7 Nonuniform Learnability

The notions of PAC learnability discussed so far in the book allow the sample sizes to depend on the accuracy and confidence parameters, but they are uniform with respect to the labeling rule and the underlying data distribution. Consequently, classes that are learnable in that respect are limited (they must have a finite VC-dimension, as stated by Theorem 6.7). In this chapter we consider more relaxed, weaker notions of learnability. We discuss the usefulness of such notions and provide characterization of the concept classes that are learnable using these definitions.

We begin this discussion by defining a notion of "nonuniform learnability" that allows the sample size to depend on the hypothesis to which the learner is compared. We then provide a characterization of nonuniform learnability and show that nonuniform learnability is a strict relaxation of agnostic PAC learnability. We also show that a sufficient condition for nonuniform learnability is that $\mathcal{H}$ is a countable union of hypothesis classes, each of which enjoys the uniform convergence property. These results will be proved in Section 7.2 by introducing a new learning paradigm, which is called Structural Risk Minimization (SRM). In Section 7.3 we specify the SRM paradigm for countable hypothesis classes, which yields the Minimum Description Length (MDL) paradigm. The MDL paradigm gives a formal justification to a philosophical principle of induction called Occam's razor. Next, in Section 7.4 we introduce *consistency* as an even weaker notion of learnability. Finally, we discuss the significance and usefulness of the different notions of learnability.

7.1 Nonuniform Learnability

"Nonuniform learnability" allows the sample size to be nonuniform with respect to the different hypotheses with which the learner is competing. We say that a hypothesis h is (ϵ, δ)-competitive with another hypothesis h' if, with probability higher than $(1-\delta)$,

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

In PAC learnability, this notion of "competitiveness" is not very useful, as we are looking for a hypothesis with an absolute low risk (in the realizable case) or

with a low risk compared to the minimal risk achieved by hypotheses in our class (in the agnostic case). Therefore, the sample size depends only on the accuracy and confidence parameters. In nonuniform learnability, however, we allow the sample size to be of the form $m_{\mathcal{H}}(\epsilon, \delta, h)$; namely, it depends also on the h with which we are competing. Formally,

DEFINITION 7.1 A hypothesis class $\mathcal{H}$ is *nonuniformly learnable* if there exist a learning algorithm, A, and a function $m_{\mathcal{H}}^{\text{NUL}} : (0,1)^2 \times \mathcal{H} \to \mathbb{N}$ such that, for every $\epsilon, \delta \in (0,1)$ and for every $h \in \mathcal{H}$, if $m \geq m_{\mathcal{H}}^{\text{NUL}}(\epsilon, \delta, h)$ then for every distribution $\mathcal{D}$, with probability of at least $1 - \delta$ over the choice of $S \sim \mathcal{D}^m$, it holds that

$$L_{\mathcal{D}}(A(S)) \leq L_{\mathcal{D}}(h) + \epsilon.$$

At this point it might be useful to recall the definition of agnostic PAC learnability (Definition 3.3):
A hypothesis class $\mathcal{H}$ is agnostically PAC learnable if there exist a learning algorithm, A, and a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ such that, for every $\epsilon, \delta \in (0,1)$ and for every distribution $\mathcal{D}$, if $m \geq m_{\mathcal{H}}(\epsilon, \delta)$, then with probability of at least $1 - \delta$ over the choice of $S \sim \mathcal{D}^m$ it holds that

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

Note that this implies that for every $h \in \mathcal{H}$

$$L_{\mathcal{D}}(A(S)) \leq L_{\mathcal{D}}(h) + \epsilon.$$

In both types of learnability, we require that the output hypothesis will be (ϵ, δ)-competitive with every other hypothesis in the class. But the difference between these two notions of learnability is the question of whether the sample size m may depend on the hypothesis h to which the error of $A(S)$ is compared. Note that that nonuniform learnability is a relaxation of agnostic PAC learnability. That is, if a class is agnostic PAC learnable then it is also nonuniformly learnable.

7.1.1 Characterizing Nonuniform Learnability

Our goal now is to characterize nonuniform learnability. In the previous chapter we have found a crisp characterization of PAC learnable classes, by showing that a class of binary classifiers is agnostic PAC learnable if and only if its VC-dimension is finite. In the following theorem we find a different characterization for nonuniform learnable classes for the task of binary classification.

THEOREM 7.2 *A hypothesis class $\mathcal{H}$ of binary classifiers is nonuniformly learnable if and only if it is a countable union of agnostic PAC learnable hypothesis classes.*

The proof of Theorem 7.2 relies on the following result of independent interest:

THEOREM 7.3 *Let $\mathcal{H}$ be a hypothesis class that can be written as a countable union of hypothesis classes, $\mathcal{H} = \bigcup_{n\in\mathbb{N}} \mathcal{H}_n$, where each $\mathcal{H}_n$ enjoys the uniform convergence property. Then, $\mathcal{H}$ is nonuniformly learnable.*

Recall that in Chapter 4 we have shown that uniform convergence is sufficient for agnostic PAC learnability. Theorem 7.3 generalizes this result to nonuniform learnability. The proof of this theorem will be given in the next section by introducing a new learning paradigm. We now turn to proving Theorem 7.2.

Proof of Theorem 7.2 First assume that $\mathcal{H} = \bigcup_{n\in\mathbb{N}} \mathcal{H}_n$ where each $\mathcal{H}_n$ is agnostic PAC learnable. Using the fundamental theorem of statistical learning, it follows that each $\mathcal{H}_n$ has the uniform convergence property. Therefore, using Theorem 7.3 we obtain that $\mathcal{H}$ is nonuniform learnable.

For the other direction, assume that $\mathcal{H}$ is nonuniform learnable using some algorithm A. For every $n \in \mathbb{N}$, let $\mathcal{H}_n = \{h \in \mathcal{H} : m^{\text{NUL}}_{\mathcal{H}}(1/8, 1/7, h) \leq n\}$. Clearly, $\mathcal{H} = \cup_{n\in\mathbb{N}} \mathcal{H}_n$. In addition, using the definition of $m^{\text{NUL}}_{\mathcal{H}}$ we know that for any distribution $\mathcal{D}$ that satisfies the realizability assumption with respect to $\mathcal{H}_n$, with probability of at least $6/7$ over $S \sim \mathcal{D}^n$ we have that $L_{\mathcal{D}}(A(S)) \leq 1/8$. Using the fundamental theorem of statistical learning, this implies that the VC-dimension of $\mathcal{H}_n$ must be finite, and therefore $\mathcal{H}_n$ is agnostic PAC learnable. □

The following example shows that nonuniform learnability is a strict relaxation of agnostic PAC learnability; namely, there are hypothesis classes that are nonuniform learnable but are not agnostic PAC learnable.

