the simplest form of the standard framework, it is shown that if H has finite Vapnik–Chervonenkis dimension, then there is a sample size, independent of both μ and c such that any hypothesis from H consistent with c on that many examples is likely to be a good approximation to c.

To state this result, we first define the *Vapnik–Chervonenkis dimension* of a hypothesis space. Let S be any set, and let $\mathcal{C}$ be any collection of subsets of S. For $\boldsymbol{s}=(s_1,\ldots,s_m)\in S^m$, let $I(\boldsymbol{s})$ denote the set $\{s_i : 1\leq i\leq m\}$. The integer $\Delta_{\mathcal{C}}(\boldsymbol{s})$ is defined to be the number of distinct sets of the form $A\cap I(\boldsymbol{s})$ as A runs through all members of $\mathcal{C}$, and $\Delta_{\mathcal{C}}(m)$ is defined to be the maximum of $\Delta_{\mathcal{C}}(\boldsymbol{s})$ over all $\boldsymbol{s}\in S^m$.

Clearly, for any m, $\Delta_{\mathcal{C}}(m) \leq 2^m$. We say that the collection $\mathcal{C}$ of sets has finite Vapnik–Chervonenkis dimension (VC dimension) d if

$$\Delta_{\mathcal{C}}(d) = 2^d \qquad \text{but} \qquad \Delta_{\mathcal{C}}(d+1) < 2^{d+1}$$

and that it has infinite VC dimension if it does not have finite VC dimension. When $\mathcal{C}$ has finite VC dimension d, a result of Sauer [11] shows that, for all $m > d$,

$$\Delta_{\mathcal{C}}(m) < \left(\frac{em}{d}\right)^d.$$

The VC dimension of a hypothesis space H of $\{0,1\}$-valued functions is defined to be the VC dimension of the collection $\mathcal{S}$ of supports of the functions in H. In this case, given $\boldsymbol{x} = (x_1, \ldots, x_m) \in X^m$, $\Delta_{\mathcal{S}}(\boldsymbol{x})$ equals the number of distinct vectors of the form $(h(x_1), \ldots, h(x_m))$, as h runs through H.

Given the probability measure μ on X, and the target concept c, we define *actual error* of a hypothesis h from H to be the probability that h and c disagree on a randomly chosen input from X. That is, we define $\mathrm{er}_\mu(h)$ to be $\mu\{x \in X : h(x) \neq c(x)\}$. Recent results have linked sample sizes required for learning to the VC dimension of the hypothesis space [1,4]. For example, in the standard learning model, where the approximating hypothesis is assumed to be consistent with the target on the training sample, we have the following sample-size bound.

Theorem 2.1. If a hypothesis space H has finite VC dimension $d > 1$, then there is $m_0 = m_0(\epsilon, \delta)$ such that if $m > m_0$ then

$$\mu^m\{(x_1, \ldots, x_m) \in X^m : \text{ for all } h \in H, h(x_i) = c(x_i)\ (1 \leq i \leq m) \Rightarrow \mathrm{er}_\mu(h) < \epsilon\} > 1 - \delta.$$

A suitable value of m_0 is

$$m_0 = \frac{1}{\epsilon(1-\sqrt{\epsilon})}\left[\ln\left(\frac{d/(d-1)}{\delta}\right) + 2d\ln\left(\frac{6}{\epsilon}\right)\right].$$

In any real learning situation, where there is a *learning algorithm* for producing the hypothesis supposed to approximate the target, it is unrealistic to assume that the hypothesis produced is consistent with the target on all of the training sample. It is more reasonable to assume only that the hypothesis is consistent with the target on a large proportion of the training sample. To account for this, and to allow the possibility of classification errors during training, the theory has been extended [4] to discuss not the learnability of functions from X to $\{0,1\}$ with an underlying distribution μ, but instead probability distributions on the set $S = X \times \{0,1\}$. We remark that any function c from X to $\{0,1\}$ together with an underlying distribution μ can be realized as a probability measure ν on S. We make the following definitions.

