PAC LEARNING

For the most part, we will focus not on individual learning algorithms, but rather on broad classes of learning algorithms characterized by the hypothesis spaces they consider, the presentation of training examples, etc. 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?

PROBABLY LEARNING AN APPROXIMATELY CORRECT HYPOTHESIS

- we consider a particular setting for the learning problem, called the probably approximately correct (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.
- For the sake of simplicity, we restrict the discussion to the case of learning boolean valued concepts from noise-free training data.

The Problem Setting

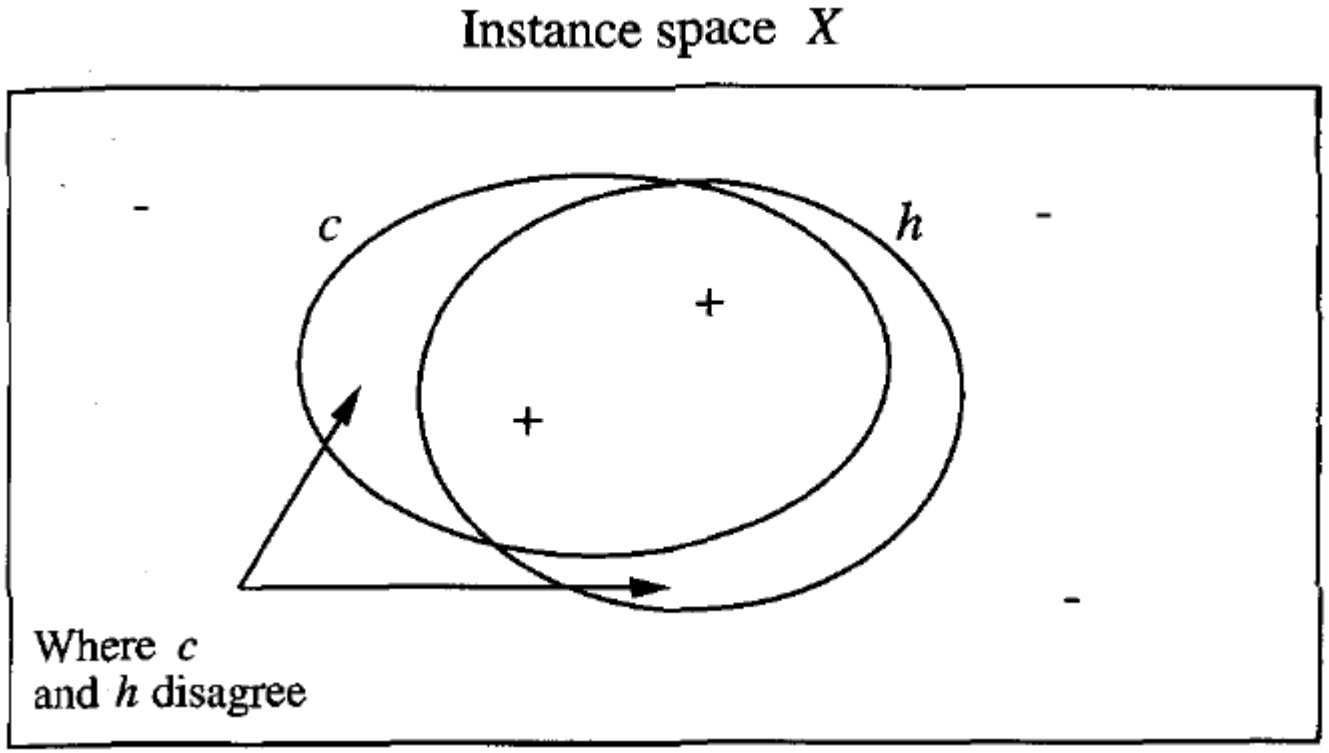
- As in earlier chapters, let X refer to the set of all possible instances over which target functions may be defined.
    - For example, X might represent the set of all people, each described by the attributes age (e.g., young or old) and height (short or tall).
- Let *C refer to some set of target 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.***
- The learner L considers some set H of possible hypotheses when attempting to learn the target concept.
    - For example, H might be the set of all hypotheses describable by conjunctions of the attributes ***age and height. 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.***
- Within this setting, we are interested in characterizing the performance of various learners L using various hypothesis spaces H, when learning individual target concepts drawn from various classes C.

