UNIT-V

COMPUTATIONAL LEARNING THEORY

5.1 INTRODUCTION

Computational learning theory provides answers to the following questions within particular problem settings.

- Is it possible to identify learning problems as difficult or easy?
- Is it possible to identify learning problems that are independent of the learning algorithm?
- Number of training examples necessary or sufficient for 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?
- Number of mistakes that a learner will make before learning the target function?
- What is the Computational complexity of classes of learning problems?

5.2 PROBABLY LEARNING AN APPROXIMATELY CORRECT HYPOTHESIS

we consider a particular setting for the learning problem, called the probably approximately correct (PAC) learning model.

5.2.1 The Problem Setting:

Computational Learning Theory considers below problem setting to answer the listed questions.

Let X refer to the set of all possible instances.

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 $c : X \rightarrow \{0, 1\}$.

We assume instances are generated at random from X according to some probability distribution **D**.

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. Within this setting, we characterize the performance of various learners.

5.2.2 Error of a Hypothesis:

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

Figure 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*.

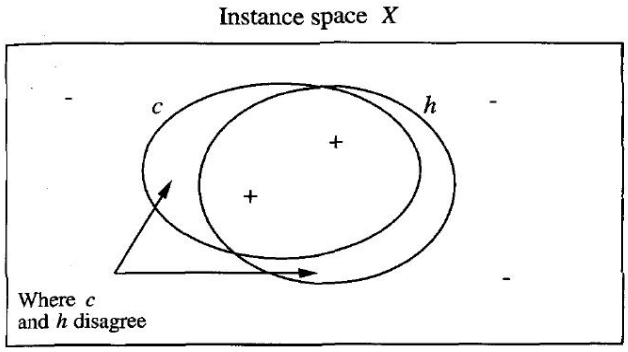


Figure:

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.

5.2.3 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.

- What kinds of statements about learnability should we guess hold true?
- We might try to characterize the number of training examples needed to learn a hypothesis h for which errorD(h) = 0.

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, , 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. Loosely speaking, 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 $error_D(h) < €$, after observing a reasonable number of training examples and performing a reasonable amount of computation. More precisely,

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 ***D*** over X, $€$ such that $0 < \delta < 1/2$, and δ such that $0 < \delta < 1/2$, learner L will with probability at least (1 - δ) output a hypothesis $h \in H$ such that $error_D(h)$ *5* $€$, in time that is polynomial in $1/€$, $1/\delta$, n, and size(c).

This definition requires two things from L.

i) L must, with arbitrarily high probability (1 - δ*)*, output a hypothesis having arbitrarily low error ***(€).***

ii)it must do so efficiently-in time that grows at most polynomially with $1/€$ and $1/\delta$, 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.

5.3 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, is the characteristic that is usually of greatest interest. 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.

The version space can derive a bound on the number of training examples required by ***any*** consistent learner, independent of the specific algorithm that generates consistent hypothesis.
The version space is defined as ***VSH,D,*** to be the set of all hypotheses ***h E H*** that correctly classify the training examples ***D.***

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

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

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

This definition is illustrated in below figure. The version space is €-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 € .

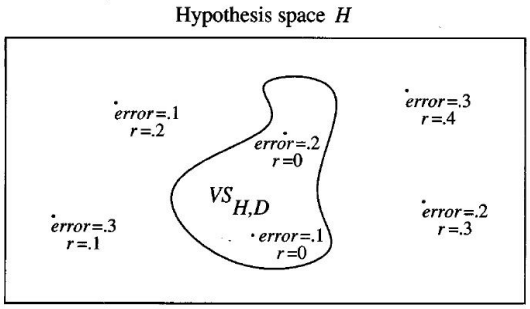


Figure :Exhausting the version space. The version space $VS_{H,D}$ is the subset of hypotheses $h \in H$, which have zero training error (denoted by $r = 0$ in the figure). Of course the true $error_D(h)$ (denoted by *error* in the figure) may be nonzero, even for hypotheses that commit zero errors over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

5.4 SAMPLE COMPLEXITY FOR INFINITE HYPOTHESIS SPACES

we consider a 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 |H|.

5.4.1 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.

First define the notion of shattering a set of instances. Consider some subset of instances S ⊆ X. For example, below figure 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 ∈ S / h(x) = 1} and {x ∈ 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 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.

