INFORMATION AND COMPUTATION 118, 65–72 (1995)

Sample Sizes for Threshold Networks with Equivalences

JOHN SHAWE-TAYLOR

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

This paper applies the theory of Probably Approximately Correct (PAC) learning to multiple output feedforward threshold networks in which the weights conform to certain equivalences. It is shown that the sample size for reliable learning can be bounded above by a formula similar to that required for single output networks with no equivalences. The best previously obtained bounds are improved for all cases. © 1995 Academic Press, Inc.

1. INTRODUCTION

This paper develops the results of Baum and Haussler (1989) bounding the sample sizes required for reliable generalisation of a single output feedforward threshold network. They prove their result using the theory of Probably Approximately Correct (PAC) learning introduced by Valiant (1984). They show that for $0 < \varepsilon \leqslant 1/2$, if a sample of size

$$m \geqslant m_0 = \frac{64W}{\varepsilon} \log \frac{64N}{\varepsilon}$$

is loaded into a feedforward network of linear threshold units with N nodes and W weights, so that a fraction $1-\varepsilon/2$ of the examples are correctly classified, then with confidence approaching certainty the network will correctly classify a fraction $1-\varepsilon$ of future examples drawn according to the same distribution. Similarly if a sample of size

$$m \geqslant m_0 = \frac{32W}{\varepsilon} \log \frac{32N}{\varepsilon}$$

is loaded with no errors then with confidence approaching certainty the network will correctly classify a fraction $1-\varepsilon$ of future examples drawn according to the same distribution. The results below will imply a significant improvement to both of these bounds.

In many cases training can be simplified if known properties of a problem can be incorporated into the structure of a network before training begins. One such technique is described by Shawe-Taylor (1989), though many similar techniques have been applied, as for example in TDNN's (Lang and Hinton, 1988). The effect of these restrictions is to constrain groups of weights to take the same value and learning algorithms are adapted to respect this constraint.

In this paper we consider the effect of this restriction on the generalisation performance of the networks and in particular the sample sizes required to obtain a given level of generalisation. This extends the work by Baum and Haussler (1989), described above by improving their bounds and also improves the results of Shawe-Taylor and Anthony (1991), who consider generalisation of multiple-output threshold networks. The remarkable fact is that in all cases the formula obtained is the same, where we now understand the number of weights W to be the number of weight classes, but N is still the number of computational nodes.

The paper is organised as follows. Section 2 introduces threshold networks and the various equivalences considered and states our main results. Section 3 gives an introduction to PAC learning and develops the necessary theory required to prove the main results. Section 4 proves important growth bounds for the networks considered, while Section 5 gives the proofs of the main results. In a final section we discuss conclusions and open questions.

2. DEFINITIONS AND MAIN RESULTS

2.1. *Symmetry and Equivalence Networks*

We begin with a definition of threshold networks. To simplify the exposition it is convenient to incorporate the threshold value into the set of weights. This is done by creating a distinguished input that always has value 1 and is called the threshold input. The following is a formal notation for these systems.

A network $\mathcal{N} = (C, I, O, n_0, E)$ is specified by a set C of computational nodes, a set I of input nodes, a subset $O \subseteq C$ of output nodes, and a node $n_0 \in I$ called the threshold node. The connectivity is given by a set $E \subseteq (C \cup I) \times C$ of connections, with $\{n_0\} \times C \subseteq E$.

With network $\mathcal{N}$ we associate a weight function on the set of connections:

$$w: E \to \mathcal{R}.$$

We say that the network $\mathcal{N}$ is in state w.

0890-5401/95 $6.00

For notational convenience choose a numbering for all nodes and refer to each node by its number. For a connection $(i, j) \in E$ we denote $w((i, j))$ by w_{ij}.

For the purposes of this paper we will assume throughout that if node i is connected to node j then $j > i$. Such a numbering can always be found provided the network is cycle free. This is the feedforward condition on the connectivity.

