Machine Learning

Chapter 07

Computational Learning Theory

INTRODUCTION

- When studying machine learning it is natural to wonder what general laws may govern machine (and non-machine) learners.
- Is it possible to identify classes of learning problems that are inherently difficult or easy, independent of the learning algorithm?
- Can one characterize the number of training examples necessary or sufficient to assure successful learning?
- How is this number affected if the learner is allowed to pose queries to the trainer, versus observing a random sample of training examples?
- Can one characterize the number of mistakes that a learner will make before learning the target function?
- Can one characterize the inherent computational complexity of classes of learning problems?
- Although general answers to all these questions are not yet known, fragments of a computational theory of learning have begun to emerge.

- This Computational Theory presents key results from this theory, providing answers to these questions within particular problem settings.
- We focus here on the problem of inductively learning an unknown target function, given only training examples of this target function and a space of candidate hypotheses.
- Within this setting, we will be chiefly concerned with questions such as how many training examples are sufficient to successfully learn the target function, and how many mistakes will the learner make before succeeding.
- As we shall see, it is possible to set quantitative bounds on these measures, depending on attributes of the learning problem such as:
  - The size or complexity of the hypothesis space considered by the learner
  - The accuracy to which the target concept must be approximated
  - The probability that the learner will output a successful hypothesis
  - The manner in which training examples are presented to the learner

- Our goal is to answer questions such as:
  - Sample complexity. How many training examples are needed for a learner to converge (with high probability) to a successful hypothesis?
  - Computational complexity. How much computational effort is needed for a learner to converge (with high probability) to a successful hypothesis?
  - Mistake bound. How many training examples will the learner misclassify before converging to a successful hypothesis?
- For example, there are various ways to specify what it means for the learner to be "successful."
- We might specify that to succeed, the learner must output a hypothesis identical to the target concept
- Alternatively, we might simply require that it output a hypothesis that agrees with the target concept most of the time, or that it usually output such a hypothesis.
- Similarly, we must specify how training examples are to be obtained by the learner

PROBABLY LEARNING AN APPROXIMATELY CORRECT HYPOTHESIS

- Consider a particular setting for the learning problem, called the probably approximately correct (PAC) learning model.
- We begin by specifying the problem setting that defines the PAC learning model, then consider the questions of how many training examples and how much computation are required in order to learn various classes of target functions within this PAC model.
- **The Problem Setting**
- let X refer to the set of all possible instances over which target functions may be defined. For example, X might represent the set of all people, each described by the attributes age (e.g., young or old) and height (short or tall).
- Let C refer to some set of target concepts that our learner might be called upon to learn. Each target concept c in C corresponds to some subset of X, or equivalently to some boolean-valued function c : X -> {0, 1).
- For example, one target concept c in C might be the concept "people who are skiers." If x is a positive example of c, then we will write c(x) = 1; if x is a negative example, c(x) = 0.

- We assume instances are generated at random from X according to some probability distribution D.
- For example, D might be the distribution of instances generated by observing people who walk out of the largest sports store in Switzerland.
- In general, D may be any distribution, and it will not generally be known to the learner.
- Training examples are generated by drawing an instance x at random according to D, then presenting x along with its target value, c(x), to the learner.
- The learner L considers some set H of possible hypotheses when attempting to learn the target concept
- After observing a sequence of training examples of the target concept c, L must output some hypothesis h from H, which is its estimate of c.
- To be fair, we evaluate the success of L by the performance of h over new instances drawn randomly from X according to D, the same probability distribution used to generate the training data

Error of a Hypothesis

- Because we are interested in how closely the learner's output hypothesis h approximates the actual target concept c, let us begin by defining the true error of a hypothesis h with respect to target concept c and instance distribution D.
- Informally, the true error of h is just the error rate we expect when applying h to future instances drawn according to the probability distribution D
- Definition: The true error (denoted errorD(h)) of hypothesis h with respect to target concept c and distribution D is the probability that h will misclassify an instance drawn at random according to D.

$$error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[c(x) \neq h(x)]$$

Here the notation $\Pr_{x \in \mathcal{D}}$ indicates that the probability is taken over the instance distribution $\mathcal{D}$.

- Figure 7.1 shows this definition of error in graphical form. The concepts c and h are depicted by the sets of instances within X that they label as positive.
- The error of h with respect to c is the probability that a randomly drawn instance will fall into the region where h and c disagree (i.e., their set difference).
- Note we have chosen to define error over the entire distribution of instances-not simply over the training examples-because this is the true error we expect to encounter when actually using the learned hypothesis h on subsequent instances drawn from D
- Note that error depends strongly on the unknown probability distribution D.
- For example, if D is a uniform probability distribution that assigns the same probability to every instance in X, then the error for the hypothesis in Figure 7.1 will be the fraction of the total instance space that falls into the region where h and c disagree
- In the extreme, if D happens to assign zero probability to the instances for which h(x) = c(x), then the error for the h in Figure 7.1 will be 1, despite the fact the h and c agree on a very large number of (zero probability) instances.
- Finally, note that the error of h with respect to c is not directly observable to the learner. L can only observe the performance of h over the training examples, and it must choose its output hypothesis on this basis only.
- We will use the term training error to refer to the fraction of training examples misclassified by h, in contrast to the true error defined above. Much of our analysis of the complexity of learning centers around the question "how probable is it that the observed training error for h gives a misleading estimate of the true errorD(h)?"