Error of a Hypothesis

Definition: The **true error** (denoted $error_{\mathcal{D}}(h)$) of hypothesis h with respect to target concept c and distribution $\mathcal{D}$ is the probability that h will misclassify an instance drawn at random according to $\mathcal{D}$.

Here t tance
distribution



The error of hypothesis h with respect to target concept c.

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 $error_D(h) = 0$.
- 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, ϵ, 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, δ, 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.

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 $(1 - \delta)$, output a hypothesis having arbitrarily low error (ϵ). Second, it must do so efficiently—in time that grows at most polynomially with $1/\epsilon$ 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.

DEFINITION 2.1 (The Realizability Assumption) There exists $h^\star \in \mathcal{H}$ s.t. $L_{(\mathcal{D},f)}(h^\star) = 0$. Note that this assumption implies that with probability 1 over random samples, S, where the instances of S are sampled according to $\mathcal{D}$ and are labeled by f, we have $L_S(h^\star) = 0$.

The i.i.d. assumption: The examples in the training set are independently and identically distributed (i.i.d.) according to the distribution $\mathcal{D}$. That is, every x_i in S is freshly sampled according to $\mathcal{D}$ and then labeled according to the labeling function, f. We denote this assumption by $S \sim \mathcal{D}^m$ where m is the size of S, and $\mathcal{D}^m$ denotes the probability over m-tuples induced by applying $\mathcal{D}$ to pick each element of the tuple independently of the other members of the tuple.

PAC Learning

In the previous chapter we have shown that for a finite hypothesis class, if the ERM rule with respect to that class is applied on a sufficiently large training sample (whose size is independent of the underlying distribution or labeling function) then the output hypothesis will be probably approximately correct. More generally, we now define *Probably Approximately Correct* (PAC) learning.

Definition

DEFINITION 3.1 (PAC Learnability) A hypothesis class $\mathcal{H}$ is PAC learnable if there exist a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: For every $\epsilon, \delta \in (0,1)$, for every distribution $\mathcal{D}$ over $\mathcal{X}$, and for every labeling function $f : \mathcal{X} \to \{0,1\}$, if the realizable assumption holds with respect to $\mathcal{H}, \mathcal{D}, f$, then when running the learning algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. examples generated by $\mathcal{D}$ and labeled by f, the algorithm returns a hypothesis h such that, with probability of at least $1 - \delta$ (over the choice of the examples), $L_{(\mathcal{D},f)}(h) \leq \epsilon$.

The definition of Probably Approximately Correct learnability contains two approximation parameters. The accuracy parameter ϵ determines how far the output classifier can be from the optimal one (this corresponds to the "approximately correct"), and a confidence parameter δ indicating how likely the classifier is to meet that accuracy requirement (corresponds to the "probably" part of "PAC").

Sample Complexity

The function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ determines the *sample complexity* of learning $\mathcal{H}$: that is, how many examples are required to guarantee a probably approximately correct solution. The sample complexity is a function of the accuracy (ϵ) and confidence (δ) parameters. It also depends on properties of the hypothesis class $\mathcal{H}$ – for example, for a finite class we showed that the sample complexity depends on log the size of $\mathcal{H}$.

Every finite hypothesis class is PAC learnable with sample complexity

$$m_{\mathcal{H}}(\epsilon, \delta) \leq \left\lceil \frac{\log(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil .$$

Three Hypotheses of Interest

The **true function** c^* is the one we are trying to learn and that labeled the training data:

$$y^{(i)} = c^*(\mathbf{x}^{(i)}), \forall i \tag{1}$$

The **expected risk minimizer** has lowest true error:

$$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}} R(h) \tag{2}$$

The **empirical risk minimizer** has lowest training error:

$$\hat{h} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \hat{R}(h) \tag{3}$$

A More General Learning Model

The model we have just described can be readily generalized, so that it can be made relevant to a wider scope of learning tasks. We consider generalizations in two aspects:

Removing the Realizability Assumption
We have required that the learning algorithm succeeds on a pair of data distribution $\mathcal{D}$ and labeling function f provided that the realizability assumption is met. For practical learning tasks, this assumption may be too strong (can we really guarantee that there is a rectangle in the color-hardness space that *fully determines* which papayas are tasty?).