For a feedforward network it makes sense to introduce the *level* of a node. This is defined as the number of connections in the longest (directed) path from an input node. Input nodes are at level 0 while nodes connected only to input nodes are at level 1, etc. Let l_i denote the level of node i. The feedforward condition on the numbering will be satisfied by requiring

$$l_j > l_i \Rightarrow j > i,$$

since if node i is connected to node j then certainly $l_j > l_i$.

At a given time the input values to the whole network are specified by a function $\mathbf{i}$ from the set of input nodes other than n_0 to $[0, 1]$:

$$\mathbf{i}: I \backslash \{n_0\} \rightarrow [0, 1].$$

The subset $\{0, 1\}$ of $[0, 1]$ may be the range or more generally any subinterval of the set $\mathscr{R}$ of real numbers. The precise domain considered does not affect any of the analysis or results. Each node also has an output value which is the weighted sum of its inputs thresholded at 0. These values are given by the function

$$o: C \rightarrow \{0, 1\}.$$

The output of the whole network is the vector of the values of this function on the set of output nodes. We denote this vector function of the weights and inputs by $F_{\mathscr{N}}(w, \mathbf{i})$.

An automorphism γ of a network $\mathscr{N} = (C, I, O, n_0, E)$ is a bijection

$$\gamma: I \cup C \rightarrow I \cup C,$$

such that $\gamma(I) = I$, $\{n_0\} \cup O \subseteq \text{fix}(\gamma)$, and the induced action on $(I \cup C) \times C$ maps E to itself. Note that if Γ is a set of automorphisms $\text{fix}(\Gamma)$ is the set of nodes that are fixed by all the automorphisms of Γ.

We say that an automorphism γ preserves the weight assignment w if $w_{ij} = w_{(\gamma i)(\gamma j)}$ for all $i \in I \cup C, j \in C$.

Let γ be an automorphism of a network $\mathscr{N} = (C, I, O, n_0, E)$ and let

$$\mathbf{i}: I \backslash \{n_0\} \rightarrow [0, 1]$$

be an input to $\mathscr{N}$. We denote by $\mathbf{i}^\gamma$ the input given by

$$\mathbf{i}^\gamma(k) = \mathbf{i}(\gamma^{-1}k)$$

for $k \in I \backslash \{n_0\}$.

The following theorem is a natural generalisation of part of the Group Invariance Theorem of Minsky and Pappert (1988) to multi-layer perceptrons.

THEOREM 2.1 (Shawe-Taylor, 1989). *Let γ be a weight preserving automorphism of the network $\mathscr{N} = (C, I, O, n_0, E)$ in state w. Then for every input vector* $\mathbf{i}$

$$F_{\mathscr{N}}(w, \mathbf{i}) = F_{\mathscr{N}}(w, \mathbf{i}^\gamma).$$

Following this theorem it is natural to consider the concept of a *symmetry network* introduced by Shawe-Taylor (1989). This is a pair $(\mathscr{N}, \Gamma)$, where $\mathscr{N}$ is a network and Γ a group of weight preserving automorphims of $\mathscr{N}$. We will also refer to the automorphisms as symmetries.

The theorem suggests a practical way of simplifying the task of training networks required to be invariant under certain automorphisms. This is done by constructing the network so that the group of automorphisms acts on the whole network as weight preserving symmetries and then constraining weight updates to retain those symmetries. In this way network output will at all times be invariant under the automorphisms. Hence to train such a network we need train it with each input in only one "position." For a symmetry network $(\mathscr{N}, \Gamma)$, we term the orbits of the connections E under the action of Γ the weight classes.

Finally we introduce the concept of an *equivalence network*. This definition abstracts from the symmetry networks precisely those properties we require to obtain our results. The class of equivalence networks is, however, far larger than that of symmetry networks and includes many classes of networks studied by other researchers (Lang and Hinton, 1988; Le Cun, 1988).

DEFINITION 2.2. An equivalence network is a threshold network in which an equivalence relation is defined on both connections and nodes. The relation is required to be trivial on the threshold and output nodes. Further, the relations on nodes and connections are required to be compatible in that connections in the same class are between pairs of nodes in corresponding classes, while there is a 1–1 correspondence between the connection classes of connections to nodes in the same node class. The connections in an equivalence class are at all times required to have equal weight values.