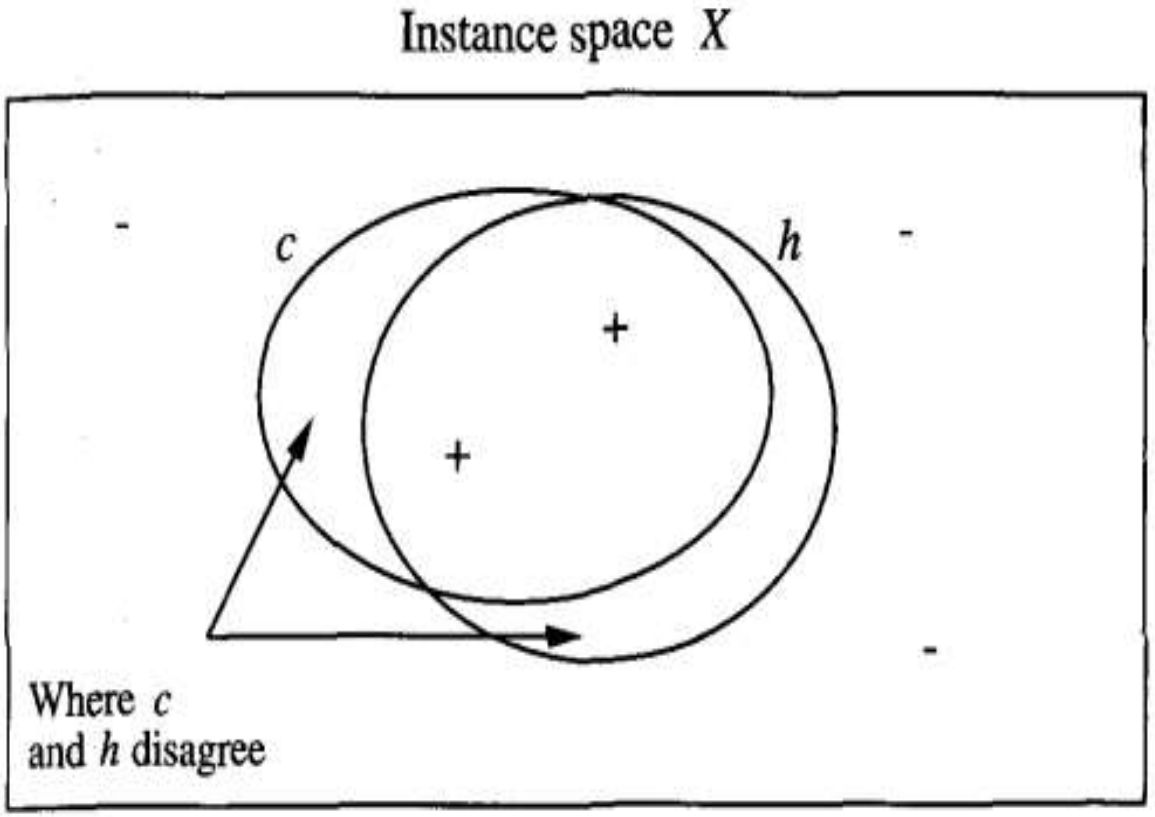


FIGURE 7.1

The error of hypothesis h with respect to target concept c. The error of h with respect to c is the probability that a randomly drawn instance will fall into the region where h and c disagree on its classification. The + and − points indicate positive and negative training examples. Note h has a nonzero error with respect to c despite the fact that h and c agree on all five training examples observed thus far.

PAC Learnability

- Our aim is to characterize classes of target concepts that can be reliably learned from a reasonable number of randomly drawn training examples and a reasonable amount of computation.
- We might try to characterize the number of training examples needed to learn a hypothesis h for which $error_D(h) = 0$.
- Unfortunately, it turns out this is futile in the setting we are considering, for two reasons. First, unless we provide training examples corresponding to every possible instance in X (an unrealistic assumption), there may be multiple hypotheses consistent with the provided training examples, and the learner cannot be certain to pick the one corresponding to the target concept. Second, given that the training examples are drawn randomly, there will always be some nonzero probability that the training examples encountered by the learner will be misleading.
- To accommodate these two difficulties, we weaken our demands on the learner in two ways. First, we will not require that the learner output a zero error hypothesis-we will require only that its error be bounded by some constant, c, that can be made arbitrarily small. Second, we will not require that the learner succeed for every sequence of randomly drawn training examples-we will require only that its probability of failure be bounded by some constant, 6, that can be made arbitrarily small. In short, we require only that the learner probably learn a hypothesis that is approximately correct-hence the term probably approximately correct learning, or PAC learning for short.