Suppose that ν is some probability measure on $S = X \times \{0,1\}$. We define the *actual error* (with respect to ν) of $h \in H$ to be

$$\mathrm{er}_\nu(h) = \nu\{(x,a) : a \neq h(x)\}.$$

For a subset F of H, we define the *haziness of* F with respect to ν by

$$\mathrm{haz}_\nu(F) = \sup\{\mathrm{er}_\nu(f) : f \in F\}.$$

A *sample* of length m of ν is a sequence $\boldsymbol{x}$ of m points of S, randomly drawn according to the distribution ν. For $h \in H$, the *observed error* of h on sample $\boldsymbol{x} = ((x_1, a_1), \ldots, (x_m, a_m))$ is

$$\mathrm{er}_{\boldsymbol{x}}(h) = \frac{1}{m}\left|\{i : h(x_i) \neq a_i\}\right|.$$

The problem is to determine whether, given $\epsilon, \delta > 0$ and $0 < \gamma \leq 1$, there is a sufficient sample size $m_0 = m_0(\gamma, \epsilon, \delta)$, independent of ν, such that for all $m > m_0$, the following holds: if a hypothesis h from H has observed error less than $(1-\gamma)\epsilon$ on a randomly chosen m-sample from X then, with probability at least $1-\delta$, h has actual error at most ϵ. If such a sample-size bound can be guaranteed then we say that H is PAC learnable, or learnable, extending the standard definition of PAC learnability.

We have the following theorem, a slight modification of a result from [4].

Theorem 2.2. [4] Let H be a hypothesis space of $\{0,1\}$-valued functions defined on an input space X. Let ν 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 ν^m) that, for $\boldsymbol{x} \in S^m$, there is some hypothesis from H such that

$$\mathrm{er}_\nu(h) > \epsilon \qquad \text{and} \qquad \mathrm{er}_{\boldsymbol{x}}(h) \leq (1-\gamma)\mathrm{er}_\nu(h)$$

is at most

$$4\,\Delta_H(2m)\exp(-\tfrac{1}{4}\gamma^2\epsilon m).$$

This leads to the following learnability result.

Proposition 2.1. Let $0 < \epsilon, \delta < 1$ and $0 < \gamma \leq 1$ and let ν be any distribution on $S = X \times \{0,1\}$. If H has finite VC dimension d, then there is $m_0 = m_0(\epsilon, \delta, \gamma)$ such that if $m > m_0$ then, for $\boldsymbol{x} \in S^m$, with probability at least $1-\delta$ (with respect to the product measure ν^m),

$$\mathrm{er}_{\boldsymbol{x}}(h) \leq (1-\gamma)\epsilon \Longrightarrow \mathrm{er}_\nu(h) \leq \epsilon.$$

A suitable value of m_0 is

$$m_0 = \frac{1}{\gamma^2\epsilon(1-\sqrt{\epsilon})}\left[4\ln\left(\frac{4}{\delta}\right) + 6d\ln\left(\frac{4}{\gamma^{2/3}\epsilon}\right)\right].$$

Proof. The proof uses Sauer's inequality mentioned above, and the fact that for any $\alpha, x > 0$, $\ln x \leq (-\ln\alpha - 1) + \alpha x$. We omit the details. □

3. PAC learning with larger range

Instead of considering just $\{0,1\}$-valued functions, we should like to consider functions taking values in some finite or countably infinite set. The same sorts of upper bounds on sufficient sample size in terms of a parameter (which we continue to call the VC dimension) that quantifies in some sense the 'expressibility' of the space of functions can be obtained. For consistency, we want the notion of VC dimension for a space of functions to reduce to the straightforward definition of VC dimension when the range space has only two elements. Various definitions have been proposed.