The definition does not prescribe that the input nodes should all fall in the same class; indeed, every threshold network can be viewed as an equivalence network by taking the trivial equivalence relations for all nodes. We now show that symmetry networks are indeed equivalence networks with the same weight classes.

LEMMA 2.3. *A symmetry network $(\mathscr{N}, \Gamma)$ is an equivalence network, where the equivalence classes are the orbits of connections and nodes respectively.*

Proof. Consider two equivalent weights w connecting node u to node v and w' connecting node u' to v'. By the definition of equivalence in the symmetry network there is an automorphism γ which takes the connection (u, v) to (u', v') and so v and v' are in the same equivalence class and so are u and u'.

Let u and u' be two equivalent nodes. Then there is an automorphism γ of $\mathcal{N}$ which takes u to u'. The automorphism γ induces a weight class preserving bijection between the inputs to node u and those of node u'. Hence u and u' have the same set of input connection types. The fact that $\{n_0\} \cup O \subseteq \mathrm{fix}(\Gamma)$ implies that the relation is trivial on these nodes, as required. ▮

We will require one technical lemma concerning the structure of equivalence networks.

LEMMA 2.4. *Let $\mathcal{N}$ be an equivalence network and let $\mathcal{C}$ be the set of classes of nodes. Then there is an indexing of the classes, C_i, $i = 1, ..., m$, such that nodes in C_i do not have connections from nodes in C_j for $j \geqslant i$.*

Proof. The result will follow if we can show that there are no directed cycles in the directed graph with vertices the equivalence classes of nodes and connections showing the presence of connections between nodes in two classes. This follows from the fact that the graph described is a quotient graph of the network itself by the equivalence relation and so if such a directed cycle exists, we will be able to find a directed cycle in the original network contradicting the feedforward condition. ▮

If we view the network as a function whose input–output performance depends on parameters, the standard algorithms can be readily adapted to update the parameters of an equivalence network. In this case the parameters are values for the weights of each equivalence class.

For example, this can be done for the generalised delta rule (Rumelhart *et al.*, 1986) and the linear programming algorithm (Shawe-Taylor and Cohen, 1990).

2.2. *Main Results*

We are now in a position to state our main results. Note that throughout this paper log means natural logarithm, while an explicit subscript is used for other bases.

THEOREM 2.5. *Let $\mathcal{N}$ be an equivalence network with W weight classes and N computational nodes in n equivalence classes. If the network correctly computes a function on a set of m inputs drawn independently according to a fixed probability distribution, where*

$$m \geqslant m_0(\varepsilon, \delta) = \frac{1}{\varepsilon(1-\sqrt{\varepsilon})}\left[\log\left(\frac{1.3 \times 2^n}{\delta}\right) + 2(W-n)\log\left(\frac{6\sqrt{N}}{\varepsilon}\right)\right],$$

then with probability at least $1-\delta$ the error rate of the network will be less than ε on inputs drawn according to the same distribution.

THEOREM 2.6. *Let $\mathcal{N}$ be an equivalence network with W weight classes and N computational nodes in n equivalence classes. If the network correctly computes a function on a fraction $1-(1-\gamma)\varepsilon$ of m inputs drawn independently according to a fixed probability distribution, where*

$$m \geqslant m_0(\varepsilon, \delta, \gamma) = \frac{1}{\gamma^2\varepsilon(1-\sqrt{\varepsilon/N})}\left[4\log\left(\frac{2^{n+2}}{\delta}\right) + 6(W-n)\log\left(\frac{4N}{\gamma^{2/3}\varepsilon}\right)\right],$$

then with probability at least $1-\delta$ the error rate of the network will be less than ε on inputs drawn according to the same distribution.

