Chapter 8

Computational Learning Theory

In chapter one we posed the problem of guessing a function given a set of sample inputs and their values. We gave some intuitive arguments to support the claim that after seeing only a small fraction of the possible inputs (and their values) that we could guess *almost correctly* the values of *most* subsequent inputs—if we knew that the function we were trying to guess belonged to an appropriately restricted subset of functions. That is, a given training set of sample patterns might be adequate to allow us to select a function, *consistent with the labeled samples*, from among a restricted set of hypotheses such that with high *probability* the function we select will be *approximately correct* (small probability of error) on subsequent samples drawn at random according to the *same distribution* from which the labeled samples were drawn. This insight led to the theory of *probably approximately correct (PAC)* learning—initially developed by Leslie Valiant [Valiant, 1984]. We present here a brief description of the theory for the case of Boolean functions. [Dietterich, 1990, Haussler, 1988, Haussler, 1990] give nice surveys of the important results.

Other overviews?

8.1 Notation and Assumptions for PAC Learning Theory

We assume a training set Ξ of n-dimensional vectors, $\mathbf{X}_i$, $i = 1, \ldots, m$, each labeled (by 1 or 0) according to a target function, f, which is unknown to

the learner. The probability of any given vector $\mathbf{X}$ being in Ξ, or later being presented to the learner, is $P(\mathbf{X})$. The probability distribution, P, can be arbitrary. (In the literature of PAC learning theory, the target function is usually called the target *concept* and is denoted by c, but to be consistent with our previous notation we will continue to denote it by f.) Our problem is to guess a function, $h(\mathbf{X})$, based on the labeled samples in Ξ. In PAC theory such a guessed function is called the *hypothesis*. We assume that the target function is some element of a set of functions, $\mathcal{C}$. We also assume that the hypothesis, h, is an element of a set, $\mathcal{H}$, of hypotheses, which includes the set, $\mathcal{C}$, of target functions. $\mathcal{H}$ is called the *hypothesis space*.

In general, h won't be identical to f, but we can strive to have the value of $h(\mathbf{X})$ = the value of $f(\mathbf{X})$ for *most* $\mathbf{X}$'s. That is, we want h to be *approximately* correct. To quantify this notion, we define the *error* of h, ε_h, as the probability that an $\mathbf{X}$ drawn randomly according to P will be misclassified:

$$\varepsilon_h = \sum_{[\mathbf{X}:h(\mathbf{X})\neq f(\mathbf{X})]} P(\mathbf{X})$$

Boldface symbols need to be smaller when they are subscripts in math environments.

We say that h is *approximately* (except for ε) *correct* if $\varepsilon_h \leq \varepsilon$, where ε is the *accuracy parameter*.

Suppose we are able to find an h that classifies *all* m randomly drawn training samples correctly; that is, h is consistent with this randomly selected training set, Ξ. If m is large enough, will such an h be approximately correct (and for what value of ε)? On some training occasions, using m randomly drawn training samples, such an h might turn out to be approximately correct (for a given value of ε), and on others it might not. We say that h is *probably* (except for δ) *approximately correct (PAC)* if the probability that it is approximately correct is greater than $1-\delta$, where δ is the *confidence parameter*. We shall show that if m is greater than some bound whose value depends on ε and δ, such an h is guaranteed to be probably approximately correct.

In general, we say that a learning algorithm *PAC-learns* functions from $\mathcal{C}$ in terms of $\mathcal{H}$ iff for every function $f\epsilon\ \mathcal{C}$, it outputs a hypothesis $h\epsilon\ \mathcal{H}$, such that with probability at least $(1-\delta)$, $\varepsilon_h \leq \varepsilon$. Such a hypothesis is called *probably* (except for δ) *approximately* (except for ε) *correct*.

We want learning algorithms that are tractable, so we want an algorithm that PAC-learns functions in polynomial time. This can only be done for certain classes of functions. If there are a finite number of hypotheses in a hypothesis set (as there are for many of the hypothesis sets we have considered), we could always produce a consistent hypothesis from this

set by testing all of them against the training data. But if there are an exponential number of hypotheses, that would take exponential time. We seek training methods that produce consistent hypotheses in less time. The time complexities for various hypothesis sets have been determined, and these are summarized in a table to be presented later.