We adopt a definition of Haussler [7], defining the VC dimension of a space of functions from a set X to a countable set Y to be the VC dimension of the collection of *graphs* of the functions. For any $h \in H$, the graph $\mathcal{G}(h)$ of h is

$$\mathcal{G}(h) = \{(x, h(x)) : x \in X\}$$

and the *graph space* of H is $\mathcal{G}(H) = \{\mathcal{G}(h) : h \in H\}$. Then the VC dimension of H is defined to be the VC dimension of the space $\mathcal{G}(H)$.

We can describe this in another way. For $\boldsymbol{y} = (y_1, \ldots, y_m) \in Y^m$, let $I_{\boldsymbol{y}} : Y^m \to \{0,1\}^m$ be defined by

$$I_{\boldsymbol{y}}((z_1, \ldots, z_m)) = (a_1, \ldots, a_m) \qquad \text{where} \qquad a_i = 1 \iff y_i = z_i.$$

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

$$\Pi_H(\boldsymbol{x}) \leq |\boldsymbol{x}^*(H)| \leq \Delta_H(m)$$

where $\Delta_H(m)$ is the maximum over all $\boldsymbol{x} \in X^m$ of $|\boldsymbol{x}^*(H)|$.

It is easy to see that if $Y = \{0,1\}$, this notion of VC dimension coincides with the standard one. With the above definition of VC dimension, we can apply the previous learnability results. We note that, as earlier, we consider probability distributions on the set $X \times Y$ and not functions from X to Y with underlying probability distributions on X. However, every pair (c, μ) where $c \in H$ and μ is a probability measure on X can be realized by a probability measure $\nu = \nu(c, \mu)$ on the product σ-algebra $\Sigma \times 2^Y$.

Our strategy is to now apply the standard theory of PAC learning of Boolean functions to learning the graphs of functions with the derived distribution ν instead of learning the functions themselves with the distribution on X. There are, however, implications of this approach which we should consider before proceeding.

PAC learning requires that one trains and tests using the same distribution. In the case at hand the distribution is only over positive examples; it is, however, the same distribution in both training and testing. The theorems therefore assure that the hypothesis function produced will (with high probability) correctly classify further examples from this distribution as positive, which of course corresponds to producing a hypothesis such that $h(x) = c(x)$.

We now present two results giving sample sizes required for learning in the countable output case.

Theorem 3.1. If a hypothesis space H of functions from an input space X to a countable set Y has finite VC dimension $d > 1$, then there is an $m_0 = m_0(\epsilon, \delta)$ such that if $m > m_0$ then

$$\mu^m\{(x_1, \ldots, x_m) \in X^m : \text{ for all } h \in H, h(x_i) = c(x_i)\ (1 \leq i \leq m) \Rightarrow \mathrm{er}_\mu(h) < \epsilon\} > 1 - \delta.$$

A suitable value of m_0 is

$$m_0(\epsilon, \delta) = \frac{1}{\epsilon(1 - \sqrt{\epsilon})}\left[\ln\left(\frac{d/(d-1)}{\delta}\right) + 2d\ln\left(\frac{6}{\epsilon}\right)\right].$$

For the case when we allow our hypothesis to incorrectly compute the function on a small fraction of the training sample, we have the following result. Note that we are still considering the discrete metric and so in the case where we are considering multiple-output feedforward networks a single output in error would count as an overall error.

Theorem 3.2. Let $0 < \epsilon < 1$ and $0 < \gamma \leq 1$. Suppose H is a hypothesis space of functions from an input space X to a countable set Y, and let ν be any probability measure on $S = X \times Y$ and $c \in H$ any target concept. Then the probability (with respect to ν^m) that, for $x \in S^m$, there is some $h \in H$ such that

$$\mathrm{er}_\nu(h) > \epsilon \qquad \text{and} \qquad \mathrm{er}_x(h) \leq (1 - \gamma)\mathrm{er}_\nu(h)$$

is at most

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

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

$$m > m_0(\epsilon, \delta, \gamma) = \frac{1}{\gamma^2\epsilon(1 - \sqrt{\epsilon})}\left[4\ln\left(\frac{4}{\delta}\right) + 6d\ln\left(\frac{4}{\gamma^{2/3}\epsilon}\right)\right].$$

4. Sample sizes for multiple-output networks

In artificial neural network research one of the key problems is that of training a network to compute a particular function and to generalize from examples. In this section, we describe a family of such networks, and apply the preceeding theory to extend the results of Baum and Haussler [3] to multiple-output networks.