3. THEORETICAL BACKGROUND

3.1. *PAC Learning*

We begin with an informal introduction to PAC learning. 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 $1-\delta$ that hypothesis will correctly classify future test examples with probability $1-\varepsilon$. The sample size is required to be polynomially dependent on $1/\varepsilon$ and $1/\delta$ and further that a learning algorithm exists which is polynomial in the size of its input. For the purposes of this paper we will neglect the requirement that a polynomial algorithm exists, since the problem of training a threshold network is known to be NP-complete. Without this requirement PAC learning is often referred to as PAC learnability and is concerned only with generalisation performance under the assumption that the network has been successfully trained. Clearly the task of training is considerably simplified when equivalences are introduced into a network of given size.

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 simply the representation of a mapping from one multi-dimensional space to another. The theory of PAC learning has been generalised to cover these cases (see Haussler, 1989, and Shawe-Taylor and Anthony, 1991).

3.2. *Definitions and Previous Results*

In order to present results for binary outputs ($\{0, 1\}$ functions) and larger ranges in a unified way we will consider throughout the task of learning the graph of a function. All the definitions reduce to the standard ones when the outputs are binary.

We consider learning from examples as selecting a suitable function from a set H of hypotheses, being functions from a space X to set Y, which has at most countable size. At all times we consider an (unknown) target function

$$c: X \to Y$$

which we are attempting to learn. To this end the space X is required to be a probability space (X, Σ, μ), with appropriate regularity conditions so that the sets considered are measurable (Blumer *et al.*, 1989). In particular, the hypotheses should be measurable when Y is given the discrete topology, as should the error sets defined below. The space $S = X \times Y$ is equipped with a σ-algebra $\Sigma \times 2^Y$ and measure $\nu = \nu(\mu, c)$, defined by its value on sets of the form $U \times \{y\}$:

$$\nu(U \times \{y\}) = \mu(U \cap c^{-1}(y)).$$

Using this measure the error of a hypothesis is defined to be

$$\mathrm{er}_\nu(h) = \nu\{(x, y) \in S \mid h(x) \neq y\}.$$

The introduction of ν allows us to consider samples being drawn from S, as they will automatically reflect the output value of the target. This approach freely generalises to stochastic concepts though we will restrict ourselves to target functions for the purposes of this paper. The error of a hypothesis h on a sample $\mathbf{x} = ((x_1, y_1), ..., (x_m, y_m)) \in S^m$ is defined to be

$$\mathrm{er}_\mathbf{x}(h) = \frac{1}{m} |\{i \mid h(x_i) \neq y_i\}|.$$

We also define the VC dimension of a set H of hypotheses by reference to the product space S. Consider a sample $\mathbf{x} = ((x_1, y_1), ..., (x_m, y_m)) \in S^m$ and the function

$$\mathbf{x}^*: H \to \{0, 1\}^m,$$

given by $\mathbf{x}^*(h)_i = 1$ if and only if $h(x_i) = y_i$, for $i = 1, ..., m$. We can now define the growth function $B_H(m)$ as

$$B_H(m) = \max_{\mathbf{x} \in S^m} |\{\mathbf{x}^*(h) \mid h \in H\}| \leqslant 2^m.$$

The Vapnik–Chervonenkis dimension of a hypothesis space H is defined as

$$\mathrm{VCdim}(H) = \begin{cases} \infty; & \text{if } B_H(m) = 2^m, \text{ for all } m; \\ \max\{m \mid B_H(m) = 2^m\}; & \text{otherwise.} \end{cases}$$

In the case of a threshold network $\mathcal{N}$, the set of functions obtainable using all possible weight assignments is termed the hypothesis space of $\mathcal{N}$ and we will refer to it as $\mathcal{N}$. For a threshold network $\mathcal{N}$, we also introduce the state growth function $S_{\mathcal{N}}(m)$. This is defined by first considering all computational nodes to be output nodes, and then counting different output sequences.

$$S_{\mathcal{N}}(m) = \max_{\mathbf{x} = (\mathbf{i}_1, ..., \mathbf{i}_m) \in X^m} |\{(F_{\mathcal{N}'}(w, \mathbf{i}_1), F_{\mathcal{N}'}(w, \mathbf{i}_2), ..., F_{\mathcal{N}'}(w, \mathbf{i}_m)) \mid w: E \to \mathscr{R}\}|,$$

where $X = [0, 1]^{|I|}$ and $\mathcal{N}'$ is obtained from $\mathcal{N}$ by setting $O = C$. We clearly have that for all $\mathcal{N}$ and m, $B_{\mathcal{N}}(m) \leqslant S_{\mathcal{N}}(m)$.