A class, $\mathcal{C}$, is *polynomially PAC learnable* in terms of $\mathcal{H}$ provided there exists a polynomial-time learning algorithm (polynomial in the number of samples needed, m, in the dimension, n, in $1/\varepsilon$, and in $1/\delta$) that PAC-learns functions in $\mathcal{C}$ in terms of $\mathcal{H}$.

Initial work on PAC assumed $\mathcal{H} = \mathcal{C}$, but it was later shown that some functions cannot be polynomially PAC-learned under such an assumption (assuming P $\neq$ NP)—but *can* be polynomially PAC-learned if $\mathcal{H}$ is a strict superset of $\mathcal{C}$! Also our definition does not specify the distribution, P, from which patterns are drawn nor does it say anything about the properties of the learning algorithm. Since $\mathcal{C}$ and $\mathcal{H}$ do not have to be identical, we have the further restrictive definition:

A *properly PAC-learnable* class is a class $\mathcal{C}$ for which there exists an algorithm that polynomially PAC-learns functions from $\mathcal{C}$ in terms of $\mathcal{C}$.

8.2 PAC Learning

8.2.1 The Fundamental Theorem

Suppose our learning algorithm selects some h randomly from among those that are consistent with the values of f on the m training patterns. The probability that the error of this randomly selected h is *greater* than some ε, with h consistent with the values of $f(\mathbf{X})$ for m instances of $\mathbf{X}$ (drawn according to arbitrary P), is less than or equal to $|\mathcal{H}|e^{-\varepsilon m}$, where $|\mathcal{H}|$ is the number of hypotheses in $\mathcal{H}$. We state this result as a theorem [Blumer, *et al.*, 1987]:

Theorem 8.1 (Blumer, *et al.***)** *Let $\mathcal{H}$ be any set of hypotheses, Ξ be a set of $m \geq 1$ training examples drawn independently according to some distribution P, f be any classification function in $\mathcal{H}$, and $\varepsilon > 0$. Then, the probability that there exists a hypothesis h consistent with f for the members of Ξ but with error greater than ε is at most $|\mathcal{H}|e^{-\varepsilon m}$.*

Proof:

Consider the set of all hypotheses, $\{h_1, h_2, \ldots, h_i, \ldots, h_S\}$, in $\mathcal{H}$, where $S = |\mathcal{H}|$. The error for h_i is ε_{h_i}= the probability that h_i will classify a pattern in error (that is, differently than f would classify it). The probability

that h_i will classify a pattern correctly is $(1-\varepsilon_{h_i})$. A subset, $\mathcal{H}_B$, of $\mathcal{H}$ will have error greater than ε. We will call the hypotheses in this subset *bad*. The probability that any particular one of these bad hypotheses, say h_b, would classify a pattern correctly is $(1-\varepsilon_{h_b})$. Since $\varepsilon_{h_b} > \varepsilon$, the probability that h_b (or any other bad hypothesis) would classify a pattern correctly is less than $(1-\varepsilon)$. The probability that it would classify *all* m independently drawn patterns correctly is then less than $(1-\varepsilon)^m$.

That is,

prob[h_b classifies all m patterns correctly $|h_b \in \mathcal{H}_B] \leq (1-\varepsilon)^m$.

prob[*some* $h \in \mathcal{H}_B$ classifies all m patterns correctly]
$= \sum_{h_b \in \mathcal{H}_B}$ prob[h_b classifies all m patterns correctly $|h_b \in \mathcal{H}_B$]
$\leq K(1-\varepsilon)^m$, where $K = |\mathcal{H}_B|$.

That is,

prob[there is a bad hypothesis that classifies all m patterns correctly]
$\leq K(1-\varepsilon)^m$.

Since $K \leq |\mathcal{H}|$ and $(1-\varepsilon)^m \leq e^{-\varepsilon m}$, we have:

prob[there is a bad hypothesis that classifies all m patterns correctly]
= prob[there is a hypothesis with error $> \varepsilon$ and that classifies all m patterns correctly] $\leq |\mathcal{H}|e^{-\varepsilon m}$.