- Consider some class C of possible target concepts and a learner L using hypothesis space H.
- we will say that the concept class C is PAC-learnable by L using H if, for any target concept c in C, L will with probability (1 - δ) output a hypothesis h with $error_D(h) < \epsilon$, after observing a reasonable number of training examples and performing a reasonable amount of computation

Definition: Consider a concept class C defined over a set of instances X of length n and a learner L using hypothesis space H. C is **PAC-learnable** by L using H if for all $c \in C$, distributions $\mathcal{D}$ over X, ϵ such that $0 < \epsilon < 1/2$, and δ such that $0 < \delta < 1/2$, learner L will with probability at least $(1 - \delta)$ output a hypothesis $h \in H$ such that $error_{\mathcal{D}}(h) \leq \epsilon$, in time that is polynomial in $1/\epsilon$, $1/\delta$, n, and $size(c)$.

- Our definition requires two things from L. First, L must, with arbitrarily high probability (1 - δ), output a hypothesis having arbitrarily low error (ϵ). Second, it must do so efficiently-in time that grows at most polynomially with 1/ϵ and 1/δ which define the strength of our demands on the output hypothesis, and with n and size(c) that define the inherent complexity of the underlying instance space X and concept class C. Here, n is the size of instances in X.

SAMPLE COMPLEXITY FOR FINITE HYPOTHESIS SPACES

- PAC-learnability is largely determined by the number of training examples required by the learner. The growth in the number of required training examples with problem size, called the sample complexity of the learning problem
- The reason is that in most practical settings the factor that most limits success of the learner is the limited availability of training data.
- Here we present a general bound on the sample complexity for a very broad class of learners, called consistent learners. A learner is consistent if it outputs hypotheses that perfectly fit the training data, whenever possible.
- It is quite reasonable to ask that a learning algorithm be consistent, given that we typically prefer a hypothesis that fits the training data over one that does not.
- Can we derive a bound on the number of training examples required by any consistent learner, independent of the specific algorithm it uses to derive a consistent hypothesis? The answer is yes
- The version space, $VS_{H,D}$, to be the set of all hypotheses h ϵ H that correctly classify the training examples D.

$$VS_{H,D} = \{h \in H | (\forall \langle x, c(x) \rangle \in D)\ (h(x) = c(x))\}$$

- The significance of the version space here is that every consistent learner outputs a hypothesis belonging to the version space, regardless of the instance space X, hypothesis space H, or training data D.
- The reason is simply that by definition the version space $VS_{H,D}$ contains every consistent hypothesis in H.
- Therefore, to bound the number of examples needed by any consistent learner, we need only bound the number of examples needed to assure that the version space contains no unacceptable hypotheses

Definition: Consider a hypothesis space H, target concept c, instance distribution $\mathcal{D}$, and set of training examples D of c. The version space $VS_{H,D}$ is said to be ϵ-**exhausted** with respect to c and $\mathcal{D}$, if every hypothesis h in $VS_{H,D}$ has error less than ϵ with respect to c and $\mathcal{D}$.

$$(\forall h \in VS_{H,D})\ error_{\mathcal{D}}(h) < \epsilon$$

- This definition is illustrated in Figure 7.2. The version space is ϵ-exhausted just in the case that all the hypotheses consistent with the observed training examples (i.e., those with zero training error) happen to have true error less than E