A feedforward neural network is an ordered pair $\mathcal{N} = (G, \mathcal{F})$, where $G = (V, E)$ is a directed acyclic graph, and $\mathcal{F}$ is a finite set of *activation functions*. V is the disjoint

union of a set I of *input nodes* and a set C of *computation nodes*, and $O \subseteq C$ is a set of *output nodes*. Further, there is a *bias node* $n_0 \in I$. The number of input nodes will be denoted $s+1$ and the number of output nodes t. The underlying graph G is such that all computation nodes are connected to the bias node, and the input nodes have zero in-degree; that is, $E \subseteq (C \cup I) \times C$ and $\{n_0\} \times C \subseteq E$. The computation nodes are labelled with the integers 1 to $n = |C|$ in such a way that if $(i,j) \in E$ then $j > i$. This can be accomplished since G is acyclic. We denote by $d(j)$ the in-degree of computation node j.

Associated with computation node j is the set of states $\Omega_j = \mathcal{R}^{d(j)}$. We let $\Omega^{(k)}$ denote the product $\Omega^{(k)} = \Omega_1 \times \ldots \times \Omega_k$, and denote $\Omega^{(n)}$ simply by Ω (this is the set of all states of the network). Any $\omega \in \Omega$ can be decomposed as $\omega = \omega_1\omega_2\ldots\omega_n$. Given such a decomposition, we denote by w^k the vector $\omega_1\omega_2\ldots\omega_k$.

Each computation node j has associated with it an *activation function*

$$f^j : \Omega_j \times \mathcal{R}^{d(j)} \to \{0,1\}$$

and $\mathcal{F}$ is the set of n activation functions. Writing $w = \omega_j$, the function h^j_w from $\mathcal{R}^{d(j)}$ to $\{0,1\}$ is given by $h^j_w(x) = f^j(w,x)$. H^j denotes the set of functions h^j_w where w runs through Ω_j and we denote $\Delta_{H^j}(m)$ by $\Delta_j(m)$.

An input $x \in \mathcal{R}^s$ to the network consists of an assignment of a real number to each non-bias input node. Further, each node has an output value of 0 or 1. The output of a node is defined recursively in terms of the outputs of the previous nodes. The output of a non-bias input node is defined to be the input on that node, and the output of n_0 is always 1. The *input vector* to computation node j depends on the input x and on ω^{j-1}, and we write it as $I_j(\omega^{j-1}, x) \in \mathcal{R}^{d(j)}$. The output of node j is then computed as

$$f^j\left(\omega_j, I_j(w^{j-1}, x)\right).$$

The function computed by the network when in state $\omega \in \Omega$ is the function F_ω from $\mathcal{R}^s$ to $\{0,1\}^t$ whose value is the $(0,1)$-vector of outputs of the output nodes, O. The set of all F_ω as ω ranges through Ω is denoted F, and we call F the set of functions computable by $\mathcal{N}$.

The *output function* of the network, which describes precisely the output of each computation node, is the function

$$\sigma : \Omega \times X \to \{0,1\}^n.$$

Entry i of $\sigma(\omega, x)$ is 1 if and only if, when the net is in state ω and receives input x, node i has output 1. For a sequence $\boldsymbol{x} = (x_1, \ldots, x_m)$ of inputs, we define $S(\boldsymbol{x})$ to be the number of distinct vectors of the form

$$(\sigma(\omega, x_1), \ldots, \sigma(\omega, x_m))$$

where ω runs through all the states in Ω, and we define $S(m)$ to be the maximum over all $\boldsymbol{x} \in X^m$ of $S(\boldsymbol{x})$. Clearly

$$\Pi_H(m) \leq \Delta_H(m) \leq S(m).$$

We bound $S(m)$ in the following lemma, obtaining the same bound as was obtained in [3] for the case of one output (indeed, the proof makes essentially the same overestimates as were made there).