□

A corollary of this theorem is:

Corollary 8.2 *Given* $m \geq (1/\varepsilon)(\ln|\mathcal{H}| + \ln(1/\delta))$ *independent samples, the probability that there exists a hypothesis in* $\mathcal{H}$ *that is consistent with* f *on these samples and has error greater than* ε *is at most* δ.

Proof: We are to find a bound on m that guarantees that

prob[there is a hypothesis with error $> \varepsilon$ and that classifies all m patterns correctly] $\leq \delta$. Thus, using the result of the theorem, we must show that $|\mathcal{H}|e^{-\varepsilon m} \leq \delta$. Taking the natural logarithm of both sides yields:

$$\ln|\mathcal{H}| - \varepsilon m \leq \ln\delta$$

or

$$m \geq (1/\varepsilon)(\ln |\mathcal{H}| + \ln(1/\delta))$$

□

This corollary is important for two reasons. First it clearly states that we can select *any* hypothesis consistent with the m samples and be assured that with probability $(1 - \delta)$ its error will be less than ε. Also, it shows that in order for m to increase no more than polynomially with n, $|\mathcal{H}|$ can be no larger than $2^{O(n^k)}$. No class larger than that can be guaranteed to be properly PAC learnable.

Here is a possible point of confusion: The bound given in the corollary is an *upper bound* on the value of m needed to guarantee polynomial probably approximately correct learning. Values of m greater than that bound are sufficient (but might not be necessary). We will present a lower (necessary) bound later in the chapter.

8.2.2 Examples

Terms

Let $\mathcal{H}$ be the set of terms (conjunctions of literals). Then, $|\mathcal{H}| = 3^n$, and

$$m \geq (1/\varepsilon)(\ln(3^n) + \ln(1/\delta))$$

$$\geq (1/\varepsilon)(1.1n + \ln(1/\delta))$$

Note that the bound on m increases only polynomially with n, $1/\varepsilon$, and $1/\delta$.

For $n = 50$, $\varepsilon = 0.01$ and $\delta = 0.01$, $m \geq 5,961$ guarantees PAC learnability.

In order to show that terms are *properly PAC learnable*, we additionally have to show that one can find in time polynomial in m and n a hypothesis h consistent with a set of m patterns labeled by the value of a term. The following procedure for finding such a consistent hypothesis requires $O(nm)$ steps (adapted from [Dietterich, 1990, page 268]):

We are given a training sequence, Ξ, of m examples. Find the first pattern, say $\mathbf{X}_1$, in that list that is labeled with a 1. Initialize a Boolean function, h, to the conjunction of the n literals corresponding to the values

of the n components of $\mathbf{X}_1$. (Components with value 1 will have corresponding positive literals; components with value 0 will have corresponding negative literals.) If there are no patterns labeled by a 1, we exit with the null concept ($h \equiv 0$ for all patterns). Then, for each additional pattern, $\mathbf{X}_i$, that is labeled with a 1, we delete from h any Boolean variables appearing in $\mathbf{X}_i$ with a sign different from their sign in h. After processing all the patterns labeled with a 1, we check all of the patterns labeled with a 0 to make sure that none of them is assigned value 1 by h. If, at any stage of the algorithm, any patterns labeled with a 0 are assigned a 1 by h, then there exists no term that consistently classifies the patterns in Ξ, and we exit with failure. Otherwise, we exit with h.

Change this paragraph if this algorithm was presented in Chapter Three.

As an example, consider the following patterns, all labeled with a 1 (from [Dietterich, 1990]):

$(0, 1, 1, 0)$

$(1, 1, 1, 0)$

$(1, 1, 0, 0)$

After processing the first pattern, we have $h = \overline{x_1}x_2x_3\overline{x_4}$; after processing the second pattern, we have $h = x_2x_3\overline{x_4}$; finally, after the third pattern, we have $h = x_2\overline{x_4}$.