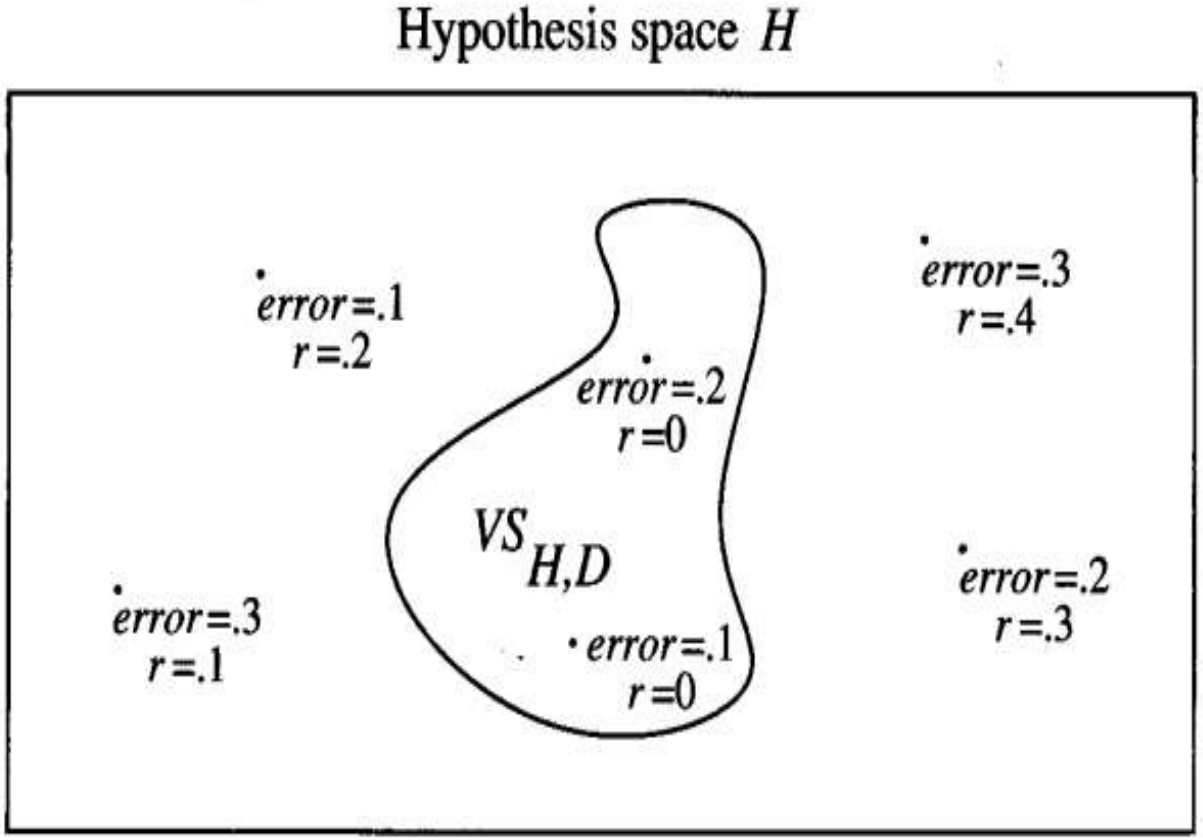


FIGURE 7.2
Exhausting the version space. The version space $VS_{H,D}$ is the subset of hypotheses $h \in H$, which have zero training error (denoted by $r = 0$ in the figure). Of course the true $error_{\mathcal{D}}(h)$ (denoted by *error* in the figure) may be nonzero, even for hypotheses that commit zero errors over the training data. The version space is said to be ϵ-exhausted when all hypotheses h remaining in $VS_{H,D}$ have $error_{\mathcal{D}}(h) < \epsilon$.

Theorem 7.1. **ϵ-exhausting the version space.** If the hypothesis space H is finite, and D is a sequence of $m \geq 1$ independent randomly drawn examples of some target concept c, then for any $0 \leq \epsilon \leq 1$, the probability that the version space $VS_{H,D}$ is not ϵ-exhausted (with respect to c) is less than or equal to

$$|H|e^{-\epsilon m}$$

Proof. Let hl, h2, . . . hk be all the hypotheses in H that have true error greater than E with respect to c. We fail to ϵ-exhaust the version space if and only if at least one of these k hypotheses happens to be consistent with all m independent random training examples. The probability that any single hypothesis having true error greater than E would be consistent with one randomly drawn example is at most (1 - ϵ). Therefore the probability that this hypothesis will be consistent with m independently drawn examples is at most (1 - ϵ)^m. Given that we have k hypotheses with error greater than ϵ, the probability that at least one of these will be consistent with all m training examples is at most

$$k(1-\epsilon)^m$$

And since $k \leq |H|$, this is at most $|H|(1-\epsilon)^m$. Finally, we use a general inequality stating that if $0 \leq \epsilon \leq 1$ then $(1-\epsilon) \leq e^{-\epsilon}$. Thus,

$$k(1-\epsilon)^m \leq |H|(1-\epsilon)^m \leq |H|e^{-\epsilon m}$$

which proves the theorem.

- We have just proved an upper bound on the probability that the version space is not ϵ-exhausted, based on the number of training examples m, the allowed error ϵ, and the size of H.
- Put another way, this bounds the probability that m training examples will fail to eliminate all "bad" hypotheses (i.e., hypotheses with true error greater than ϵ), for any consistent learner using hypothesis space H.
- Let us use this result to determine the number of training examples required to reduce this probability of failure below some desired level δ.

$$|H|e^{-\epsilon m} \leq \delta \tag{7.1}$$

Rearranging terms to solve for m, we find

$$m \geq \frac{1}{\epsilon}(\ln|H| + \ln(1/\delta)) \tag{7.2}$$

- To summarize, the inequality shown in Equation (7.2) provides a general bound on the number of training examples sufficient for any consistent learner to successfully learn any target concept in H, for any desired values of δ and ϵ.
- This number m of training examples is sufficient to assure that any consistent hypothesis will be probably (with probability $(1 - \delta)$) approximately (within error ϵ) correct. Notice m grows linearly in $1/\epsilon$ and logarithmically in $1/\delta$.
- It also grows logarithmically in the size of the hypothesis space H.
- Note that the above bound can be a substantial overestimate.
- For example, although the probability of failing to exhaust the version space must lie in the interval [0, 1], the bound given by the theorem grows linearly with IHI.