Learning Problems beyond Binary Classification

The learning task that we have been discussing so far has to do with predicting a binary label to a given example (like being tasty or not). However, many learning tasks take a different form. For example, one may wish to predict a real valued number (say, the temperature at 9:00 p.m. tomorrow) or a label picked from a finite set of labels (like the topic of the main story in tomorrow's paper). It turns out that our analysis of learning can be readily extended to such and many other scenarios by allowing a variety of loss functions.

Releasing the Realizability Assumption – Agnostic PAC Learning

A More Realistic Model for the Data-Generating Distribution

Recall that the realizability assumption requires that there exists $h^\star \in \mathcal{H}$ such that $\mathbb{P}_{x\sim\mathcal{D}}[h^\star(x) = f(x)] = 1$. In many practical problems this assumption does not hold. Furthermore, it is maybe more realistic not to assume that the labels are fully determined by the features we measure on input elements (in the case of the papayas, it is plausible that two papayas of the same color and softness will have different taste). In the following, we relax the realizability assumption by replacing the "target labeling function" with a more flexible notion, a data-labels generating distribution.

DEFINITION 3.3 (Agnostic PAC Learnability) A hypothesis class $\mathcal{H}$ is agnostic PAC learnable if there exist a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: For every $\epsilon, \delta \in (0,1)$ and for every distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, when running the learning algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. examples generated by $\mathcal{D}$, the algorithm returns a hypothesis h such that, with probability of at least $1 - \delta$ (over the choice of the m training examples),

$$L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon.$$

Realizable vs Agnostic

- **Definition**: A family $\mathcal{H}_n$ of hypothesis classes is **efficiently properly** PAC-Learnable if there exists a learning rule A such that $\forall_n \forall \epsilon, \delta > 0$, $\exists m(n, \epsilon, \delta)$, $\forall \mathcal{D}$ s.t. $L_{\mathcal{D}}(h) = 0$ for some $h \in \mathcal{H}$, $\forall^{\delta}_{S \sim \mathcal{D}^{m(n,\epsilon,\delta)}}$,

$$L_{\mathcal{D}}\big(A(S)\big) \leq \epsilon$$

and $A(S)(x)$ can be computed in time $poly(n, {}^{1}/_{\epsilon}, log\, {}^{1}/_{\delta})$
and A always outputs a predictor in $\mathcal{H}_n$

- **Definition**: A family $\mathcal{H}_n$ of hypothesis classes is **efficiently properly agnostically** PAC-Learnable if there exists a learning rule A such that $\forall_n \forall \epsilon, \delta > 0$, $\exists m(n, \epsilon, \delta)$, $\forall \mathcal{D}$ $\forall^{\delta}_{S \sim \mathcal{D}^{m(n,\epsilon,\delta)}}$,

$$L_{\mathcal{D}}\big(A(S)\big) \leq \inf_{h \in \mathcal{H}_n} L_{\mathcal{D}}(h) + \epsilon$$

and $A(S)(x)$ can be computed in time $poly(n, {}^{1}/_{\epsilon}, log\, {}^{1}/_{\delta})$
and A always outputs a predictor in $\mathcal{H}_n$

The Scope of Learning Problems Modeled

We next extend our model so that it can be applied to a wide variety of learning tasks. Let us consider some examples of different learning tasks.

Multiclass Classification Our classification does not have to be binary. Take, for example, the task of document classification: We wish to design a program that will be able to classify given documents according to topics (e.g., news, sports, biology, medicine).