Linearly Separable Functions

Let $\mathcal{H}$ be the set of all linearly separable functions. Then, $|\mathcal{H}| \leq 2^{n^2}$, and

$$m \geq (1/\varepsilon)\big(n^2 \ln 2 + \ln(1/\delta)\big)$$

Again, note that the bound on m increases only polynomially with n, $1/\varepsilon$, and $1/\delta$.

For $n = 50$, $\varepsilon = 0.01$ and $\delta = 0.01$, $m \geq 173,748$ guarantees PAC learnability.

To show that linearly separable functions are *properly PAC learnable*, we would have additionally to show that one can find in time polynomial in m and n a hypothesis h consistent with a set of m labeled linearly separable patterns.

Linear programming is polynomial.

8.2.3 Some Properly PAC-Learnable Classes

Some properly PAC-learnable classes of functions are given in the following table. (Adapted from [Dietterich, 1990, pages 262 and 268] which also gives references to proofs of some of the time complexities.)

$\mathcal{H}$	$\lvert\mathcal{H}\rvert$	Time Complexity	P. Learnable?
terms	3^n	polynomial	yes
k-term DNF (k disjunctive terms)	$2^{O(kn)}$	NP-hard	no
k-DNF (a disjunction of k-sized terms)	$2^{O(n^k)}$	polynomial	yes
k-CNF (a conjunction of k-sized clauses)	$2^{O(n^k)}$	polynomial	yes
k-DL (decision lists with k-sized terms)	$2^{O(n^k k \lg n)}$	polynomial	yes
lin. sep.	$2^{O(n^2)}$	polynomial	yes
lin. sep. with (0,1) weights	?	NP-hard	no
k-2NN	?	NP-hard	no
DNF (all Boolean functions)	2^{2^n}	polynomial	no

(Members of the class k-2NN are two-layer, feedforward neural networks with exactly k hidden units and one output unit.)

Summary: In order to show that a class of functions is *Properly PAC-Learnable* :

1. Show that there is an algorithm that produces a consistent hypothesis on m n-dimensional samples in time polynomial in m and n.
2. Show that the sample size, m, needed to ensure PAC learnability is polynomial (or better) in $(1/\varepsilon)$, $(1/\delta)$, and n by showing that $\ln|\mathcal{H}|$ is polynomial or better in the number of dimensions.

As hinted earlier, sometimes enlarging the class of hypotheses makes learning easier. For example, the table above shows that k-CNF is PAC learnable, but k-term-DNF is not. And yet, k-term-DNF is a subclass of k-CNF! So, even if the target function were in k-term-DNF, one would be able to find a hypothesis in k-CNF that is probably approximately correct for the target function. Similarly, linearly separable functions implemented by TLUs whose weight values are restricted to 0 and 1 are not properly PAC learnable, whereas unrestricted linearly separable functions are. It is possible that enlarging the space of hypotheses makes finding one that is consistent with the training examples easier. An interesting question is whether or not the class of functions in k-2NN is polynomially PAC

learnable if the hypotheses are drawn from k'-2NN with $k' > k$. (At the time of writing, this matter is still undecided.)

Although PAC learning theory is a powerful analytic tool, it (like complexity theory) deals mainly with worst-case results. The fact that the class of two-layer, feedforward neural networks is not polynomially PAC learnable is more an attack on the theory than it is on the networks, which have had many successful applications. As [Baum, 1994, page 416-17] says: " ... humans are capable of learning in the natural world. Therefore, a proof within some model of learning that learning is not feasible is an indictment of the model. We should examine the model to see what constraints can be relaxed and made more realistic."

8.3 The Vapnik-Chervonenkis Dimension

8.3.1 Linear Dichotomies

Consider a set, $\mathcal{H}$, of functions, and a set, Ξ, of (unlabeled) patterns. One measure of the expressive power of a set of hypotheses, relative to Ξ, is its ability to make *arbitrary* classifications of the patterns in Ξ.[1] If there are m patterns in Ξ, there are 2^m different ways to divide these patterns into two disjoint and exhaustive subsets. We say there are 2^m different *dichotomies* of Ξ. If Ξ were to include *all* of the 2^n Boolean patterns, for example, there are 2^{2^n} ways to dichotomize them, and (of course) the set of all possible Boolean functions dichotomizes them in all of these ways. But a subset, $\mathcal{H}$, of the Boolean functions might not be able to dichotomize an arbitrary set, Ξ, of m Boolean patterns in all 2^m ways. In general (that is, even in the non-Boolean case), we say that if a subset, $\mathcal{H}$, of functions can dichotomize a set, Ξ, of m patterns in all 2^m ways, then $\mathcal{H}$ *shatters* Ξ.