SAMPLE COMPLEXITY FOR INFINITE HYPOTHESIS SPACES

- We showed that sample complexity for PAC learning grows as the logarithm of the size of the hypothesis space.
- While Equation (7.2) is quite useful, there are two drawbacks to characterizing sample complexity in terms of |H|.
- First, it can lead to quite weak bounds (recall that the bound on δ can be significantly greater than 1 for large |H|). Second, in the case of infinite hypothesis spaces we cannot apply Equation (7.2) at all!
- Here we consider a second measure of the complexity of H, called the Vapnik-Chervonenkis dimension of H (VC dimension, or VC(H), for short).
- As we shall see, we can state bounds on sample complexity that use VC(H) rather than IHI. In many cases, the sample complexity bounds based on VC(H) will be tighter than those from Equation (7.2)

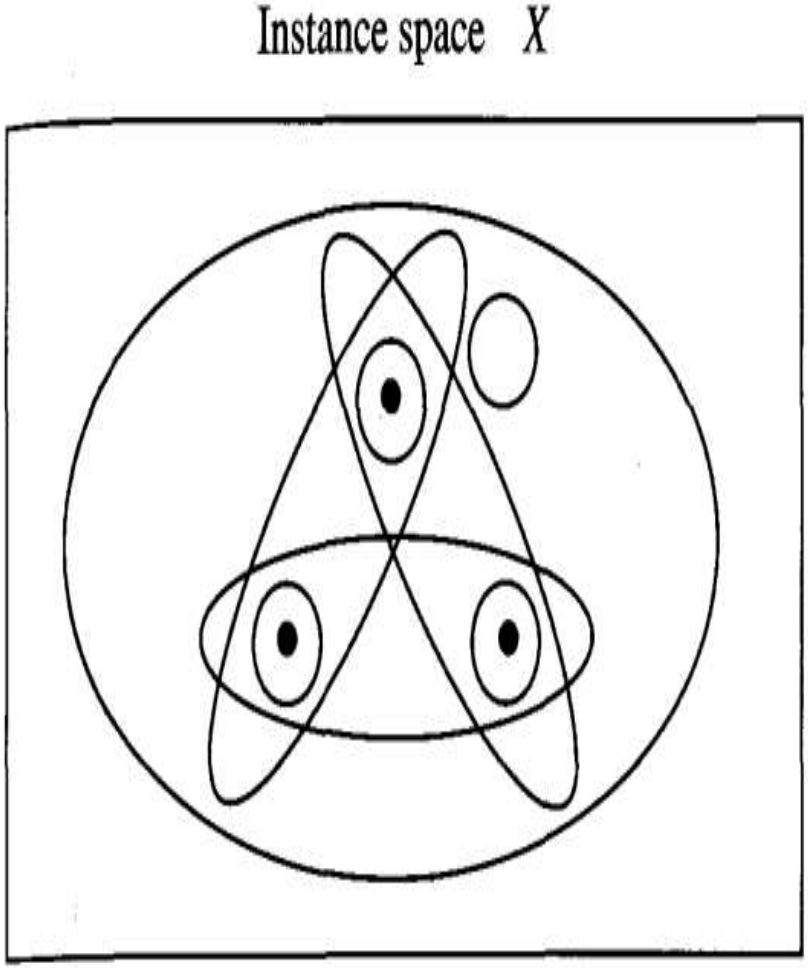
Shattering a Set of Instances

- The VC dimension measures the complexity of the hypothesis space H, not by the number of distinct hypotheses |H|, but instead by the number of distinct instances from X that can be completely discriminated using H.

To make this notion more precise, we first define the notion of *shattering* a set of instances. Consider some subset of instances $S \subseteq X$. For example, Figure 7.3 shows a subset of three instances from X. Each hypothesis h from H imposes some dichotomy on S; that is, h partitions S into the two subsets $\{x \in S | h(x) = 1\}$ and $\{x \in S | h(x) = 0\}$. Given some instance set S, there are $2^{|S|}$ possible dichotomies, though H may be unable to represent some of these. We say that H shatters S if every possible dichotomy of S can be represented by some hypothesis from H.

Definition: A set of instances S is **shattered** by hypothesis space H if and only if for every dichotomy of S there exists some hypothesis in H consistent with this dichotomy.

Figure 7.3 illustrates a set S of three instances that is shattered by the hypothesis space. Notice that each of the 2^3 dichotomies of these three instances is covered by some hypothesis.



