

PAC Learning and Artificial Neural Networks

Martin Anthony and Norman Biggs

Department of Mathematics, London School of Economics and Political Science (University of London), Houghton St., London WC2A 2AE, United Kingdom

Running head: PAC learning and artificial neural networks

Corresponding author: Martin Anthony.

Telephone: +0719557623

email: `anthony@vax.lse.ac.uk`

INTRODUCTION

In this article, we discuss the ‘probably approximately correct’ (PAC) learning paradigm as it applies to artificial neural networks. The PAC learning model is a probabilistic framework for the study of learning and generalization. It is useful not only for neural classification problems, but also for learning problems more often associated with mainstream artificial intelligence, such as the inference of Boolean functions. In PAC theory, the notion of successful learning is formally defined using probability theory. Very roughly speaking, if a large enough sample of randomly drawn training examples is presented, then it should be likely that, after learning, the neural network will classify most other randomly drawn examples correctly. The PAC model formalises the terms ‘likely’ and ‘most’. Furthermore, the learning algorithm must be expected to act quickly, since otherwise it may be of little use in practice.

There are thus two main emphases in PAC learning theory. First, there is the issue of how many training examples should be presented. Secondly, there is the question of whether learning can be achieved using a fast algorithm. These are known, respectively, as the *sample complexity* and *computational complexity* problems. This article provides a brief introduction to these. We highlight the importance of the Vapnik-Chervonenkis dimension, a combinatorial parameter which measures the ‘expressive power’ of a neural network, and describe how this parameter quantifies fairly precisely the sample complexity of PAC learning. In discussing the computational complexity of PAC learning, we shall present a result which illustrates that in some cases the problem of PAC learning is inherently intractable.

PAC LEARNING

Basic Definitions

In this section, we describe the basic ‘probably approximately correct’ (PAC) model of learning introduced by Valiant (1984). This model is applicable to neural networks with one output unit which outputs either the value 0 or 1; thus, it applies to *classification* problems. In the PAC model, it is assumed that the neural network receives a sequence of *examples* x , each labelled with the value $t(x)$ of the particular *target function* which is being ‘learned’. A fundamental assumption of this model is that these examples are presented independently and at random according to some fixed (but unknown) probability distribution on the set of all examples.

We first explain how to formalise the notion of generalization. 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 that the target function t can be computed by the neural network in some

state. A **training sample for t** of length m is an element \mathbf{s} of $(X \times \{0, 1\})^m$, of the form

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

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 desired that the resulting state is an approximation to the target function.

Probability and Approximation

If $L(\mathbf{s})$ is the function computed by the network after training sample $\mathbf{s} \in S(m, t)$ has been presented and learning algorithm L has been applied, one way in which to assess the success of the learning process is to measure how close $L(\mathbf{s})$ is to t . Since there is assumed to be some probability distribution, P , on the set of all examples, and since t takes only the values 0 or 1, we may define the **error**, $\text{er}_P(h, t)$, of a function h (with respect to t) to be the P -probability that a randomly chosen example is classified incorrectly by h . In other words,

$$\text{er}_P(h, t) = P(\{x \in X : h(x) \neq t(x)\}).$$

The aim is to ensure 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 P , the sample vector \mathbf{x} is drawn randomly from X^m according to the product probability distribution P^m . Thus, more formally, we want it to be true that with high P^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.

The learning algorithm L is a **PAC-learning algorithm** for the network if *for any* given $\delta, \epsilon > 0$ there is a sample length $m_0(\delta, \epsilon)$ such that *for all* target functions t computable by the network and *for all* probability distributions P on the set of examples, we have

$$m \geq m_0(\delta, \epsilon) \Rightarrow P^m(\{\mathbf{s} \in S(m, t) : \text{er}_P(L(\mathbf{s}), t) > \epsilon\}) < \delta.$$

In other words, provided the sample has length at least $m_0(\delta, \epsilon)$ then it is ‘probably’ the case that after training on that sample, the function computed by the network is ‘approximately’ correct. (We should note that the product probability distribution P^m is really defined not on subsets of $S(m, t)$ but on sets of 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 P occurs twice in the definition: first in the requirement that the P^m -probability of a sample be small and secondly through the fact that the error of $L(\mathbf{s})$ is measured with reference to P . The crucial feature of the definition is that we require that the sample length $m_0(\delta, \epsilon)$ be independent of P and of t . It is not immediately clear

that this is possible, but the following informal arguments explain why it can be done. If a particular example has not been seen in a large sample \mathbf{s} , the chances are that this example has low probability (with respect to P) and therefore misclassification of that example contributes little to the error of the function $L(\mathbf{s})$. In other words, the penalty paid for misclassification of a particular example is its probability, and, very loosely speaking, the two occurrences of the probability distribution in the definition can therefore ‘balance’ or ‘cancel’ each other.