Example 7.1 Consider a binary classification problem with the instance domain being $\mathcal{X} = \mathbb{R}$. For every $n \in \mathbb{N}$ let $\mathcal{H}_n$ be the class of polynomial classifiers of degree n; namely, $\mathcal{H}_n$ is the set of all classifiers of the form $h(x) = \text{sign}(p(x))$ where $p : \mathbb{R} \to \mathbb{R}$ is a polynomial of degree n. Let $\mathcal{H} = \bigcup_{n\in\mathbb{N}} \mathcal{H}_n$. Therefore, $\mathcal{H}$ is the class of all polynomial classifiers over $\mathbb{R}$. It is easy to verify that $\text{VCdim}(\mathcal{H}) = \infty$ while $\text{VCdim}(\mathcal{H}_n) = n + 1$ (see Exercise 12). Hence, $\mathcal{H}$ is not PAC learnable, while on the basis of Theorem 7.3, $\mathcal{H}$ is nonuniformly learnable.

7.2 Structural Risk Minimization

So far, we have encoded our prior knowledge by specifying a hypothesis class $\mathcal{H}$, which we believe includes a good predictor for the learning task at hand. Yet another way to express our prior knowledge is by specifying preferences over hypotheses within $\mathcal{H}$. In the Structural Risk Minimization (SRM) paradigm, we do so by first assuming that $\mathcal{H}$ can be written as $\mathcal{H} = \bigcup_{n\in\mathbb{N}} \mathcal{H}_n$ and then specifying a weight function, $w : \mathbb{N} \to [0, 1]$, which assigns a weight to each hypothesis class, $\mathcal{H}_n$, such that a higher weight reflects a stronger preference for the hypothesis class. In this section we discuss how to learn with such prior knowledge. In the next section we describe a couple of important weighting schemes, including Minimum Description Length.

Concretely, let $\mathcal{H}$ be a hypothesis class that can be written as $\mathcal{H} = \bigcup_{n\in\mathbb{N}} \mathcal{H}_n$. For example, $\mathcal{H}$ may be the class of all polynomial classifiers where each $\mathcal{H}_n$ is the class of polynomial classifiers of degree n (see Example 7.1). Assume that for each n, the class $\mathcal{H}_n$ enjoys the uniform convergence property (see Definition 4.3 in Chapter 4) with a sample complexity function $m^{\text{UC}}_{\mathcal{H}_n}(\epsilon, \delta)$. Let us also define the function $\epsilon_n : \mathbb{N} \times (0,1) \to (0,1)$ by

$$\epsilon_n(m, \delta) = \min\{\epsilon \in (0,1) : m^{\text{UC}}_{\mathcal{H}_n}(\epsilon, \delta) \leq m\}. \tag{7.1}$$

In words, we have a fixed sample size m, and we are interested in the lowest possible upper bound on the gap between empirical and true risks achievable by using a sample of m examples.

From the definitions of uniform convergence and ϵ_n, it follows that for every m and δ, with probability of at least $1-\delta$ over the choice of $S \sim \mathcal{D}^m$ we have that

$$\forall h \in \mathcal{H}_n, \quad |L_{\mathcal{D}}(h) - L_S(h)| \leq \epsilon_n(m, \delta). \tag{7.2}$$

Let $w : \mathbb{N} \to [0,1]$ be a function such that $\sum_{n=1}^{\infty} w(n) \leq 1$. We refer to w as a *weight function* over the hypothesis classes $\mathcal{H}_1, \mathcal{H}_2, \ldots$. Such a weight function can reflect the importance that the learner attributes to each hypothesis class, or some measure of the complexity of different hypothesis classes. If $\mathcal{H}$ is a finite union of N hypothesis classes, one can simply assign the same weight of $1/N$ to all hypothesis classes. This equal weighting corresponds to no a priori preference to any hypothesis class. Of course, if one believes (as prior knowledge) that a certain hypothesis class is more likely to contain the correct target function, then it should be assigned a larger weight, reflecting this prior knowledge. When $\mathcal{H}$ is a (countable) infinite union of hypothesis classes, a uniform weighting is not possible but many other weighting schemes may work. For example, one can choose $w(n) = \frac{6}{\pi^2 n^2}$ or $w(n) = 2^{-n}$. Later in this chapter we will provide another convenient way to define weighting functions using description languages.

The SRM rule follows a "bound minimization" approach. This means that the goal of the paradigm is to find a hypothesis that minimizes a certain upper bound on the true risk. The bound that the SRM rule wishes to minimize is given in the following theorem.

THEOREM 7.4 *Let $w : \mathbb{N} \to [0,1]$ be a function such that $\sum_{n=1}^{\infty} w(n) \leq 1$. Let $\mathcal{H}$ be a hypothesis class that can be written as $\mathcal{H} = \bigcup_{n\in\mathbb{N}} \mathcal{H}_n$, where for each n, $\mathcal{H}_n$ satisfies the uniform convergence property with a sample complexity function $m^{UC}_{\mathcal{H}_n}$. Let ϵ_n be as defined in Equation (7.1). Then, for every $\delta \in (0,1)$ and distribution $\mathcal{D}$, with probability of at least $1-\delta$ over the choice of $S \sim \mathcal{D}^m$, the following bound holds (simultaneously) for every $n \in \mathbb{N}$ and $h \in \mathcal{H}_n$.*

$$|L_{\mathcal{D}}(h) - L_S(h)| \leq \epsilon_n(m, w(n) \cdot \delta).$$

Therefore, for every $\delta \in (0,1)$ and distribution $\mathcal{D}$, with probability of at least