Lemma 4.1. With the above notation, for any positive integer m,

$$S(m) \leq \prod_{j=1}^{n} \Delta_j(m).$$

Proof. For any i between 1 and n, let $\mathcal{N}_i$ be the subnetwork induced by the input nodes and nodes 1 to i, and let

$$\sigma_i : \Omega^{(i)} \times X \to \{0,1\}^i$$

be the output function of $\mathcal{N}_i$. Further, let $S_i(m)$ be defined for the net $\mathcal{N}_i$ in the same way as $S(m)$ was defined for $\mathcal{N}$. We claim that for any i between 1 and n,

$$S_i(m) \leq \prod_{j=1}^{i} \Delta_j(m)$$

from which the result will follow. We prove the claim by induction.

The base case is easily seen to be true; $S_1(m) = \Delta_1(m)$, since the output function in this case is exactly the output of node 1.

Assume that the claim holds for $i = k-1$ ($k \geq 2$) and consider now the case $i = k$. Observe that, writing $\omega \in \Omega^{(k)}$ as $\omega = w^{k-1}\omega_k$, where $w^{k-1} \in \Omega^{(k-1)}$ and $\omega_k \in \Omega_k$,

$$\sigma_k(\omega^k, x) = \sigma_k(\omega^{k-1}\omega_k, x) = (\sigma_{k-1}(\omega^{k-1}, x), f^k(\omega_k, \mathbf{I}_k(\omega^{k-1}, x))).$$

Thus, for any $x = (x_1, \ldots, x_m) \in X^m$, the number of vectors of the form

$$(\sigma_k(\omega^{k-1}\omega_k, x_1), \ldots, \sigma_k(\omega^{k-1}\omega_k, x_m))$$

as $\omega = \omega^{k-1}\omega_k$ ranges through $\Omega^{(k)}$ is at most $\Delta_k(m)S_{k-1}(m)$, and hence

$$S_k(m) \leq \Delta_k(m)S_{k-1}(m) \leq \Delta_k(m) \prod_{j=1}^{k-1} \Delta_j(m) = \prod_{j=1}^{k} \Delta_j(m). \qquad \square$$

We say that $\mathcal{N}$ is a feedforward linear threshold network in the case when each activation function $f \in \mathcal{F}$ computes the inner product of ω_j with $I_j(\omega^{j-1}, x)$ and outputs 1 if this is positive and 0 otherwise. In this case, H^j has VC dimension $d(j)$, and, as in [3], we have:

Corollary 4.2. Let H be the space of functions computable by a feedforward linear threshold neural network $\mathcal{N}$ with W variable weight, n computational nodes and possibly more than one output node. Then

$$\text{VC dim}(\mathbf{H}) \leq 2W \log_2(en).$$

In particular, the VC dimension of the network can be bounded independently of the number of output nodes.

Natarajan [10] has shown that for (not necessarily feedforward) linear threshold nets with n nodes (including inputs), the VC dimension is at most of the order of $n^3 \ln n$. The above result shows that it is in fact at most $n^2 \ln n$ for the case of feedforward linear threshold nets with n computation nodes. What is more Natarajan only considers the networks as functions of Boolean vector inputs, a restriction which we do not require.

The result, together with theorems 3.1 and 3.2, provides upper bounds on the size of a training sample required for the network to give valid generalization, in the cases when the training performance is required to be exact and in the case when an allowance for error is made on the training set.