The Finite Case

We shall show that if the network computes only a finite number of functions (for example, when the weights of a neural network are restricted to a finite set of allowed values), then there is a PAC learning algorithm for the network.

We say that the learning algorithm L is **consistent** if, given any training sample $\mathbf{s} = ((x_1, t(x_1)), (x_2, t(x_2)), \dots, (x_m, t(x_m)))$, the functions $L(\mathbf{s})$ and t agree on x_i , for each i between 1 and m . Such a condition seems quite natural. We should note, however, that neither the standard on-line perceptron learning algorithm nor the on-line backpropagation algorithm are, in general, consistent. But the batch versions of these algorithms, in which one repeatedly cycles through the training sample until no further changes are required *are*, consistent algorithms.

Suppose that the network is capable of computing a total of M different functions and let t be any one of these. If h is computable by the network and has error $\epsilon_h \geq \epsilon$ with respect to t and P , then the probability (with respect to the product distribution P^m) that h agrees with t on a random sample is clearly at most $(1 - \epsilon_h)^m$. This is at most $\exp(-\epsilon_h m)$, using a standard approximation. Thus, since there are certainly at most M such functions h , the probability that *some* function computable by the network has error at least ϵ *and* is consistent with a randomly chosen sample \mathbf{s} is at most $M \exp(-\epsilon m)$. For any fixed positive δ , this probability is less than δ provided

$$m \geq m_0(\delta, \epsilon) = \frac{1}{\epsilon} \log \left(\frac{M}{\delta} \right),$$

a bound independent of both the distribution and the target function.

This analysis shows that if a network only computes a finite number of functions, then there is a PAC learning algorithm for the network and, moreover, *any* consistent learning algorithm for the network is a PAC learning algorithm. The argument fails if the network in question computes infinitely many functions and it is not immediately clear that PAC learning is possible in such circumstances. In the next section, we present a theory which shows that, in many such cases, it is possible.

PAC LEARNING AND THE VC-DIMENSION

The Vapnik-Chervonenkis Dimension

In this section, we show how the problem of PAC learning can be addressed by means of a combinatorial parameter known as the Vapnik-Chervonenkis dimension (henceforth, VC-dimension). Suppose \mathcal{N} is a neural network which outputs 0 or 1 and suppose that \mathcal{N} accepts examples from a set X (for example, $X = \mathbf{R}^n$ where n is the number of inputs). We say that a set T of examples is **shattered** by \mathcal{N} if for each of the $2^{|T|}$ possible ways of dividing T into two disjoint sets T_1 and T_0 , there is *some* function f computable by \mathcal{N} such that $f(x) = 1$ if $x \in T_1$ and $f(x) = 0$ if $x \in T_0$. In what follows, it is sometimes convenient to say that x is a positive (resp. negative) example of f if $f(x) = 1$ (resp. $f(x) = 0$). The **VC-dimension** of \mathcal{N} , denoted $\text{VCdim}(\mathcal{N})$, is defined to be the largest size of a set of examples shattered by \mathcal{N} . The VC-dimension may be thought of as a measure of the ‘expressive power’ of the network, although Vapnik and Chervonenkis (1971) defined this parameter in a more general context and not specifically in the context of neural networks. It should be noted that the notion of Vapnik-Chervonenkis dimension is, in a sense, an extension to that of linear (or vector-space) dimension. Dudley (1978) proved that if \mathcal{F} is a vector space of real functions defined on a set X and if, for $f \in \mathcal{F}$, we define $f_+ : X \rightarrow \{0, 1\}$ by $f_+(x) = 1 \iff f(x) > 0$, then the VC-dimension of $\{f_+ : f \in \mathcal{F}\}$ is the linear dimension of \mathcal{F} .

It is instructive at this stage to determine the VC-dimension of the simplest neural network, the **simple real perceptron** \mathcal{P}_n on n inputs. This network consists of n real-valued inputs, each of which is connected by a weighted connection to the single, linear threshold, output unit. (The weights can be any real numbers.) It is clear that, for functions computable by \mathcal{P}_n , the sets of positive examples and negative examples are separated by a hyperplane.