$1-\delta$ *it holds that*

$$\forall h \in \mathcal{H}, \quad L_{\mathcal{D}}(h) \leq L_S(h) + \min_{n:h\in\mathcal{H}_n} \epsilon_n(m, w(n)\cdot\delta). \tag{7.3}$$

Proof For each n define $\delta_n = w(n)\delta$. Applying the assumption that uniform convergence holds for all n with the rate given in Equation (7.2), we obtain that if we fix n in advance, then with probability of at least $1-\delta_n$ over the choice of $S \sim \mathcal{D}^m$,

$$\forall h \in \mathcal{H}_n, \quad |L_{\mathcal{D}}(h) - L_S(h)| \leq \epsilon_n(m, \delta_n).$$

Applying the union bound over $n = 1, 2, \ldots$, we obtain that with probability of at least $1 - \sum_n \delta_n = 1 - \delta \sum_n w(n) \geq 1 - \delta$, the preceding holds for all n, which concludes our proof. □

Denote

$$n(h) = \min\{n : h \in \mathcal{H}_n\}, \tag{7.4}$$

and then Equation (7.3) implies that

$$L_{\mathcal{D}}(h) \leq L_S(h) + \epsilon_{n(h)}(m, w(n(h))\cdot\delta).$$

The SRM paradigm searches for h that minimizes this bound, as formalized in the following pseudocode:

Structural Risk Minimization (SRM)

prior knowledge:
$\mathcal{H} = \bigcup_n \mathcal{H}_n$ where $\mathcal{H}_n$ has uniform convergence with $m^{\text{UC}}_{\mathcal{H}_n}$
$w : \mathbb{N} \to [0,1]$ where $\sum_n w(n) \leq 1$
define: ϵ_n as in Equation (7.1) ; $n(h)$ as in Equation (7.4)
input: training set $S \sim \mathcal{D}^m$, confidence δ
output: $h \in \text{argmin}_{h\in\mathcal{H}} \left[L_S(h) + \epsilon_{n(h)}(m, w(n(h))\cdot\delta)\right]$

Unlike the ERM paradigm discussed in previous chapters, we no longer just care about the empirical risk, $L_S(h)$, but we are willing to trade some of our bias toward low empirical risk with a bias toward classes for which $\epsilon_{n(h)}(m, w(n(h))\cdot\delta)$ is smaller, for the sake of a smaller estimation error.

Next we show that the SRM paradigm can be used for nonuniform learning of every class, which is a countable union of uniformly converging hypothesis classes.

THEOREM 7.5 *Let $\mathcal{H}$ be a hypothesis class such that $\mathcal{H} = \bigcup_{n\in\mathbb{N}} \mathcal{H}_n$, where each $\mathcal{H}_n$ has the uniform convergence property with sample complexity $m^{UC}_{\mathcal{H}_n}$. Let $w : \mathbb{N} \to [0,1]$ be such that $w(n) = \frac{6}{n^2\pi^2}$. Then, $\mathcal{H}$ is nonuniformly learnable using the SRM rule with rate*

$$m^{NUL}_{\mathcal{H}}(\epsilon, \delta, h) \;\leq\; m^{UC}_{\mathcal{H}_{n(h)}}\left(\epsilon/2\,,\, \tfrac{6\delta}{(\pi n(h))^2}\right).$$

Proof Let A be the SRM algorithm with respect to the weighting function w. For every $h \in \mathcal{H}$, ϵ, and δ, let $m \geq m^{\text{UC}}_{\mathcal{H}_{n(h)}}(\epsilon, w(n(h))\delta)$. Using the fact that $\sum_n w(n) = 1$, we can apply Theorem 7.4 to get that, with probability of at least $1-\delta$ over the choice of $S \sim \mathcal{D}^m$, we have that for every $h' \in \mathcal{H}$,

$$L_{\mathcal{D}}(h') \leq L_S(h') + \epsilon_{n(h')}(m, w(n(h'))\delta).$$

The preceding holds in particular for the hypothesis $A(S)$ returned by the SRM rule. By the definition of SRM we obtain that

$$L_{\mathcal{D}}(A(S)) \leq \min_{h'} \left[L_S(h') + \epsilon_{n(h')}(m, w(n(h'))\delta)\right] \leq L_S(h) + \epsilon_{n(h)}(m, w(n(h))\delta).$$

Finally, if $m \geq m^{\text{UC}}_{\mathcal{H}_{n(h)}}(\epsilon/2, w(n(h))\delta)$ then clearly $\epsilon_{n(h)}(m, w(n(h))\delta) \leq \epsilon/2$. In addition, from the uniform convergence property of each $\mathcal{H}_n$ we have that with probability of more than $1-\delta$,

$$L_S(h) \leq L_{\mathcal{D}}(h) + \epsilon/2.$$

Combining all the preceding we obtain that $L_{\mathcal{D}}(A(S)) \leq L_{\mathcal{D}}(h) + \epsilon$, which concludes our proof. □

Note that the previous theorem also proves Theorem 7.3.

Remark 7.2 (No-Free-Lunch for Nonuniform Learnability) We have shown that any countable union of classes of finite VC-dimension is nonuniformly learnable. It turns out that, for any infinite domain set, $\mathcal{X}$, the class of all binary valued functions over $\mathcal{X}$ is not a countable union of classes of finite VC-dimension. We leave the proof of this claim as a (nontrivial) exercise (see Exercise 5). It follows that, in some sense, the no free lunch theorem holds for nonuniform learning as well: namely, whenever the domain is not finite, there exists no nonuniform learner with respect to the class of all deterministic binary classifiers (although for each such classifier there exists a trivial algorithm that learns it – ERM with respect to the hypothesis class that contains only this classifier).