We are now in a position to state three important theorems that we shall require for our analysis.

THEOREM 3.1 (Anthony *et al.*, 1990). *If a hypothesis space H has growth function $B_H(m)$ then for any $\varepsilon > 0$ and $k > m$ and*

$$0 < r < 1 - \frac{1}{\sqrt{\varepsilon k}}$$

the probability that there is a function in H which agrees with a randomly chosen m sample and has error greater than ε is less than

$$\frac{\varepsilon k(1-r)^2}{\varepsilon k(1-r)^2 - 1} B_H(m+k) \exp\left\{-r\varepsilon \frac{km}{m+k}\right\}.$$

This result can be used to obtain the following bound on sample size required for PAC learnability of a hypothesis space with VC dimension d. The theorem improves the bounds reported by Blumer *et al.* (1989).

THEOREM 3.2 (Anthony *et al.*, 1990). *If a hypothesis space H has finite VC dimension $d > 1$, then there is $m_0 = m_0(\varepsilon, \delta)$ such that if $m > m_0$ then the probability that a hypothesis consistent with a randomly chosen sample of size* *m has error greater than ε is less than δ. A suitable value of m_0 is*

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

For the case where 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 3.3 (Shawe-Taylor and Anthony, 1991). *Let $0<\varepsilon<1$ and $0<\gamma\leqslant 1$. Suppose H is a hypothesis space of functions from an input space X to a possibly countable set Y, and let v be any probability measure on $S = X \times Y$. Then the probability (with respect to v^m) that, for $\mathbf{x} \in S^m$, there is some $h \in H$ such that*

$$\mathrm{er}_v(h) > \varepsilon \quad \textit{and} \quad \mathrm{er}_{\mathbf{x}}(h) \leqslant (1-\gamma)\,\mathrm{er}_v(h)$$

is at most

$$4B_H(2m)\exp\left(-\frac{\gamma^2\varepsilon m}{4}\right).$$

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

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

4. THE GROWTH FUNCTION FOR EQUIVALENCE NETWORKS

We will bound the number of output sequences $B_{\mathcal{N}}(m)$ for a number m of inputs by the number of distinct state sequences $S_{\mathcal{N}}(m)$ that can be generated from the m inputs by different weight assignments. This follows the approach taken by Shawe-Taylor and Anthony (1991).

THEOREM 4.1. *Let $\mathcal{N}$ be an equivalence network with W weight equivalence classes and a total of N computational nodes in n equivalence classes. Then we can bound $S_{\mathcal{N}}(m)$ by*

$$S_{\mathcal{N}}(m) \leqslant 2^n\left(\frac{Nem}{W-n}\right)^{W-n}.$$

Proof. Let C_i, $i = 1, \ldots, n$, be the equivalence classes of computational nodes indexed as guaranteed by Lemma 2.4 (ignoring the equivalence classes of input and threshold nodes and making a suitable renumbering) with $|C_i| = c_i$. Further let the number of distinct equivalence classes of input connections not including the threshold input for nodes in C_i be n_i (i.e., the actual number of input connection equivalence classes is $n_i + 1$). This number is independent of the choice of vertex in the class by the definition of an equivalence network. Then

$$\sum_{i=1}^{n} c_i = N$$

and

$$\sum_{i=1}^{n} n_i = W - n$$

Denote by $\mathcal{N}_j$ the network obtained by taking only the first j node equivalence classes. We will prove by induction that