Theorem 1 *For any positive integer n , let \mathcal{P}_n be the simple real perceptron with n inputs. Then*

$$\text{VCdim}(\mathcal{P}_n) = n + 1.$$

Proof Let T be *any* set of $n + 2$ examples. It can be shown that there is a non-empty subset T_1 of $E_{\mathbf{x}}$ such that, if $T_0 = T \setminus T_1$, then $\text{conv}(T_1) \cap \text{conv}(T_0) \neq \emptyset$, where $\text{conv}(A)$ denotes the convex hull of A . (This follows from Radon’s theorem, which may be found in Grunbaum (1967), for instance.) It follows immediately that the sets T_1 and T_0 cannot be separated by a hyperplane; in other words, there can be no function f computable by \mathcal{P}_n such that $f(x) = 1$ if $x \in T_1$ and $f(x) = 0$ if $x \in T_0$. Therefore T is not shattered and the VC-dimension of \mathcal{P}_n must be at most $n + 1$. It remains to prove the reverse inequality. Let o denote the origin of \mathbf{R}^n and, for $1 \leq i \leq n$, let e_i be the point with a 1 in the i th coordinate and all other coordinates 0. Then \mathcal{P}_n shatters the set $T = \{o, e_1, e_2, \dots, e_n\}$ of $n + 1$ examples. To see this, suppose that $T_1 \subseteq T$. For $i = 1, 2, \dots, n$, let α_i be 1 if $e_i \in T_1$

and -1 otherwise, and let θ be $-1/2$ if $o \in T_1$, $1/2$ otherwise. Then it is straightforward to verify that if h is the function computed by the perceptron when the threshold is θ and the weights are $\alpha_1, \alpha_2, \dots, \alpha_n$, then $h(x) = 1$ if $x \in T_1$ and $h(x) = 0$ if $x \in T_0$. Therefore T is shattered and, consequently, $\text{VCdim}(\mathcal{P}_n) \geq n + 1$. \square

Finite VC-Dimension Characterises PAC Learning

We have observed that if \mathcal{N} computes only a finite number of functions, then any consistent learning algorithm is a PAC algorithm, and a value of $m_0(\delta, \epsilon)$ involving the number of computable functions can be determined. It turns out that, as far as PAC learning is concerned, it is not the size of the set of computable functions which is crucial, but the *VC-dimension* of the network. More precisely, we have the following key result, due to Blumer *et al.* (1989) and Ehrenfeucht *et al.* (1989).

Theorem 2 *If a neural network \mathcal{N} has finite VC-dimension $d \geq 1$, then any consistent learning algorithm L for \mathcal{N} is a PAC learning algorithm. Moreover, there is a constant K such that a sufficient sample length $m_0(\delta, \epsilon)$ for any such algorithm is*

$$K\epsilon^{-1} (d \ln(\epsilon^{-1}) + \ln(\delta^{-1})).$$

On the other hand, there is a constant c such that for any PAC learning algorithm for \mathcal{N} , the sufficient sample length $m_0(\delta, \epsilon)$ must be at least $c\epsilon^{-1} (d + \ln(\delta^{-1}))$, for all $\epsilon \leq 1/8$ and $\delta \leq 1/100$.

In fact, an analogue of Theorem 2 holds for general classes of $\{0, 1\}$ -valued functions, and not simply those computable by neural networks.

VC-Dimension of Neural Networks

We now discuss some results on the VC-dimensions of certain types of network. A more detailed treatment of this topic may be found in VAPNIK-CHEVONENKIS DIMENSION OF NEURAL NETS. First, we start with the feedforward linear threshold network. The first part of the following result is due to Baum and Haussler (1989) and the second part is due to Maass (1993a).