It is interesting to compare the nonuniform learnability result given in Theorem 7.5 to the task of agnostic PAC learning any specific $\mathcal{H}_n$ separately. The prior knowledge, or bias, of a nonuniform learner for $\mathcal{H}$ is weaker – it is searching for a model throughout the entire class $\mathcal{H}$, rather than being focused on one specific $\mathcal{H}_n$. The cost of this weakening of prior knowledge is the increase in sample complexity needed to compete with any specific $h \in \mathcal{H}_n$. For a concrete evaluation of this gap, consider the task of binary classification with the zero-one loss. Assume that for all n, $\text{VCdim}(\mathcal{H}_n) = n$. Since $m^{\text{UC}}_{\mathcal{H}_n}(\epsilon, \delta) = C\,\frac{n+\log(1/\delta)}{\epsilon^2}$ (where C is the contant appearing in Theorem 6.8), a straightforward calculation shows that

$$m^{\text{NUL}}_{\mathcal{H}}(\epsilon, \delta, h) - m^{\text{UC}}_{\mathcal{H}_n}(\epsilon/2, \delta) \leq 4C\frac{2\log(2n)}{\epsilon^2}.$$

That is, the cost of relaxing the learner's prior knowledge from a specific $\mathcal{H}_n$ that contains the target h to a countable union of classes depends on the log of

the index of the first class in which h resides. That cost increases with the index of the class, which can be interpreted as reflecting the value of knowing a good priority order on the hypotheses in $\mathcal{H}$.

7.3 Minimum Description Length and Occam's Razor

Let $\mathcal{H}$ be a countable hypothesis class. Then, we can write $\mathcal{H}$ as a countable union of singleton classes, namely, $\mathcal{H} = \bigcup_{n\in\mathbb{N}}\{h_n\}$. By Hoeffding's inequality (Lemma 4.5), each singleton class has the uniform convergence property with rate $m^{\mathrm{UC}}(\epsilon,\delta) = \frac{\log(2/\delta)}{2\epsilon^2}$. Therefore, the function ϵ_n given in Equation (7.1) becomes $\epsilon_n(m,\delta) = \sqrt{\frac{\log(2/\delta)}{2m}}$ and the SRM rule becomes

$$\operatorname*{argmin}_{h_n\in\mathcal{H}}\left[L_S(h) + \sqrt{\frac{-\log(w(n)) + \log(2/\delta)}{2m}}\right].$$

Equivalently, we can think of w as a function from $\mathcal{H}$ to $[0,1]$, and then the SRM rule becomes

$$\operatorname*{argmin}_{h\in\mathcal{H}}\left[L_S(h) + \sqrt{\frac{-\log(w(h)) + \log(2/\delta)}{2m}}\right].$$

It follows that in this case, the prior knowledge is solely determined by the weight we assign to each hypothesis. We assign higher weights to hypotheses that we believe are more likely to be the correct one, and in the learning algorithm we prefer hypotheses that have higher weights.

In this section we discuss a particular convenient way to define a weight function over $\mathcal{H}$, which is derived from the length of descriptions given to hypotheses. Having a hypothesis class, one can wonder about how we describe, or represent, each hypothesis in the class. We naturally fix some description language. This can be English, or a programming language, or some set of mathematical formulas. In any of these languages, a description consists of finite strings of symbols (or characters) drawn from some fixed alphabet. We shall now formalize these notions.

Let $\mathcal{H}$ be the hypothesis class we wish to describe. Fix some finite set Σ of symbols (or "characters"), which we call the alphabet. For concreteness, we let $\Sigma = \{0,1\}$. A string is a finite sequence of symbols from Σ; for example, $\boldsymbol{\sigma} = (0,1,1,1,0)$ is a string of length 5. We denote by $|\boldsymbol{\sigma}|$ the length of a string. The set of all finite length strings is denoted Σ^*. A description language for $\mathcal{H}$ is a function $d : \mathcal{H} \to \Sigma^*$, mapping each member h of $\mathcal{H}$ to a string $d(h)$. $d(h)$ is called "the description of h," and its length is denoted by $|h|$.

We shall require that description languages be *prefix-free*; namely, for every distinct h, h', $d(h)$ is not a prefix of $d(h')$. That is, we do not allow that any string $d(h)$ is exactly the first $|h|$ symbols of any longer string $d(h')$. Prefix-free collections of strings enjoy the following combinatorial property:

LEMMA 7.6 (Kraft Inequality) *If $\mathcal{S} \subseteq \{0,1\}^*$ is a prefix-free set of strings, then*

$$\sum_{\boldsymbol{\sigma} \in \mathcal{S}} \frac{1}{2^{|\boldsymbol{\sigma}|}} \leq 1.$$

Proof Define a probability distribution over the members of $\mathcal{S}$ as follows: Repeatedly toss an unbiased coin, with faces labeled 0 and 1, until the sequence of outcomes is a member of $\mathcal{S}$; at that point, stop. For each $\boldsymbol{\sigma} \in \mathcal{S}$, let $P(\boldsymbol{\sigma})$ be the probability that this process generates the string $\boldsymbol{\sigma}$. Note that since $\mathcal{S}$ is prefix-free, for every $\boldsymbol{\sigma} \in \mathcal{S}$, if the coin toss outcomes follow the bits of $\boldsymbol{\sigma}$ then we will stop only once the sequence of outcomes equals $\boldsymbol{\sigma}$. We therefore get that, for every $\boldsymbol{\sigma} \in \mathcal{S}$, $P(\boldsymbol{\sigma}) = \frac{1}{2^{|\boldsymbol{\sigma}|}}$. Since probabilities add up to at most 1, our proof is concluded. □

In light of Kraft's inequality, any prefix-free description language of a hypothesis class, $\mathcal{H}$, gives rise to a weighting function w over that hypothesis class – we will simply set $w(h) = \frac{1}{2^{|h|}}$. This observation immediately yields the following:

THEOREM 7.7 *Let $\mathcal{H}$ be a hypothesis class and let $d : \mathcal{H} \to \{0,1\}^*$ be a prefix-free description language for $\mathcal{H}$. Then, for every sample size, m, every confidence parameter, $\delta > 0$, and every probability distribution, $\mathcal{D}$, with probability greater than $1 - \delta$ over the choice of $S \sim \mathcal{D}^m$ we have that,*

$$\forall h \in \mathcal{H}, \quad L_{\mathcal{D}}(h) \leq L_S(h) + \sqrt{\frac{|h| + \ln(2/\delta)}{2m}},$$

where $|h|$ is the length of $d(h)$.