$$S_{\mathcal{N}_j}(m) \leqslant \prod_{i=1}^{j} B_i(mc_i),$$

where B_i is the growth function for nodes in the class C_i. We take $\mathcal{N}_0$ to be the network composed of only the input nodes, i.e., no computational nodes. Hence we may take $S_{\mathcal{N}_0}(m) = 1$, since there can be no adaption of the function computed. Clearly the above result holds for $j = 0$, since an empty product has value 1. Assuming inductively that the result holds for $j \geqslant 0$, consider the addition of the $(j+1)$st class to create the network $\mathcal{N}_{j+1}$ from the network $\mathcal{N}_j$. There are $S_{\mathcal{N}_j}(m)$ distinguishable weight assignments to the initial part of the network. For each of these assignments a particular sequence of m inputs is presented to the class C_{j+1}. We must determine an upper bound on the number of different state sequences that can be generated from each of these input sequences by choosing the weights for inputs to nodes in C_{j+1}. The particular set of nodes connected to a node in C_{j+1} will in general be different for different nodes in C_{j+1}. For each input to the whole network consider the outputs of each of these sets collated together to give c_{j+1} inputs to a single representative from C_{j+1}. For each distinct state sequence over the c_{j+1} nodes in the class in response to the m original input vectors, there will correspond a different output sequence of this single node in response to the mc_{j+1} collated inputs. Hence the growth function for the single node bounds the growth function for the states in the equivalence network. By considering the growth function for this individual node we obtain that the number of distinguishable functions obtainable is bounded by $B_{j+1}(mc_{j+1})$. In other words there will be at most this many essentially different weight assignments, giving distinct sequences of output vectors on the nodes in C_{j+1} in response to the m inputs. Hence we obtain

$$S_{\mathcal{N}_{j+1}}(m) \leqslant S_{\mathcal{N}_j}(m)\, B_{j+1}(mc_{j+1})$$

and the claim follows by induction.

Consider now the growth function of a single node u in C_i. Let v be a node with one input for each of the $n_i + 1$ equivalence classes of connections to u. If the inputs to v are the sum of the inputs on the corresponding class of connections to u, the two nodes determine an identical mapping from weight assignments to outputs. Hence their growth functions are identical. It is well known that the growth function of a threshold node with n_i non-threshold inputs is bounded by

$$B_i(m) = 2 \sum_{j=0}^{n_i} \binom{m-1}{j} \leqslant 2 \left(\frac{em}{n_i}\right)^{n_i}.$$

It should be noted that we have made slightly more efficient use of the bound on the growth function than was made by Baum and Haussler (1989). They had the same formula without the factor of 2 but with $n_i + 1$ in place of n_i. The difference is obtained by using the Cover (1965), bound on the growth function rather than that derived from the VC dimension of the node. This gives a bound on the function

$$S_{\mathcal{N}}(m) \leqslant \prod_{i=1}^{n} 2 \left(\frac{emc_i}{n_i}\right)^{n_i}.$$

Consider the function

$$f(x) = x \log x.$$

This is a convex function and so for a set of values $x_1, ..., x_M$, we have

$$\frac{1}{M} \sum_{i=1}^{M} x_i \log x_i \geqslant \frac{X}{M} \log \frac{X}{M},$$

where $X = \sum_{i=1}^{M} x_i$. Consider taking the x's to be c_i copies of n_i/c_i for each $i = 1, ..., n$. We obtain

$$\frac{1}{N} \sum_{i=1}^{n} n_i \log \frac{n_i}{c_i} \geqslant \frac{W-n}{N} \log \frac{W-n}{N}.$$

Hence

$$\prod_{i=1}^{n} \left(\frac{c_i}{n_i}\right)^{n_i} \leqslant \left(\frac{N}{W-n}\right)^{W-n},$$

and so

$$S_{\mathcal{N}}(m) \leqslant 2^n \left(\frac{emN}{W-n}\right)^{W-n},$$

as required. ∎

The bounds we have obtained make it possible to bound the Vapnik–Chervonenkis dimension of equivalence networks. Though we will not need these results, we give them here for completeness.