Regression In this task, one wishes to find some simple *pattern* in the data - a functional relationship between the $\mathcal{X}$ and $\mathcal{Y}$ components of the data. For example, one wishes to find a linear function that best predicts a baby's birth weight on the basis of ultrasound measures of his head circumference, abdominal circumference, and femur length.

Generalized Loss Functions

Given any set $\mathcal{H}$ (that plays the role of our hypotheses, or models) and some domain Z let ℓ be any function from $\mathcal{H} \times Z$ to the set of nonnegative real numbers, $\ell : \mathcal{H} \times Z \to \mathbb{R}_+$.

We call such functions *loss functions*.

Note that for prediction problems, we have that $Z = \mathcal{X} \times \mathcal{Y}$. However, our notion of the loss function is generalized beyond prediction tasks, and therefore it allows Z to be any domain of examples (for instance, in unsupervised learning tasks such as the one described in Chapter 22, Z is not a product of an instance domain and a label domain).

We now define the *risk function* to be the expected loss of a classifier, $h \in \mathcal{H}$, with respect to a probability distribution D over Z, namely,

$$L_{\mathcal{D}}(h) \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{z \sim \mathcal{D}}[\ell(h, z)]. \tag{3.3}$$

That is, we consider the expectation of the loss of h over objects z picked randomly according to $\mathcal{D}$. Similarly, we define the *empirical risk* to be the expected loss over a given sample $S = (z_1, \ldots, z_m) \in Z^m$, namely,

$$L_S(h) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{i=1}^{m} \ell(h, z_i). \tag{3.4}$$

The loss functions used in the preceding examples of classification and regression tasks are as follows:

- **0–1 loss:** Here, our random variable z ranges over the set of pairs $\mathcal{X} \times \mathcal{Y}$ and the loss function is

$$\ell_{0-1}(h, (x, y)) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } h(x) = y \\ 1 & \text{if } h(x) \neq y \end{cases}$$

  This loss function is used in binary or multiclass classification problems.

- **Square Loss:** Here, our random variable z ranges over the set of pairs $\mathcal{X} \times \mathcal{Y}$ and the loss function is

$$\ell_{\text{sq}}(h, (x, y)) \stackrel{\text{def}}{=} (h(x) - y)^2.$$

This loss function is used in regression problems.

Sample Complexity Results

Definition 0.1. The **sample complexity** of a learning algorithm is the number of examples required to achieve arbitrarily small error (with respect to the optimal hypothesis) with high probability (i.e. close to 1).

Four Cases we care about…

	Realizable	Agnostic
Finite $\lvert\mathcal{H}\rvert$	$N \geq \frac{1}{\epsilon}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{1}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ all $h \in \mathcal{H}$ with $R(h) \geq \epsilon$ have $\hat{R}(h) > 0$.	$N \geq \frac{1}{2\epsilon^2}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{2}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ for all $h \in \mathcal{H}$ we have that $\lvert R(h) - \hat{R}(h)\rvert < \epsilon$.
Infinite $\lvert\mathcal{H}\rvert$		

Sample Complexity Results

Definition 0.1. The **sample complexity** of a learning algorithm is the number of examples required to achieve arbitrarily small error (with respect to the optimal hypothesis) with high probability (i.e. close to 1).

Four Cases we care about…

	Realizable	Agnostic
Finite $\lvert\mathcal{H}\rvert$	$N \geq \frac{1}{\epsilon}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{1}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ all $h \in \mathcal{H}$ with have $\hat{R}(h) > 0$.	$N \geq \frac{1}{2\epsilon^2}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{2}{\delta})\right]$ labeled examples are sufficient so $-\delta)$ for t $\lvert R(h) -$
Infinite $\lvert\mathcal{H}\rvert$		

We need a new definition of "complexity" for a Hypothesis space for these results (see *VC Dimension*)

Questions For Today

1. Given a classifier with zero training error, what can we say about generalization error?
   (Sample Complexity, Realizable Case)

2. Given a classifier with low training error, what can we say about generalization error?
   (Sample Complexity, Agnostic Case)

3. Is there a theoretical justification for regularization to avoid overfitting?
   (Structural Risk Minimization)