Proof Choose $w(h) = 1/2^{|h|}$, apply Theorem 7.4 with $\epsilon_n(m, \delta) = \sqrt{\frac{\ln(2/\delta)}{2m}}$, and note that $\ln(2^{|h|}) = |h| \ln(2) < |h|$. □

As was the case with Theorem 7.4, this result suggests a learning paradigm for $\mathcal{H}$ – given a training set, S, search for a hypothesis $h \in \mathcal{H}$ that minimizes the bound, $L_S(h) + \sqrt{\frac{|h|+\ln(2/\delta)}{2m}}$. In particular, it suggests trading off empirical risk for saving description length. This yields the Minimum Description Length learning paradigm.

Minimum Description Length (MDL)

prior knowledge:
$\mathcal{H}$ is a countable hypothesis class
$\mathcal{H}$ is described by a prefix-free language over $\{0,1\}$
For every $h \in \mathcal{H}$, $|h|$ is the length of the representation of h
input: A training set $S \sim \mathcal{D}^m$, confidence δ
output: $h \in \operatorname{argmin}_{h \in \mathcal{H}} \left[L_S(h) + \sqrt{\frac{|h|+\ln(2/\delta)}{2m}} \right]$

Example 7.3 Let $\mathcal{H}$ be the class of all predictors that can be implemented using some programming language, say, C++. Let us represent each program using the

binary string obtained by running the gzip command on the program (this yields a prefix-free description language over the alphabet $\{0,1\}$). Then, $|h|$ is simply the length (in bits) of the output of gzip when running on the C++ program corresponding to h.

7.3.1 Occam's Razor

Theorem 7.7 suggests that, having two hypotheses sharing the same empirical risk, the true risk of the one that has shorter description can be bounded by a lower value. Thus, this result can be viewed as conveying a philosophical message:

A short explanation (that is, a hypothesis that has a short length) tends to be more valid than a long explanation.

This is a well known principle, called Occam's razor, after William of Ockham, a 14th-century English logician, who is believed to have been the first to phrase it explicitly. Here, we provide one possible justification to this principle. The inequality of Theorem 7.7 shows that the more complex a hypothesis h is (in the sense of having a longer description), the larger the sample size it has to fit to guarantee that it has a small true risk, $L_{\mathcal{D}}(h)$.

At a second glance, our Occam razor claim might seem somewhat problematic. In the context in which the Occam razor principle is usually invoked in science, the language according to which complexity is measured is a natural language, whereas here we may consider any arbitrary abstract description language. Assume that we have two hypotheses such that $|h'|$ is much smaller than $|h|$. By the preceding result, if both have the same error on a given training set, S, then the true error of h may be much higher than the true error of h', so one should prefer h' over h. However, we could have chosen a different description language, say, one that assigns a string of length 3 to h and a string of length 100000 to h'. Suddenly it looks as if one should prefer h over h'. But these are the same h and h' for which we argued two sentences ago that h' should be preferable. Where is the catch here?

Indeed, there is no inherent generalizability difference between hypotheses. The crucial aspect here is the dependency order between the initial choice of language (or, preference over hypotheses) and the training set. As we know from the basic Hoeffding's bound (Equation (4.2)), if we commit to any hypothesis *before* seeing the data, then we are guaranteed a rather small estimation error term $L_{\mathcal{D}}(h) \leq L_S(h) + \sqrt{\frac{\ln(2/\delta)}{2m}}$. Choosing a description language (or, equivalently, some weighting of hypotheses) is a weak form of committing to a hypothesis. Rather than committing to a single hypothesis, we spread out our commitment among many. As long as it is done independently of the training sample, our generalization bound holds. Just as the choice of a single hypothesis to be evaluated by a sample can be arbitrary, so is the choice of description language.

7.4 Other Notions of Learnability – Consistency

The notion of learnability can be further relaxed by allowing the needed sample sizes to depend not only on ϵ, δ, and h but also on the underlying data-generating probability distribution $\mathcal{D}$ (that is used to generate the training sample and to determine the risk). This type of performance guarantee is captured by the notion of *consistency*[1] of a learning rule.

DEFINITION 7.8 (Consistency) Let Z be a domain set, let $\mathcal{P}$ be a set of probability distributions over Z, and let $\mathcal{H}$ be a hypothesis class. A learning rule A is *consistent* with respect to $\mathcal{H}$ and $\mathcal{P}$ if there exists a function $m_{\mathcal{H}}^{\text{CON}} : (0,1)^2 \times \mathcal{H} \times \mathcal{P} \to \mathbb{N}$ such that, for every $\epsilon, \delta \in (0,1)$, every $h \in \mathcal{H}$, and every $\mathcal{D} \in \mathcal{P}$, if $m \geq m_{\mathcal{H}}^{\text{NUL}}(\epsilon, \delta, h, \mathcal{D})$ then with probability of at least $1 - \delta$ over the choice of $S \sim \mathcal{D}^m$ it holds that

$$L_{\mathcal{D}}(A(S)) \leq L_{\mathcal{D}}(h) + \epsilon.$$

If $\mathcal{P}$ is the set of all distributions,[2] we say that A is *universally consistent* with respect to $\mathcal{H}$.

The notion of consistency is, of course, a relaxation of our previous notion of nonuniform learnability. Clearly if an algorithm nonuniformly learns a class $\mathcal{H}$ it is also universally consistent for that class. The relaxation is strict in the sense that there are consistent learning rules that are not successful nonuniform learners. For example, the algorithm `Memorize` defined in Example 7.4 later is universally consistent for the class of all binary classifiers over $\mathbb{N}$. However, as we have argued before, this class is not nonuniformly learnable.

Example 7.4 Consider the classification prediction algorithm `Memorize` defined as follows. The algorithm memorizes the training examples, and, given a test point x, it predicts the majority label among all labeled instances of x that exist in the training sample (and some fixed default label if no instance of x appears in the training set). It is possible to show (see Exercise 6) that the `Memorize` algorithm is universally consistent for every countable domain $\mathcal{X}$ and a finite label set $\mathcal{Y}$ (w.r.t. the zero-one loss).

Intuitively, it is not obvious that the `Memorize` algorithm should be viewed as a *learner*, since it lacks the aspect of generalization, namely, of using observed data to predict the labels of unseen examples. The fact that `Memorize` is a consistent algorithm for the class of all functions over any countable domain set therefore raises doubt about the usefulness of consistency guarantees. Furthermore, the sharp-eyed reader may notice that the "bad learner" we introduced in Chapter 2,

[1] In the literature, consistency is often defined using the notion of either convergence in probability (corresponding to weak consistency) or almost sure convergence (corresponding to strong consistency).