Theorem 3 *There is $K > 0$ such that, if \mathcal{N} is any feedforward linear threshold network having W variable weights and thresholds and N threshold units, then $\text{VCdim}(\mathcal{N}) \leq KW \log N$. Furthermore, there is $c > 0$ such that some feedforward linear threshold networks having W weights and N threshold units have VC-dimension at least $cW \log N$; in other words, the upper bound is tight to within a constant.*

Anthony and Holden (1993) have investigated the VC-dimension of a general type of network first introduced in the 1960s, and which includes simple RADIAL BASIS FUNCTION NETWORKS and ‘polynomial discriminators’. A *linearly-weighted* neural network, with n real inputs and a single boolean output, is defined by a fixed set $\phi_1, \phi_2, \dots, \phi_k$ 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 $\mathbf{w} = (w_1, w_2, \dots, w_k)$. The output corresponding to example $x \in \mathbf{R}^n$ and state \mathbf{w} is 1 or 0 according as the weighted sum $\sum_{i=1}^k w_i \phi_i(x)$ is positive or not. For example, a **polynomial discriminator** is obtained when all the functions ϕ_i are monomials, that is, products of the components of x such as $x_1^2 x_3 x_n^3$. The *degree* d of a polynomial discriminator is the maximum total degree of any ϕ_i , in the usual sense. Anthony and Holden (1993) proved that the VC-dimension of a polynomial discriminator of degree d is at most $\binom{n+d}{d}$, and that if one only permits $\{0, 1\}$ -valued inputs, the VC-dimension is at most $\sum_{i=1}^d \binom{n}{i}$. The **radial basis function networks** 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 ‘centre’ y_i : $\phi_i(x) = \phi(\|x - y_i\|)$, where ϕ is a fixed function and $\|z\|$ is the usual Euclidean norm of z . Anthony and Holden showed that the VC-dimension of a radial basis function network of this form is equal to k , the number of basis functions, if $\phi(r)$ takes any one of the forms r , $\exp(-cr^2)$, $(r^2 + c^2)^\alpha$, $(r^2 + c^2)^{-\beta}$, in which α and β are positive constants with $\beta < 1$.

Other work on the VC-dimension of neural networks includes that of MacIntyre and Sontag (1993) and Goldberg and Jerrum (1993). In both of these papers, techniques from logic are used to study the VC-dimension of neural networks of certain types. The first paper proves, among other things, the finiteness of the VC-dimension of feedforward networks in which the output unit is a linear threshold and all other computational units have the standard sigmoid activation function $f(x) = 1/(1 + e^{-x})$. However, explicit bounds on the VC-dimension are not given. In the second paper, upper bounds are obtained for

the VC-dimension of networks similar to that just described, but in which the activation functions are not the standard sigmoid ones.

THE COMPUTATIONAL COMPLEXITY OF PAC LEARNING

Efficiency with Respect to Accuracy, Example Size and Sample Length

Thus far, a learning algorithm has been defined as a function which maps training samples into hypotheses. We shall now be more specific about the computational effectiveness of this function. If the process of PAC learning by an algorithm L is to be of practical value, it should be possible to implement the algorithm ‘quickly’. We wish to quantify the behaviour of a learning algorithm for a particular neural network architecture with respect to the size of the network. In particular, we wish to consider how the running time of the algorithm varies with the number n of inputs to the network: for a learning algorithm to be efficient, this running time should increase polynomially with n . However, there is another important consideration in any discussion of efficiency. 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 time taken to produce a probably approximately correct output should be constrained in some appropriate way as ϵ decreases; the appropriate condition is that the running time must be polynomial in $1/\epsilon$. Formally, we say that a learning algorithm L is *efficient with respect to accuracy ϵ , example size n and sample length m* if its running time is polynomial in the length m of the training sample and if there is a value of $m_0(\delta, \epsilon)$ sufficient for PAC learning which is polynomial in n and ϵ^{-1} .

Hardness Results

In complexity theory, two important classes of problems, RP and NP, are defined. The class RP is the class of all problems which can be solved by ‘randomised’ algorithms in polynomial time, while NP is the class of problems which can be solved by non-deterministic Turing machines in polynomial time. (We refer the reader to the book by Cormen, Leiserson and Rivest (1990).) It is conjectured, and widely believed, that these classes are not the same; more precisely, it is believed that RP is a strict subset of NP. This is known as the ‘RP \neq NP’ conjecture. For fixed k , for each n , let \mathcal{P}_n^k be the neural network which consists of k linear threshold units, each connected to all of n inputs, the outputs of these threshold networks then being combined together by a hard-wired AND gate. Thus, the network outputs 1 if and only if all k threshold units output 1. Blum and Rivest (1988) proved (essentially) the following result. (See also Anthony and Biggs (1992).)

Theorem 4 *Let \mathcal{P}_n^k be as described, where $k \geq 2$. If there is a PAC learning algorithm for \mathcal{P}_n^k which is efficient with respect to accuracy, example size and number of inputs then the ‘RP \neq NP’ conjecture is false.*

Thus it is extremely unlikely that there is an efficient PAC learning algorithm for this surprisingly simple class of neural networks.