PROPOSITION 4.2. *The Vapnik–Chervonenkis dimension of an equivalence network $\mathcal{N}$ with W weight classes and $N \geqslant 4$ computational nodes in n equivalence classes where each computational node is connected to at least 2 non-threshold inputs is bounded by*

$$2(W-n) \log_2 eN.$$

Proof. Consider $m = 2(W-n) \log_2 eN$. We will use the bound on the growth function to show that in this case $S_{\mathcal{N}}(m) < 2^m$, implying that $m > \mathrm{VCdim}(\mathcal{N})$. Substituting the value of m into the bound on $S_{\mathcal{N}}(m)$ gives the inequality we must show as

$$2^W (eN \log_2(eN))^{W-n} < 2^{2(W-n)\log_2 eN}.$$

Taking the logarithm to the base 2 gives

$$W + (W-n) \log_2(eN \log_2(eN)) < 2(W-n) \log_2 eN,$$

or

$$W < (W-n) \log_2(eN/\log_2(eN)).$$

Note that $N \geqslant 4$ implies that $eN > 2^{1.5} \log_2(eN)$ and

$$\log_2(eN/\log_2(eN)) > 1.5.$$

This implies that it will be sufficient to show that

$$W \leqslant 1.5(W-n),$$

but this must follow if each node has at least 2 non-threshold connections. ∎

5. PROOF OF MAIN RESULTS

Using the results of the last section we are now in a position to prove Theorems 2.5 and 2.6.

Proof of Theorem 2.5. We use Theorem 3.1 which bounds the probability that a hypothesis with error greater than ε can match an m-sample. Substituting our bound on the growth function of an equivalence network and choosing

$$k = \left\lceil m\left(\frac{r\varepsilon m}{W-n} - 1\right)\right\rceil$$

and

$$r = 1 - \sqrt{\frac{W-n}{\varepsilon k}},$$

we obtain the following bound on the probability

$$\left(\frac{W-n}{W-n-1}\right) 2^n \left(\frac{e^4 \varepsilon m^2}{(W-n)^2}\right)^{W-n} N^{W-n} \exp(-\varepsilon m).$$

By choosing m greater than

$$\frac{1}{\varepsilon(1-\sqrt{\varepsilon})}\left[\log\left(\frac{1.3}{\delta}\right) + 2(W-n)\log\left(\frac{6\sqrt{N}}{\varepsilon}\right)\right]$$

we guarantee that the above probability is less than $\delta 2^n$. Hence taking $m > m_0$, where m_0 is given by

$$m_0 = m_0(\varepsilon, \delta) = \frac{1}{\varepsilon(1-\sqrt{\varepsilon})}\left[\log\left(\frac{1.3}{\delta 2^{-n}}\right) + 2(W-n)\log\left(\frac{6\sqrt{N}}{\varepsilon}\right)\right],$$

gives the required probability less than δ. ∎

Our second main result can be obtained more directly.

Proof of Theorem 2.6. We use Theorem 3.3 which bounds the probability that a hypothesis with error greater than ε can match all but a fraction $(1-\gamma)$ of an m-sample. The bound on the sample size is obtained from the probability bound by using the inequality

$$B_H(2m) \leqslant \left(\frac{e2m}{d}\right)^d.$$

In other words, the bound on m guarantees that

$$4\left(\frac{e2m}{d}\right)^d \exp\left(-\frac{\gamma^2 \varepsilon m}{4}\right) < \delta.$$

By adjusting the parameters we will convert the probability expression to that obtained by substituting our growth function. We can then read off a sample size by making the same substitution in the sample size formula. Consider setting $d = W-n$, $\varepsilon = \varepsilon'/N$ and $m = Nm'$. With these substitutions the bound for the probability is

$$4B_H(2Nm')\exp\left(-\frac{\gamma^2 \varepsilon' m'}{4}\right).$$

When we use the above inequality for the growth function this gives 2^{-n} times the quantity we require to be less than δ:

$$4\left(\frac{2eNm'}{W-n}\right)^{W-n} \exp\left(-\frac{\gamma^2 \varepsilon' m'}{4}\right).$$

Making the substitutions in the sample size formula gives

$$m' = \frac{1}{\gamma^2 \varepsilon'(1-\sqrt{\varepsilon'/N})}\left[4\log\left(\frac{4}{\delta}\right) + 6(W-n)\log\left(\frac{4N}{\gamma^{2/3}\varepsilon'}\right)\right].$$