[2] Formally, we assume that Z is endowed with some sigma algebra of subsets Ω, and by "all distributions" we mean all probability distributions that have Ω contained in their associated family of measurable subsets.

which led to overfitting, is in fact the `Memorize` algorithm. In the next section we discuss the significance of the different notions of learnability and revisit the No-Free-Lunch theorem in light of the different definitions of learnability.

7.5 Discussing the Different Notions of Learnability

We have given three definitions of learnability and we now discuss their usefulness. As is usually the case, the usefulness of a mathematical definition depends on what we need it for. We therefore list several possible goals that we aim to achieve by defining learnability and discuss the usefulness of the different definitions in light of these goals.

What Is the Risk of the Learned Hypothesis?
The first possible goal of deriving performance guarantees on a learning algorithm is bounding the risk of the output predictor. Here, both PAC learning and nonuniform learning give us an upper bound on the true risk of the learned hypothesis based on its empirical risk. Consistency guarantees do not provide such a bound. However, it is always possible to estimate the risk of the output predictor using a validation set (as will be described in Chapter 11).

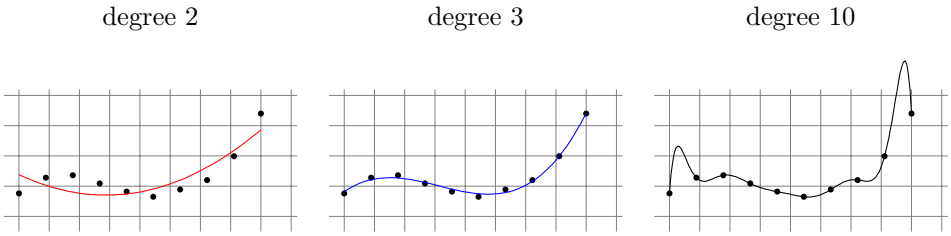
How Many Examples Are Required to Be as Good as the Best Hypothesis in $\mathcal{H}$?
When approaching a learning problem, a natural question is how many examples we need to collect in order to learn it. Here, PAC learning gives a crisp answer. However, for both nonuniform learning and consistency, we do not know in advance how many examples are required to learn $\mathcal{H}$. In nonuniform learning this number depends on the best hypothesis in $\mathcal{H}$, and in consistency it also depends on the underlying distribution. In this sense, PAC learning is the only useful definition of learnability. On the flip side, one should keep in mind that even if the estimation error of the predictor we learn is small, its risk may still be large if $\mathcal{H}$ has a large approximation error. So, for the question "How many examples are required to be as good as the Bayes optimal predictor?" even PAC guarantees do not provide us with a crisp answer. This reflects the fact that the usefulness of PAC learning relies on the quality of our prior knowledge.

PAC guarantees also help us to understand what we should do next if our learning algorithm returns a hypothesis with a large risk, since we can bound the part of the error that stems from estimation error and therefore know how much of the error is attributed to approximation error. If the approximation error is large, we know that we should use a different hypothesis class. Similarly, if a nonuniform algorithm fails, we can consider a different weighting function over (subsets of) hypotheses. However, when a consistent algorithm fails, we have no idea whether this is because of the estimation error or the approximation error. Furthermore, even if we are sure we have a problem with the estimation

error term, we do not know how many more examples are needed to make the estimation error small.

How to Learn? How to Express Prior Knowledge?
Maybe the most useful aspect of the theory of learning is in providing an answer to the question of "how to learn." The definition of PAC learning yields the limitation of learning (via the No-Free-Lunch theorem) and the necessity of prior knowledge. It gives us a crisp way to encode prior knowledge by choosing a hypothesis class, and once this choice is made, we have a generic learning rule – ERM. The definition of nonuniform learnability also yields a crisp way to encode prior knowledge by specifying weights over (subsets of) hypotheses of $\mathcal{H}$. Once this choice is made, we again have a generic learning rule – SRM. The SRM rule is also advantageous in model selection tasks, where prior knowledge is partial. We elaborate on model selection in Chapter 11 and here we give a brief example.

Consider the problem of fitting a one dimensional polynomial to data; namely, our goal is to learn a function, $h : \mathbb{R} \to \mathbb{R}$, and as prior knowledge we consider the hypothesis class of polynomials. However, we might be uncertain regarding which degree d would give the best results for our data set: A small degree might not fit the data well (i.e., it will have a large approximation error), whereas a high degree might lead to overfitting (i.e., it will have a large estimation error). In the following we depict the result of fitting a polynomial of degrees 2, 3, and 10 to the same training set.



It is easy to see that the empirical risk decreases as we enlarge the degree. Therefore, if we choose $\mathcal{H}$ to be the class of all polynomials up to degree 10 then the ERM rule with respect to this class would output a 10 degree polynomial and would overfit. On the other hand, if we choose too small a hypothesis class, say, polynomials up to degree 2, then the ERM would suffer from underfitting (i.e., a large approximation error). In contrast, we can use the SRM rule on the set of all polynomials, while ordering subsets of $\mathcal{H}$ according to their degree, and this will yield a 3rd degree polynomial since the combination of its empirical risk and the bound on its estimation error is the smallest. In other words, the SRM rule enables us to select the right model on the basis of the data itself. The price we pay for this flexibility (besides a slight increase of the estimation error relative to PAC learning w.r.t. the optimal degree) is that we do not know in

advance how many examples are needed to compete with the best hypothesis in $\mathcal{H}$.

Unlike the notions of PAC learnability and nonuniform learnability, the definition of consistency does not yield a natural learning paradigm or a way to encode prior knowledge. In fact, in many cases there is no need for prior knowledge at all. For example, we saw that even the `Memorize` algorithm, which intuitively should not be called a learning algorithm, is a consistent algorithm for any class defined over a countable domain and a finite label set. This hints that consistency is a very weak requirement.

Which Learning Algorithm Should We Prefer?