As an example, consider a set Ξ of m patterns in the n-dimensional space, $\mathcal{R}^n$. (That is, the n components of these patterns are real numbers.) We define a *linear dichotomy* as one implemented by an $(n-1)$-dimensional hyperplane in the n-dimensional space. How many linear dichotomies of m patterns in n dimensions are there? For example, as shown in Fig. 8.1, there are 14 dichotomies of four points in two dimensions (each separating line yields two dichotomies depending on whether the points on one side of the line are classified as 1 or 0). (Note that even though there are an infinite number of hyperplanes, there are, nevertheless, only a finite number

[1] And, of course, if a hypothesis drawn from a set that could make arbitrary classifications of a set of training patterns, there is little likelihood that such a hypothesis will generalize well beyond the training set.

of ways in which hyperplanes can dichotomize a finite number of patterns. Small movements of a hyperplane typically do not change the classifications of any patterns.)

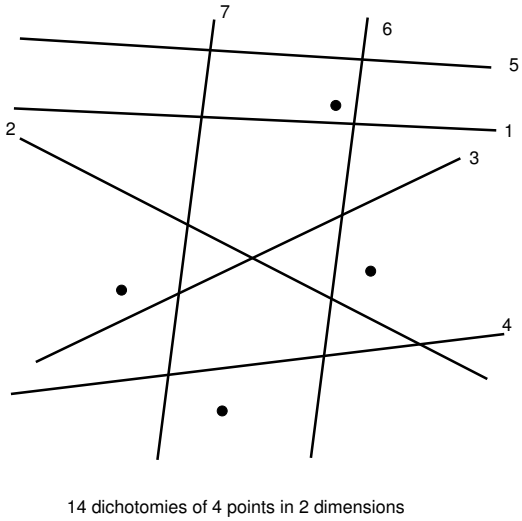

Figure 8.1: Dichotomizing Points in Two Dimensions

The number of dichotomies achievable by hyperplanes depends on how the patterns are disposed. For the maximum number of linear dichotomies, the points must be in what is called *general position*. For $m > n$, we say that a set of m points is in *general position* in an n-dimensional space if and only if no subset of $(n+1)$ points lies on an $(n-1)$-dimensional hyperplane. When $m \leq n$, a set of m points is in general position if no $(m-2)$-dimensional hyperplane contains the set. Thus, for example, a set of $m \geq 4$ points is in general position in a three-dimensional space if no four of them lie on a (two-dimensional) plane. We will denote the number of linear dichotomies of m points in general position in an n-dimensional space by the expression $\Pi_L(m,n)$.

It is not too difficult to verify that:

Include the derivation.

$$\Pi_L(m,n) = 2\sum_{i=0}^{n} C(m-1,i) \qquad \text{for } m > n, \text{and}$$

$$= 2^m \qquad \text{for } m \leq n$$

where $C(m-1,i)$ is the binomial coefficient $\frac{(m-1)!}{(m-1-i)!i!}$.

The table below shows some values for $\Pi_L(m,n)$.

m (no. of patterns)	n (dimension)				
	1	2	3	4	5
1	2	2	2	2	2
2	**4**	4	4	4	4
3	6	**8**	8	8	8
4	8	14	**16**	16	16
5	10	22	30	**32**	32
6	12	32	52	62	**64**
7	14	44	84	114	126
8	16	58	128	198	240

Note that the class of linear dichotomies shatters the m patterns if $m \leq n+1$. The bold-face entries in the table correspond to the highest values of m for which linear dichotomies shatter m patterns in n dimensions.