Note that if a set of instances is not shattered by a hypothesis space, then there must be some concept (dichotomy) that can be defined over the instances, but that cannot be represented by the hypothesis space. The ability of H to shatter a set of instances is thus a measure of its capacity to represent target concepts defined over these instances.

FIGURE 7.3

A set of three instances shattered by eight hypotheses. For every possible dichotomy of the instances, there exists a corresponding hypothesis.

The Vapnik-Chervonenkis Dimension

- The ability to shatter a set of instances is closely related to the inductive bias of a hypothesis space .
- An unbiased hypothesis space is one capable of representing every possible concept (dichotomy) definable over the instance space X. Put briefly, an unbiased hypothesis space H is one that shatters the instance space X. What if H cannot shatter X, but can shatter some large subset S of X? Intuitively, it seems reasonable to say that the larger the subset of X that can be shattered, the more expressive H. The VC dimension of H is precisely this measure.

Definition: The **Vapnik-Chervonenkis dimension**, $VC(H)$, of hypothesis space H defined over instance space X is the size of the largest finite subset of X shattered by H. If arbitrarily large finite sets of X can be shattered by H, then $VC(H) \equiv \infty$.

Note that for any finite H, $VC(H) \leq \log_2 |H|$. To see this, suppose that $VC(H) = d$. Then H will require 2^d distinct hypotheses to shatter d instances. Hence, $2^d \leq |H|$, and $d = VC(H) \leq \log_2 |H|$.

ILLUSTRATIVE EXAMPLES

- In order to develop an intuitive feeling for VC(H), consider a few example hypothesis spaces.
- To get started, suppose the instance space X is the set of real numbers X = R (e.g., describing the height of people), and H the set of intervals on the real number line.
- In other words, H is the set of hypotheses of the form $a < x < b$, where a and b may be any real constants.
- What is VC(H)? To answer this question, we must find the largest subset of X that can be shattered by H.
- Consider a particular subset containing two distinct instances, say S = {3.1,5.7}.
- Can S be shattered by H? Yes. For example, the four hypotheses $(1 < x < 2)$, $(1 < x < 4)$, $(4 < x < 7)$, and $(1 < x < 7)$ will do.
- Together, they represent each of the four dichotomies over S, covering neither instance, either one of the instances, and both of the instances, respectively. Since we have found a set of size two that can be shattered by H, we know the VC dimension of H is at least two. Is there a set of size three that can be shattered?
- Consider a set S = (x0 x1, x2} containing three arbitrary instances. Without loss of generality, assume $x0 < x1 < x2$. Clearly this set cannot be shattered, because the dichotomy that includes x0 and x2, but not X1, cannot be represented by a single closed interval.

- Consider a set S = (x0, x1, x2} containing three. Therefore, no subset S of size three can be shattered, and VC(H) = 2. Note here that H is infinite, but VC(H) finite.
- The definition of VC dimension indicates that if we find any set of instances of size d that can be shattered, then VC(H) >=d. To show that VC(H) < d, we must show that no set of size d can be shattered. In this example, no sets of size four can be shattered, so VC(H) = 3. More generally, it can be shown that the VC dimension of linear decision surfaces in an r dimensional space (i.e., the VC dimension of a perceptron with r inputs) is r + 1.

FIGURE 7.4
The VC dimension for linear decision surfaces in the x, y plane is 3. (a) A set of three points that can be shattered using linear decision surfaces. (b) A set of three that cannot be shattered.

- As one final example, suppose each instance in X is described by the conjunction of exactly three boolean literals, and suppose that each hypothesis in H is described by the conjunction of up to three boolean literals. What is VC(H)? We can show that it is at least 3, as follows.
- Represent each instance by a 3-bit string corresponding to the values of each of its three literals l1, l2, and l3. Consider the following set of three instances:

$instance_1$: 100
$instance_2$: 010
$instance_3$: 001

- This set of three instances can be shattered by H, because a hypothesis can be constructed for any desired dichotomy as follows: If the dichotomy is to exclude $instance_i$, add the literal $\neg l_i$ to the hypothesis.

Sample Complexity and the VC Dimension

Earlier we considered the question "How many randomly drawn training examples suffice to probably approximately learn any target concept in C?" (i.e., how many examples suffice to ϵ-exhaust the version space with probability $(1 - \delta)$?). Using $VC(H)$ as a measure for the complexity of H, it is possible to derive an alternative answer to this question, analogous to the earlier bound of Equation (7.2). This new bound (see Blumer et al. 1989) is

$$m \geq \frac{1}{\epsilon}(4 \log_2(2/\delta) + 8VC(H) \log_2(13/\epsilon)) \tag{7.7}$$

Note that just as in the bound from Equation (7.2), the number of required training examples m grows logarithmically in $1/\delta$. It now grows log times linear in $1/\epsilon$, rather than linearly. Significantly, the $\ln|H|$ term in the earlier bound has now been replaced by the alternative measure of hypothesis space complexity, $VC(H)$ (recall $VC(H) \leq \log_2|H|$).