This sample size is sufficient to guarantee the probability is less than $2^n \delta$. To ensure the probability is less than δ we must therefore take a sample of size

$$\frac{1}{\gamma^2 \varepsilon'(1-\sqrt{\varepsilon'/N})}\left[4\log\left(\frac{2^{n+2}}{\delta}\right) + 6(W-n)\log\left(\frac{4N}{\gamma^{2/3}\varepsilon'}\right)\right],$$

as required. ∎

6. CONCLUSION

The problem of training feedforward neural networks remains a major hurdle to the application of this approach to large scale systems. A very promising technique for simplifying the training problem is to include equivalences in the network structure which can be justified by a priori knowledge of the application domain. This paper has extended previous results concerning sample sizes for feed-forward networks to cover so called equivalence networks in which weights are constrained in this way. At the same time we have improved the sample size bounds previously obtained for standard threshold networks (Baum and Haussler, 1989) and multiple output networks (Shawe-Taylor and Anthony, 1989).

The results are of the same order as previous results and imply similar bounds on the Vapnik–Chervonenkis dimension, namely $2(W-n)\log_2 eN$, though the free parameters have been reduced from W to $W-n$. They perhaps give circumstantial evidence for the conjecture that the $\log_2 eN$ factor in this expression is real, in that the same expression obtains even if the number of computational nodes is increased by expanding the equivalence classes of weights. Equivalence networks may be a useful area in which to search for high growth functions and perhaps show that for certain classes the VC dimension is $\Omega((W-n)\log N)$.

Note added in proof. The conjecture has subsequently been proved true in: Maass, W. (1993), On the complexity of learning on neural networks, *in* "Proceedings of First European Conference on Computational Learning Theory," EuroCOLT '93, pp. 1–17.

Received September 21, 1992; final manuscript received September 14, 1993

REFERENCES

Anthony, A., Biggs, N., and Shawe-Taylor, J. (1990), "Learnability and Formal Concept Analysis," Technical Report CSD-TR-624, Department of Computer Science, Royal Holloway and Bedford New College, University of London.

Anthony, M., Biggs, N., and Shawe-Taylor, J. (1990), The learnability of formal concepts, *in* "Proceedings of Third Workshop on Computational Learning Theory," pp. 246–257.

Baum, E., and Haussler, D. (1989), What size net gives valid generalization, *Neural Comput.* **1**, 151–160.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. (1989), Learnability and the Vapnik–Chervonenkis dimension, *J. Assoc. Comput. Mach.* **36**, 929–965.

Cover, T. (1965), Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition, *IEEE Trans. Electron. Comput.* **14**, 326–334.

Haussler, D. (1989), Preliminary extended abstract, for "COLT '89."

Lang, K., and Hinton, G. (1988), "The Development of TDNN Architecture for Speech Recognition," Technical Report CMU-CS-88-152, Carnegie–Mellon University.

Le Cun, Y. (1988), A theoretical framework for back propagation, *in* "Connectionist Models: A Summer School" (D. Touretzsky, Ed.), Morgan Kaufmann, San Mateo, CA.

Minsky, M., and Papert, S. (1988), "Perceptrons, Expanded Edition," MIT Press, Cambridge, MA.

Rumelhart, D., Hinton G., and Williams, R. (1986), Learning representations by back-propagating errors, *Nature* **323**, 533–536.

Sauer, N. (1972), On the density of families of sets, *J. Combin. Theory Ser. A* **13**, 145-147.

Shawe-Taylor, J. (1989), Building symmetries into feedforward network architectures, *in* "Proceedings of First IEE Conference on Artificial Neural Networks," pp. 158–162.

Shawe-Taylor, J., and Anthony, M. (1991), Sample sizes for multiple output feedforward networks, *Network* **2**, 107–117.

Shawe-Taylor, J., and Cohen, D. (1990), The linear programming algorithm, *Neural Networks* **3**, 575–582.

Valiant, L. (1984), A theory of the learnable, *Comm. ACM* **27**, 1134–1142.