8.3.2 Capacity

Let $P_{m,n} = \frac{\Pi_L(m,n)}{2^m}$ = the probability that a randomly selected dichotomy (out of the 2^m possible dichotomies of m patterns in n dimensions) will be linearly separable. In Fig. 8.2 we plot $P_{\lambda(n+1),n}$ versus λ and n, where $\lambda = m/(n+1)$.

Note that for large n (say $n > 30$) how quickly $P_{m,n}$ falls from 1 to 0 as m goes above $2(n+1)$. For $m < 2(n+1)$, *any* dichotomy of the m points is almost certainly linearly separable. But for $m > 2(n+1)$, a randomly selected dichotomy of the m points is almost certainly not linearly separable. For this reason $m = 2(n+1)$ is called the *capacity* of a TLU [Cover, 1965]. Unless the number of training patterns exceeds the capacity, the fact that a TLU separates those training patterns according to their labels means nothing in terms of how well that TLU will generalize to new patterns. There is nothing special about a separation found for $m < 2(n+1)$ patterns—almost *any* dichotomy of those patterns would have been linearly separable. To make sure that the separation found is forced by the training set and thus generalizes well, it has to be the case that there are very few linearly separable functions that would separate the m training patterns.

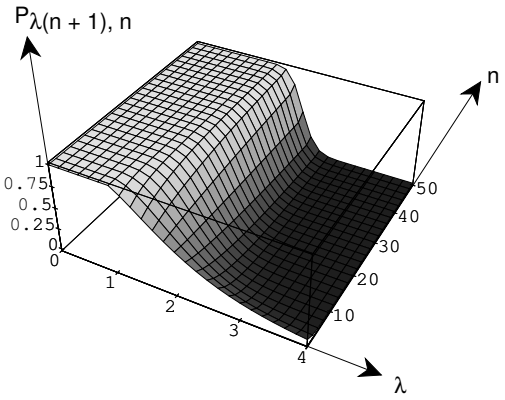

Figure 8.2: Probability that a Random Dichotomy is Linearly Separable

Analogous results about the generalizing abilities of neural networks have been developed by [Baum & Haussler, 1989] and given intuitive and experimental justification in [Baum, 1994, page 438]:

> "The results seemed to indicate the following heuristic rule holds. If M examples [can be correctly classified by] a net with W weights (for $M >> W$), the net will make a fraction ε of errors on new examples chosen from the same [uniform] distribution where $\varepsilon = W/M$."

8.3.3 A More General Capacity Result

Corollary 7.2 gave us an expression for the number of training patterns sufficient to guarantee a required level of generalization—assuming that the function we were guessing was a function belonging to a class of known and finite cardinality. The capacity result just presented applies to linearly separable functions for non-binary patterns. We can extend these ideas to general dichotomies of non-binary patterns.

In general, let us denote the maximum number of dichotomies of *any* set of m n-dimensional patterns by hypotheses in $\mathcal{H}$ as $\Pi_{\mathcal{H}}(m, n)$. The number of dichotomies will, of course, depend on the disposition of the m points

in the n-dimensional space; we take $\Pi_{\mathcal{H}}(m, n)$ to be the maximum over all possible arrangements of the m points. (In the case of the class of linearly separable functions, the maximum number is achieved when the m points are in general position.) For each class, $\mathcal{H}$, there will be some maximum value of m for which $\Pi_{\mathcal{H}}(m, n) = 2^m$, that is, for which $\mathcal{H}$ shatters the m patterns. This maximum number is called the *Vapnik-Chervonenkis (VC) dimension* and is denoted by VCdim($\mathcal{H}$) [Vapnik & Chervonenkis, 1971].

We saw that for the class of linear dichotomies, VCdim($Linear$) = (n + 1). As another example, let us calculate the VC dimension of the hypothesis space of single intervals on the real line—used to classify points on the real line. We show an example of how points on the line might be dichotomized by a single interval in Fig. 8.3. The set Ξ could be, for example, {0.5, 2.5, - 2.3, 3.14}, and one of the hypotheses in our set would be [1, 4.5]. This hypothesis would label the points 2.5 and 3.14 with a 1 and the points - 2.3 and 0.5 with a 0. This set of hypotheses (single intervals on the real line) can arbitrarily classify any two points. But no single interval can classify three points such that the outer two are classified as 1 and the inner one as 0. Therefore the VC dimension of single intervals on the real line is 2. As soon as we have many more than 2 training patterns on the real line and provided we know that the classification function we are trying to guess is a single interval, then we begin to have good generalization.