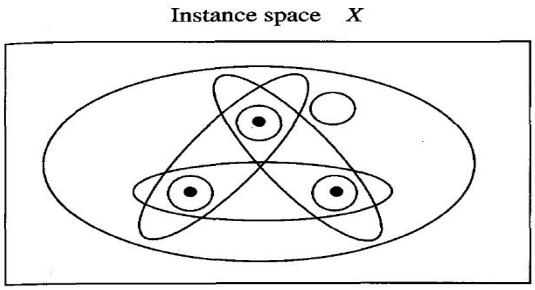


Figure :A set of three instances shattered by eight hypotheses. For every possible dichotomy of the instances, there exists a corresponding hypothesis.

7.4.2 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 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)* = ∞.

- For any finite H,VC(H)≤ log2 |H

Example1 for finding VC(H):

- Suppose the instance space X is the set of real numbers X = ℝ (e.g., describing the height of people), and H is the set of hypotheses of the form a < ***x*** < ***b,*** where a and b may be any real constants.
- To find VC(H), 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}. Here S is shattered by H.
- For example, consider the four hypotheses (1 < ***x*** < 2), (1 < ***x*** < 4), ***(4*** < ***x*** < 7), and (1 < ***x*** < 7). These hypotheses together 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, the VC dimension of H is at least two.
- There is no subset S of size three that can be shattered by H, and VC(H) = 2.
- Here H is infinite, but VC(H) finite.

Example2 for finding VC(H):

- Consider the set X of instances corresponding to points on the ***x,*** y plane
- Let H be the set of all linear decision surfaces in the plane.
- Now we will find the VC dimension of this H?
- Any two distinct points in the plane can be shattered by H, because we can find four linear surfaces that include neither, either, or both points.
- For a sets of three points, if the points are not colinear, then there will be 2^3 linear surfaces that shatter the points.
- VC(H) in this case is three as no sets of size four can be shattered.
- **The VC dimension of linear decision surfaces in an r dimensional space is r + 1.**

Example3 for finding VC(H):

- Let each instance in X is described by the conjunction of exactly three Boolean literals.
- Also let each hypothesis in H is described by the conjunction of up to three boolean literals.
- To find its VC(H), lets 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

instance2 :010

instance3 :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 instancei, add the literal ¬li to the hypothesis.
- For example, if we wish to include instance2, but exclude instance1 and instance3. Then we use the hypothesis ¬l1 ¬l3.
- The VC dimension for conjunctions of n boolean literals is at least n.

7.4.3 Sample Complexity and the VC Dimension

Upper bound on sample complexity:

Based on VC(H) as a measure for the complexity of H, the upper bound on the number of training examples sufficient to probably approximately learn any target concept in C, for any desired **ϵ** and δ is given below:

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

Lower bound on sample complexity: Consider any concept class ***C*** such that ***VC(C)*** ≥ 2, any learner L, and any *0* < ***ϵ*** < 1/8, and 0 < δ < 1/100. Then there exists a distribution 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 ***errorD(h)*** > ϵ.

5.5 THE MISTAKE BOUND MODEL OF LEARNING

In the mistake bound model of learning, the learner is evaluated by the total number of mistakes it makes before it converges to the correct hypothesis.

- Assume the learner receives a sequence of training examples. For each example x, the learner must predict the target value c(x), before it is shown the correct target value by the trainer.
- Here we consider the no. of mistakes that the learner make in its predictions before it learns the target concept.
- This mistake bound learning problem can be studied in various specific settings. For example, we might count the number of mistakes made before PAC learning the target concept.

5.5.1 Mistake Bound for the FIND-S Algorithm

- Let us see the bound on the total number of mistakes that FIND-S will make before exactly learning the target concept c.
- If c Є H (consisting of conjunction of n boolean literals), then FIND-S can never mistakenly classify a negative example as positive.
- To calculate the number of mistakes it will make, we need to count the number of mistakes it will make misclassifying truly positive examples as negative.
- How many such mistakes can occur before FIND-S learns c exactly?
- Consider the first positive example encountered by FIND-S. The learner will certainly make a mistake classifying this example, because its initial hypothesis labels every instance negative.
- The result will be that half of the $2n$ terms in its initial hypothesis will be eliminated, leaving only n terms.
- For each subsequent positive example that is mistakenly classified by the current hypothesis, at least one more of the remaining n terms must be eliminated from the hypothesis.

- Therefore, the total number of mistakes can be at most $n + 1$. This number of mistakes will be required in the worst case, corresponding to learning the most general possible target concept (∀x)c(x) = 1.

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⊆ 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.