Corollary 4.3. Given an accuracy parameter ϵ and a confidence parameter δ, for a feedforward network with W variable weights and n computational nodes, with probability greater than $1-\delta$ the network will give correct output with probability greater than $1-\epsilon$ on inputs drawn according to some distribution, provided it correctly computes the function on a sample (drawn from the same distribution) of size at least

$$m_0 = m_0(\epsilon, \delta) = \frac{1}{\epsilon(1-\sqrt{\epsilon})}\left[\ln\left(\frac{1.3}{\delta}\right) + 4(W+n)\log_2(en)\ln\left(\frac{6}{\epsilon}\right)\right].$$

Corollary 4.4. Given an accuracy parameter ϵ and a confidence parameter δ, for a feedforward network with W variable weights and n computational nodes, with probability greater than $1-\delta$ the network will give correct output with probability greater than $1-\epsilon$ on inputs drawn according to some distribution, provided it correctly computes the function on a fraction of $1-(1-\gamma)\epsilon$ of a sample (drawn from the same distribution) of size at least

$$m_0 = m_0(\epsilon, \delta, \gamma) = \frac{1}{\gamma^2\epsilon(1-\sqrt{\epsilon})}\left[4\ln\left(\frac{4}{\delta}\right) + 12(W+n)\log_2(en)\ln\left(\frac{4}{\gamma^{2/3}\epsilon}\right)\right].$$

5. Conclusions

This paper has considered the problem of estimating sample sizes required for PAC learning of functions with countable ranges. This is of particular interest in the case of feedforward networks, where sample-size estimates are only available for the single-output case. The results show that the same sample size is sufficient as was required in the single-output case. It appears that the extra information contained in each sample is cancelled out by the more stringent training we are effectively requiring by using the discrete metric.

The bounds we have obtained are, however, only upper bounds as in the case of the single-output networks. As mentioned in the introduction Baum [2] has shown that the VC dimension of a multilayer perceptron with d inputs and one hidden layer of $\lceil N/d \rceil$ units is at least N. Using general lower bounds on sample sizes for learning [6]

in hypothesis spaces with given VC dimension, we can conclude that we need samples of size at least

$$m \geq \max\left(\frac{1-\epsilon}{\epsilon}\ln\frac{1}{\delta}, \frac{N-1}{32\epsilon}\right)$$

which is $\Omega(W/\epsilon)$. It is not known whether the two log factors, $\log_2(en)$ and $\ln(1/\epsilon)$, are real in the single- or multiple-output cases.

References

[1] Anthony M, Biggs N and Shawe-Taylor J 1990 The learnability of formal concepts *Proc. COLT '90 (Rochester, NY)* ed M Fulk and J Case (San Mateo, CA: Morgan Kaufmann) pp 246–57

[2] Baum E B 1988 On the capabilities of multilayer perceptrons *J. Complexity* **4** 193–215

[3] Baum E and Haussler D 1989 What size net gives valid generalization? *Neural Computation* **1** 151–60

[4] Blumer A, Ehrenfeucht A, Haussler D and Warmuth M K 1989 Learnability and the Vapnik–Chervonenkis dimension *J. ACM* **36** 929–65

[5] Cover T M 1965 Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition *IEEE Trans. Electron. Comput.* **EC-14** 326–34

[6] Ehrenfeucht A, Haussler D, Kearns M and Valiant L G 1989 A general lower bound on the number of examples needed for learning *Inf. Comput.* **82** 247–61

[7] Haussler D 1989 unpublished

[8] Haussler D 1989 Generalizing the PAC model for neural net and other learning applications *Technical Report* UCSC-CRL-89-30, University of California Computer Research Laboratory, Santa Cruz, CA

[9] Mitchison G J and Durbin R M 1989 Bounds on the learning capacity of some multi-layer networks *Biol. Cybern.* **60** 345–56

[10] Natarajan B K 1989 On learning sets and functions *Machine Learning* **4** 67–97

[11] Sauer N 1972 On the density of families of sets *J. Combin. Theory* A **13** 145–7

[12] Valiant L G A theory of the learnable 1984 *Commun. ACM* **27** 1134–42