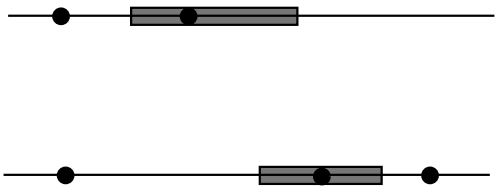

Figure 8.3: Dichotomizing Points by an Interval

The VC dimension is a useful measure of the expressive power of a hypothesis set. Since *any* dichotomy of VCdim($\mathcal{H}$) or fewer patterns in general position in n dimensions can be achieved by *some* hypothesis in $\mathcal{H}$, we must have many more than VCdim($\mathcal{H}$) patterns in the training set in order that a hypothesis consistent with the training set is sufficiently constrained to imply good generalization. Our examples have shown that the concept of VC dimension is not restricted to Boolean functions.

8.3.4 Some Facts and Speculations About the VC Dimension

- If there are a finite number, $|\mathcal{H}|$, of hypotheses in $\mathcal{H}$, then:
  $\text{VCdim}(\mathcal{H}) \leq \log(|\mathcal{H}|)$
- The VC dimension of terms in n dimensions is n.
- Suppose we generalize our example that used a hypothesis set of single intervals on the real line. Now let us consider an n-dimensional feature space and tests of the form $L_i \leq x_i \leq H_i$. We allow only one such test per dimension. A hypothesis space consisting of conjunctions of these tests (called *axis-parallel hyper-rectangles*) has VC dimension bounded by:
  $n \leq \text{ VCdim } \leq 2n$
- As we have already seen, TLUs with n inputs have a VC dimension of $n + 1$.
- [Baum, 1994, page 438] gives experimental evidence for the proposition that " ... multilayer [neural] nets have a VC dimension roughly equal to their total number of [adjustable] weights."

8.4 VC Dimension and PAC Learning

There are two theorems that connect the idea of VC dimension with PAC learning [Blumer, *et al.*, 1990]. We state these here without proof.

Theorem 8.3 (Blumer, *et al.***)** *A hypothesis space $\mathcal{H}$ is PAC learnable iff it has finite VC dimension.*

Theorem 8.4 *A set of hypotheses, $\mathcal{H}$, is properly PAC learnable if:*

1. $m \geq (1/\varepsilon)\max[4\lg(2/\delta),\ 8\ \text{VCdim}\ \lg(13/\varepsilon)]$, *and*
2. *if there is an algorithm that outputs a hypothesis $h \in \mathcal{H}$ consistent with the training set in polynomial (in m and n) time.*

The second of these two theorems improves the bound on the number of training patterns needed for linearly separable functions to one that is linear in n. In our previous example of how many training patterns were needed to ensure PAC learnability of a linearly separable function if $n = 50$,

$\varepsilon = 0.01$, and $\delta = 0.01$, we obtained $m \geq 173,748$. Using the Blumer, *et al.* result we would get $m \geq 52,756$.

As another example of the second theorem, let us take $\mathcal{H}$ to be the set of closed intervals on the real line. The VC dimension is 2 (as shown previously). With $n = 50$, $\varepsilon = 0.01$, and $\delta = 0.01$, $m \geq 16,551$ ensures PAC learnability.

There is also a theorem that gives a lower (necessary) bound on the number of training patterns required for PAC learning [Ehrenfeucht, *et al.*, 1988]:

Theorem 8.5 *Any PAC learning algorithm must examine at least* $\Omega(1/\varepsilon \lg(1/\delta) + \text{VCdim}(\mathcal{H}))$ *training patterns.*

The difference between the lower and upper bounds is $O(\log(1/\varepsilon)\text{VCdim}(\mathcal{H})/\varepsilon)$.

8.5 Bibliographical and Historical Remarks

To be added.