5.5.2 Mistake Bound for the HALVING Algorithm

- Let us see the bound on the total number of mistakes that List-then-Eliminate and Candidate-elimination algorithm will make before exactly learning the target concept c.
- Let us assume this prediction is made by taking a majority vote among the hypotheses in the current version space. This combination of learning the version space, together with using a majority vote to make subsequent predictions, is called the HALVING algorithm.
- The HALVING algorithm can make a mistake when the majority of hypotheses in its current version space incorrectly classify the new example.
- Once the correct classification is revealed to the learner, the version space will be reduced to at most half its current size.
- Given that each mistake reduces the size of the version space by at least half, and given that the initial version space contains only |H| members, the maximum number of (worst case) mistakes possible before the version space contains just one member is $\log_2 |H|$.

5.5.3 Optimal Mistake Bounds

- Let us see optimal mistake bound, i.e. the lowest worst-case mistake bound over all possible learning algorithms.
- Assume 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.

- **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 $M_A(C)$.

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

- Opt(C) is the number of mistakes made for the hardest target concept in C, using the hardest training sequence, by the best algorithm.
- 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.

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

5.5.4 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.

- One of the property of Weighted-Majority algorithm 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.

- Another 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.
- Weighted-Majority algorithm is given below:

---

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 $w_i \leftarrow \beta w_i$

UNIT-V
SECTION-A

Objective Questions

1. Suppose you have a PAC learning algorithm A for a concept class C such that with probability at least 0.5, the algorithm will output an approximately correct hypothesis. Suppose that for deployment purposes, you need an algorithm which can output the approximately correct hypothesis with probability at least 0.998. Is it possible to make use of algorithm A for this purpose? If so, how? [ ]

A. No, we cannot make use of A
B. Yes, repeat A three times and choose the best hypothesis
C. Yes, repeat A _ve times and choose the best hypothesis
D. Yes, repeat A nine times and choose the best hypothesis

2. To say that the VC-dimension of a class is at least k, is it necessary for the class to be able to shatter any configuration of k points? [ ]
A. Yes B. No

3. What is the VC-dimension of the class of axis-parallel rectangles? [ ]
A. 3 B. 4 C. 5 D. 6

4. The VC dimension of hypothesis space H1 is larger than the VC dimension of hypothesis space H2. Which of the following can be inferred from this?[ ]
A. The number of examples required for learning a hypothesis in H1 is larger than the number of examples required for H2.
B. The number of examples required for learning a hypothesis in H1 is smaller than the number of examples required for H2.
C. The number of examples required for learning a hypothesis in H1 is equal to the number of examples required for H2.
D. No relation to number of samples required for PAC learning

5. For a particular learning task, if the requirement of error parameter □ changes from 0.1 to 0.01. How many more samples will be required for PAC learning? [ ]

A. same B. 2 times C. 10 times D. 100 times

6. Suppose the VC dimension of a hypothesis space is 4. Which of the following are true? [ ]
A. No sets of 4 points can be shattered by the hypothesis space.

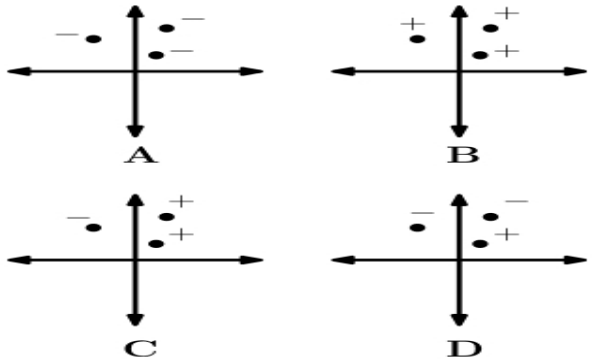
B. Atleast one set of 4 points can be shattered by the hypothesis space.
C. All sets of 4 points can be shattered by the hypothesis space.
D. No set of 5 points can be shattered by the hypothesis space.

7. Consider a circle in 2D whose center is at the origin. What is its VC dimension? [ ]

A. 1 B.2 C.3 D.4

8. Under a binary classification setting, which of the following sets of three labeled points cannot be shattered by a circle centered at the origin?

[ ]



A. A B. B C. C D. D

SECTION-B

Descriptive Questions

1. Define true error.
2. Describe these terms in brief (I) PAC Hypothesis (II) Mistake bound model of learning
3. Explain mistake bounds for Find-S algorithm and halving algorithm.
4. Describe the probability learning an approximately correct hypothesis.
5. Explain sample complexity for finite hypothesis space.
6. Explain sample complexity for infinite hypothesis spaces.
7. Describe optimal mistake bounds.
8. Explain the significance of VC Dimension measure with suitable example.

9.Discuss weighted majority algorithm.

10.How to calculate sample complexity for infinite hypothesis space?