DISCUSSION

We have considered basic PAC learning as it applies to learning in artificial neural networks. There are two distinct aspects: the length of training sample to be used, and the efficiency of learning. In other words, we have the *sample complexity* problem and the *computational complexity* problem. The Vapnik-Chervonenkis dimension of a neural network determines in a fairly precise way the length of sample sufficient for PAC learning. This dimension can, in many cases, be related to the structure of the network, as in the examples presented here. Techniques from computational complexity theory can be applied to show that in a number of cases, *efficient* algorithmic PAC learning is impossible unless the $NP \neq RP$ conjecture is false.

There are several recent important extensions and generalizations of the PAC model which can be applied to artificial neural networks. We refer the reader to the survey of Anthony and Biggs (1993) and to the references cited there. One very important recent paper is that of Haussler (1992). This paper studies ways in which the PAC model may be extended in order to discuss the learning of functions whose values are other than simply 0 or 1. In particular, Haussler bounds the length of training sample which should be used for valid learning in neural networks with real-valued outputs. There have also been new approaches to and new results on the computational complexity problem; see Maass (1993b), for example.

REFERENCES

- * Anthony, M. and Biggs, N. , 1992, Computational Learning Theory: An Introduction, Cambridge, UK: Cambridge University Press.
- * Anthony, M. and Biggs, N. , 1993, Computational learning theory for artificial neural networks, Mathematical Approaches to Neural Networks (J.G. Taylor, ed.): North Holland Mathematical Library, North Holland (Elsevier), pp. 25–63.

Anthony, M. and Holden, S.B. , On the power of polynomial discriminators and radial basis function networks. In Proc. 6th Annual A.C.M. Conference on Computational Learning Theory, New York: A.C.M. Press, pp. 158–164.

Baum, E.B. and Haussler, D. , 1989, What size net gives valid generalization?, Neural Computation, 1:151–160.

Blum, A. and Rivest, R.L. , 1988, Training a 3-node neural network is NP-complete, in

Proceedings of the 1988 Workshop on Computational Learning Theory, San Mateo: Morgan Kaufmann, pp. 9–18. (See also: Neural Networks, 5 (1), 1992: 117–127.)

Blumer, A. , Ehrenfeucht, A. , Haussler, D. and Warmuth, M.K. , 1989, Learnability and the Vapnik-Chervonenkis dimension, Journal of the A.C.M., 36(4):929–965.

* Cormen, T.H. , Leiserson, C.E. and Rivest, R.L. , 1990, Introduction to Algorithms, Cambridge, MA: MIT Press.

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

Ehrenfeucht, A. , Haussler, D. , Kearns, M. and Valiant, L. , 1989, A general lower bound on the number of examples needed for learning, Information and Computation, 82(3):247–261.

Goldberg, P. and Jerrum, M. , 1993, Bounding the Vapnik-Chervonenkis Dimension of concept classes parameterized by real numbers, Proc. of the Sixth Annual A.C.M. Conference on Computational Learning Theory, New York: A.C.M. Press, pp. 361–369.

Grunbaum, B. , 1967, Convex Polytopes, London: John Wiley.

Haussler, D. , 1992, Decision theoretic generalizations of the pac model for neural net and other learning applications, Information and Computation, 100:78–150.

Maass, W. , 1993a, Bounds on the computational power and learning complexity of analog neural nets, in Proceedings of the Twenty-Fifth Annual A.C.M. Symposium on the Theory of Computing, pp. 335–344. (See also Maass, W., Neural nets with superlinear VC-dimension, to appear in Neural Computation.)

Maass, W. , 1993b, Agnostic PAC-learning of functions on analog neural nets, to appear in Neural Computation (extended abstract appears in Advances in Neural Information Processing Systems 6).

MacIntyre, A. and Sontag, E.D. , 1993, Finiteness results for sigmoidal “neural” networks, in Proceedings of the Twenty-Fifth Annual A.C.M. Symp. on the Theory of Computing, pp. 325–334.

Valiant, L.G. , 1984, A theory of the learnable, Comm. ACM, 27(11):1134–1142.

Vapnik, V.N. and Chervonenkis, A.Ya. , 1971, On the uniform convergence of relative frequencies of events to their probabilities, Theory of Probability and its Applications,

16(2):264-280.