Theorem 7.3. Lower bound on sample complexity. Consider any concept class C such that $VC(C) \geq 2$, any learner L, and any $0 < \epsilon < \frac{1}{8}$, and $0 < \delta < \frac{1}{100}$. Then there exists a distribution $\mathcal{D}$ and target concept in C such that if L observes fewer examples than

$$\max\left[\frac{1}{\epsilon}\log(1/\delta), \frac{VC(C)-1}{32\epsilon}\right]$$

then with probability at least δ, L outputs a hypothesis h having $error_{\mathcal{D}}(h) > \epsilon$.

THE MISTAKE BOUND MODEL OF LEARNING

- While we have focused thus far on the PAC learning model, computational learning theory considers a variety of different settings and questions.
- Different learning settings that have been studied vary by how the training examples are generated (e.g., passive observation of random examples, active querying by the learner), noise in the data (e.g., noisy or error-free), the definition of success (e.g., the target concept must be learned exactly, or only probably and approximately), assumptions made by the learner (e.g., regarding the distribution of instances and whether C G H), and the measure according to which the learner is evaluated (e.g., number of training examples, number of mistakes, total time).
- we consider the mistake bound model of learning, in which the learner is evaluated by the total number of mistakes it makes before it converges to the correct hypothesis.As in the PAC setting, we assume the learner receives a sequence of training examples.
- However, here we demand that upon receiving each example x, the learner must predict the target value c(x), before it is shown the correct target value by the trainer.
- The question considered is "How many mistakes will the learner make in its predictions before it learns the target concept?' This question is significant in practical settings where learning must be done while the system is in actual use, rather than during some off-line training stage.

- For example, we might count the number of mistakes made before PAC learning the target concept. In the examples below, we consider instead the number of mistakes made before learning the target concept exactly. Learning the target concept exactly means converging to a hypothesis such that (For all(x)h(x) = c(x).

Mistake Bound for the FIND-S Algorithm

- To illustrate, consider again the hypothesis space H consisting of conjunctions of up to n boolean literals l1, .1n and their negations (e.g., Rich Λ ¬Handsome). Recall the FIND-S algorithm ,which incrementally computes the maximally specific hypothesis consistent with the training examples. A straightforward implementation of FIND-S for the hypothesis space H is as follows:

FIND-S:

- Initialize h to the most specific hypothesis $l_1 \wedge \neg l_1 \wedge l_2 \wedge \neg l_2 \ldots l_n \wedge \neg l_n$
- For each positive training instance x
  - Remove from h any literal that is not satisfied by x
- Output hypothesis h.

- FIND-S converges in the limit to a hypothesis that makes no errors, provided C subset of H and provided the training data is noise-free.
- FIND-S begins with the most specific hypothesis (which classifies every instance a negative example), then incrementally generalizes this hypothesis as needed to cover observed positive training examples.
- For the hypothesis representation used here, this generalization step consists of deleting unsatisfied literals.

Mistake Bound for the HALVING Algorithm

- As a second example, consider an algorithm that learns by maintaining a description of the version space, incrementally refining the version space as each new training example is encountered. The CANDIDATE-ELIMINATION algorithm and the LIST-THEN-ELIMINATE algorithm are examples of concept learning algorithms.
- In this section we derive a worst-case bound on the number of mistakes that will be made by such a learner, for any finite hypothesis space H, assuming again that the target concept must be learned exactly.

- To analyze the number of mistakes made while learning we must first specify precisely how the learner will make predictions given a new instance x.
- Let us assume this prediction is made by taking a majority vote among the hypotheses in the current version space. If the majority of version space hypotheses classify the new instance as positive, then this prediction is output by the learner. Otherwise a negative prediction is output.
- This combination of learning the version space, together with using a majority vote to make subsequent predictions, is often called the HALVING algorithm.
- What is the maximum number of mistakes that can be made by the HALVING algorithm, for an arbitrary finite H, before it exactly learns the target concept? Notice that learning the target concept "exactly" corresponds to reaching a state where the version space contains only a single hypothesis (as usual, we assume the target concept c is in H).
- To derive the mistake bound, note that the only time the HALVING algorithm can make a mistake is when the majority of hypotheses in its current version space incorrectly classify the new example. In this case, once the correct classification is revealed to the learner, the version space will be reduced to at most half its current size (i.e., only those hypotheses that voted with the minority will be retained).
- Given that each mistake reduces the size of the version space by at least half, and given that the initial version space contains only I H I members, the maximum number of mistakes possible before the version space contains just one member is log2 I H I. In fact one can show the bound is log2|H|. Consider, for example, the case in which IHI = 7. The first mistake must reduce IHI to at most 3, and the second mistake will then reduce it to 1.

- Optimal Mistake Bounds

The above analyses give worst-case mistake bounds for two specific algorithms: FIND-S and CANDIDATE-ELIMINATION. It is interesting to ask what is the optimal mistake bound for an arbitrary concept class C, assuming $H = C$. By optimal mistake bound we mean the lowest worst-case mistake bound over all possible learning algorithms. To be more precise, for any learning algorithm A and any target concept c, let $M_A(c)$ denote the maximum over all possible sequences of training examples of the number of mistakes made by A to exactly learn c. Now for any nonempty concept class C, let $M_A(C) \equiv \max_{c \in C} M_A(c)$. Note that above we showed $M_{Find-S}(C) = n + 1$ when C is the concept class described by up to n boolean literals. We also showed $M_{Halving}(C) \leq log_2(|C|)$ for any concept class C.

We define the optimal mistake bound for a concept class C below.

Definition: Let C be an arbitrary nonempty concept class. The **optimal mistake bound** for C, denoted $Opt(C)$, is the minimum over all possible learning algorithms A of $M_A(C)$.

$$Opt(C) \equiv \min_{A \in learning\ algorithms} M_A(C)$$

Speaking informally, this definition states that $Opt(C)$ is the number of mistakes made for the hardest target concept in C, using the hardest training sequence, by the best algorithm. Littlestone (1987) shows that for any concept class C, there is an interesting relationship among the optimal mistake bound for C, the bound of the HALVING algorithm, and the VC dimension of C, namely

$$VC(C) \leq Opt(C) \leq M_{Halving}(C) \leq log_2(|C|)$$

Furthermore, there exist concept classes for which the four quantities above are exactly equal. One such concept class is the powerset C_P of any finite set of instances X. In this case, $VC(C_P) = |X| = \log_2(|C_P|)$, so all four quantities must be equal. Littlestone (1987) provides examples of other concept classes for which $VC(C)$ is strictly less than $Opt(C)$ and for which $Opt(C)$ is strictly less than $M_{Halving}(C)$.

- **WEIGHTED-MAJORITY Algorithm**
- We consider a generalization of the HALVING algorithm called the WEIGHTED-MAJORITY algorithm.
- The WEIGHTED-MAJORITY algorithm makes predictions by taking a weighted vote among a pool of prediction algorithms and learns by altering the weight associated with each prediction algorithm.
- These prediction algorithms can be taken to be the alternative hypotheses in H, or they can be taken to be alternative learning algorithms that themselves vary over time.
- All that we require of a prediction algorithm is that it predict the value of the target concept, given an instance
- One interesting property of the WEIGHTED-MAJORITY algorithm is that it is able to accommodate inconsistent training data. This is because it does not eliminate a hypothesis that is found to be inconsistent with some training example, but rather reduces its weight.
- A second interesting property is that we can bound the number of mistakes made by WEIGHTED-MAJORITY in terms of the number of mistakes committed by the best of the pool of prediction algorithms.

- The WEIGHTED-MAJORITY algorithm begins by assigning a weight of 1 to each prediction algorithm, then considers the training examples. Whenever a prediction algorithm misclassifies a new training example its weight is decreased by multiplying it by some number B, where 0 <B < 1. The exact definition of the WEIGHTED-MAJORITY algorithm is given in Table 7.1.
- Notice if B = 0 then WEIGHTED-MAJORITY is identical to the HALVING algorithm. On the other hand, if we choose some other value for B, no prediction algorithm will ever be eliminated completely. If an algorithm misclassifies a training example, it will simply receive a smaller vote in the future. We now show that the number of mistakes committed by the WEIGHTEDMAJORITY algorithm can be bounded in terms of the number of mistakes made by the best prediction algorithm in the voting pool.

Theorem 7.5. Relative mistake bound for WEIGHTED-MAJORITY. Let D be any sequence of training examples, let A be any set of n prediction algorithms, and let k be the minimum number of mistakes made by any algorithm in A for the training sequence D. Then the number of mistakes over D made by the WEIGHTED-MAJORITY algorithm using $\beta = \frac{1}{2}$ is at most

$$2.4(k + \log_2 n)$$

a_i denotes the i^{th} prediction algorithm in the pool A of algorithms. w_i denotes the weight associated with a_i.

- For all i initialize $w_i \leftarrow 1$
- For each training example $\langle x, c(x) \rangle$
  - Initialize q_0 and q_1 to 0
  - For each prediction algorithm a_i
    - If $a_i(x) = 0$ then $q_0 \leftarrow q_0 + w_i$
      If $a_i(x) = 1$ then $q_1 \leftarrow q_1 + w_i$
  - If $q_1 > q_0$ then predict $c(x) = 1$
    If $q_0 > q_1$ then predict $c(x) = 0$
    If $q_1 = q_0$ then predict 0 or 1 at random for $c(x)$
  - For each prediction algorithm a_i in A do
    If $a_i(x) \neq c(x)$ then $w_i \leftarrow \beta w_i$

TABLE 7.1
WEIGHTED-MAJORITY algorithm.