One may argue that even though consistency is a weak requirement, it is desirable that a learning algorithm will be consistent with respect to the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$, which gives us a guarantee that for enough training examples, we will always be as good as the Bayes optimal predictor. Therefore, if we have two algorithms, where one is consistent and the other one is not consistent, we should prefer the consistent algorithm. However, this argument is problematic for two reasons. First, maybe it is the case that for most "natural" distributions we will observe in practice that the sample complexity of the consistent algorithm will be so large so that in every practical situation we will not obtain enough examples to enjoy this guarantee. Second, it is not very hard to make any PAC or nonuniform learner consistent with respect to the class of all functions from $\mathcal{X}$ to $\mathcal{Y}$. Concretely, consider a countable domain, $\mathcal{X}$, a finite label set $\mathcal{Y}$, and a hypothesis class, $\mathcal{H}$, of functions from $\mathcal{X}$ to $\mathcal{Y}$. We can make any nonuniform learner for $\mathcal{H}$ be consistent with respect to the class of *all* classifiers from $\mathcal{X}$ to $\mathcal{Y}$ using the following simple trick: Upon receiving a training set, we will first run the nonuniform learner over the training set, and then we will obtain a bound on the true risk of the learned predictor. If this bound is small enough we are done. Otherwise, we revert to the `Memorize` algorithm. This simple modification makes the algorithm consistent with respect to all functions from $\mathcal{X}$ to $\mathcal{Y}$. Since it is easy to make any algorithm consistent, it may not be wise to prefer one algorithm over the other just because of consistency considerations.

7.5.1 The No-Free-Lunch Theorem Revisited

Recall that the No-Free-Lunch theorem (Theorem 5.1 from Chapter 5) implies that no algorithm can learn the class of all classifiers over an infinite domain. In contrast, in this chapter we saw that the `Memorize` algorithm is consistent with respect to the class of all classifiers over a countable infinite domain. To understand why these two statements do not contradict each other, let us first recall the formal statement of the No-Free-Lunch theorem.

Let $\mathcal{X}$ be a countable infinite domain and let $\mathcal{Y} = \{\pm 1\}$. The No-Free-Lunch theorem implies the following: For any algorithm, A, and a training set size, m, there exist a distribution over $\mathcal{X}$ and a function $h^\star : \mathcal{X} \to \mathcal{Y}$, such that if A

will get a sample of m i.i.d. training examples, labeled by $h^\star$, then A is likely to return a classifier with a larger error.

The consistency of `Memorize` implies the following: For every distribution over $\mathcal{X}$ and a labeling function $h^\star : \mathcal{X} \to \mathcal{Y}$, there exists a training set size m (that depends on the distribution and on $h^\star$) such that if `Memorize` receives at least m examples it is likely to return a classifier with a small error.

We see that in the No-Free-Lunch theorem, we first fix the training set size, and then find a distribution and a labeling function that are bad for this training set size. In contrast, in consistency guarantees, we first fix the distribution and the labeling function, and only then do we find a training set size that suffices for learning this particular distribution and labeling function.

7.6 Summary

We introduced nonuniform learnability as a relaxation of PAC learnability and consistency as a relaxation of nonuniform learnability. This means that even classes of infinite VC-dimension can be learnable, in some weaker sense of learnability. We discussed the usefulness of the different definitions of learnability.

For hypothesis classes that are countable, we can apply the Minimum Description Length scheme, where hypotheses with shorter descriptions are preferred, following the principle of Occam's razor. An interesting example is the hypothesis class of all predictors we can implement in C++ (or any other programming language), which we can learn (nonuniformly) using the MDL scheme.

Arguably, the class of all predictors we can implement in C++ is a powerful class of functions and probably contains all that we can hope to learn in practice. The ability to learn this class is impressive, and, seemingly, this chapter should have been the last chapter of this book. This is not the case, because of the computational aspect of learning: that is, the runtime needed to apply the learning rule. For example, to implement the MDL paradigm with respect to all C++ programs, we need to perform an exhaustive search over all C++ programs, which will take forever. Even the implementation of the ERM paradigm with respect to all C++ programs of description length at most 1000 bits requires an exhaustive search over 2^{1000} hypotheses. While the sample complexity of learning this class is just $\frac{1000+\log(2/\delta)}{\epsilon^2}$, the runtime is $\geq 2^{1000}$. This is a huge number – much larger than the number of atoms in the visible universe. In the next chapter we formally define the computational complexity of learning. In the second part of this book we will study hypothesis classes for which the ERM or SRM schemes can be implemented efficiently.

7.7 Bibliographic Remarks

Our definition of nonuniform learnability is related to the definition of an Occam-algorithm in Blumer, Ehrenfeucht, Haussler & Warmuth (1987). The concept of SRM is due to (Vapnik & Chervonenkis 1974, Vapnik 1995). The concept of MDL is due to (Rissanen 1978, Rissanen 1983). The relation between SRM and MDL is discussed in Vapnik (1995). These notions are also closely related to the notion of *regularization* (e.g. Tikhonov (1943)). We will elaborate on regularization in the second part of this book.

The notion of consistency of estimators dates back to Fisher (1922). Our presentation of consistency follows Steinwart & Christmann (2008), who also derived several no-free-lunch theorems.

7.8 Exercises

1. Prove that for any finite class $\mathcal{H}$, and any description language $d : \mathcal{H} \to \{0,1\}^*$, the VC-dimension of $\mathcal{H}$ is at most $2 \sup\{|d(h)| : h \in \mathcal{H}\}$ – the maximum description length of a predictor in $\mathcal{H}$. Furthermore, if d is a prefix-free description then $\text{VCdim}(H) \leq \sup\{|d(h)| : h \in \mathcal{H}\}$.
2. Let $\mathcal{H} = \{h_n : n \in \mathbb{N}\}$ be an infinite countable hypothesis class for binary classification. Show that it is impossible to assign weights to the hypotheses in $\mathcal{H}$ such that
   - $\mathcal{H}$ could be learnt nonuniformly using these weights. That is, the weighting function $w : \mathcal{H} \to [0,1]$ should satisfy the condition $\sum_{h \in \mathcal{H}} w(h) \leq 1$.
   - The weights would be monotonically nondecreasing. That is, if $i < j$, then $w(h_i) \leq w(h_j)$.
3. - Consider a hypothesis class $\mathcal{H} = \bigcup_{n=1}^{\infty} \mathcal{H}_n$, where for every $n \in \mathbb{N}$, $\mathcal{H}_n$ is finite. Find a weighting function $w : \mathcal{H} \to [0,1]$ such that $\sum_{h \in \mathcal{H}} w(h) \leq 1$ and so that for all $h \in \mathcal{H}$, $w(h)$ is determined by $n(h) = \min\{n : h \in \mathcal{H}_n\}$ and by $|\mathcal{H}_{n(h)}|$.
   - **(*)** Define such a function w when for all n $\mathcal{H}_n$ is countable (possibly infinite).
4. Let $\mathcal{H}$ be some hypothesis class. For any $h \in \mathcal{H}$, let $|h|$ denote the description length of h, according to some fixed description language. Consider the MDL learning paradigm in which the algorithm returns:

$$h_S \in \arg\min_{h \in \mathcal{H}} \left[L_S(h) + \sqrt{\frac{|h| + \ln(2/\delta)}{2m}} \right],$$

   where S is a sample of size m. For any $B > 0$, let $\mathcal{H}_B = \{h \in \mathcal{H} : |h| \leq B\}$, and define

$$h_B^* = \arg\min_{h \in \mathcal{H}_B} L_{\mathcal{D}}(h).$$

Prove a bound on $L_{\mathcal{D}}(h_S) - L_{\mathcal{D}}(h^*_B)$ in terms of B, the confidence parameter δ, and the size of the training set m.

- Note: Such bounds are known as *oracle inequalities* in the literature: We wish to estimate how good we are compared to a reference classifier (or "oracle") h^*_B.

5. In this question we wish to show a No-Free-Lunch result for nonuniform learnability: namely, that, over any infinite domain, the class of *all* functions is not learnable even under the relaxed nonuniform variation of learning.

   Recall that an algorithm, A, *nonuniformly learns* a hypothesis class $\mathcal{H}$ if there exists a function $m^{\text{NUL}}_{\mathcal{H}} : (0,1)^2 \times \mathcal{H} \to \mathbb{N}$ such that, for every $\epsilon, \delta \in (0,1)$ and for every $h \in \mathcal{H}$, if $m \geq m^{\text{NUL}}_{\mathcal{H}}(\epsilon, \delta, h)$ then for every distribution $\mathcal{D}$, with probability of at least $1-\delta$ over the choice of $S \sim D^m$, it holds that

   $$L_{\mathcal{D}}(A(S)) \leq L_{\mathcal{D}}(h) + \epsilon.$$

   If such an algorithm exists then we say that $\mathcal{H}$ is *nonuniformly learnable*.

   1. Let A be a nonuniform learner for a class $\mathcal{H}$. For each $n \in \mathbb{N}$ define $\mathcal{H}^A_n = \{h \in \mathcal{H} : m^{\text{NUL}}(0.1, 0.1, h) \leq n\}$. Prove that each such class $\mathcal{H}_n$ has a finite VC-dimension.
   2. Prove that if a class $\mathcal{H}$ is nonuniformly learnable then there are classes $\mathcal{H}_n$ so that $\mathcal{H} = \bigcup_{n\in\mathbb{N}} \mathcal{H}_n$ and, for every $n \in \mathbb{N}$, $\text{VCdim}(\mathcal{H}_n)$ is finite.
   3. Let $\mathcal{H}$ be a class that shatters an infinite set. Then, for every sequence of classes $(\mathcal{H}_n : n \in \mathbb{N})$ such that $\mathcal{H} = \bigcup_{n\in\mathbb{N}} \mathcal{H}_n$, there exists some n for which $\text{VCdim}(\mathcal{H}_n) = \infty$.
      Hint: Given a class $\mathcal{H}$ that shatters some infinite set K, and a sequence of classes $(\mathcal{H}_n : n \in \mathbb{N})$, each having a finite VC-dimension, start by defining subsets $K_n \subseteq K$ such that, for all n, $|K_n| > \text{VCdim}(\mathcal{H}_n)$ and for any $n \neq m$, $K_n \cap K_m = \emptyset$. Now, pick for each such K_n a function $f_n : K_n \to \{0,1\}$ so that no $h \in \mathcal{H}_n$ agrees with f_n on the domain K_n. Finally, define $f : X \to \{0,1\}$ by combining these f_n's and prove that $f \in \left(\mathcal{H} \setminus \bigcup_{n\in\mathbb{N}} \mathcal{H}_n\right)$.
   4. Construct a class $\mathcal{H}_1$ of functions from the unit interval $[0,1]$ to $\{0,1\}$ that is nonuniformly learnable but not PAC learnable.
   5. Construct a class $\mathcal{H}_2$ of functions from the unit interval $[0,1]$ to $\{0,1\}$ that is not nonuniformly learnable.

6. In this question we wish to show that the algorithm `Memorize` is a consistent learner for every class of (binary-valued) functions over any countable domain. Let $\mathcal{X}$ be a countable domain and let $\mathcal{D}$ be a probability distribution over $\mathcal{X}$.

   1. Let $\{x_i : i \in \mathbb{N}\}$ be an enumeration of the elements of $\mathcal{X}$ so that for all $i \leq j$, $\mathcal{D}(\{x_i\}) \leq \mathcal{D}(\{x_j\})$. Prove that

      $$\lim_{n\to\infty} \sum_{i \geq n} \mathcal{D}(\{x_i\}) = 0.$$

   2. Given any $\epsilon > 0$ prove that there exists $\epsilon_D > 0$ such that

      $$\mathcal{D}(\{x \in \mathcal{X} : \mathcal{D}(\{x\}) < \epsilon_D\}) < \epsilon.$$

3. Prove that for every $\eta > 0$, if n is such that $\mathcal{D}(\{x_i\}) < \eta$ for all $i > n$, then for every $m \in \mathbb{N}$,

$$\mathop{\mathbb{P}}_{S \sim \mathcal{D}^m} [\exists x_i : (D(\{x_i\}) > \eta \text{ and } x_i \notin S)] \leq n e^{-\eta m}.$$

4. Conclude that if $\mathcal{X}$ is countable then for every probability distribution $\mathcal{D}$ over $\mathcal{X}$ there exists a function $m_{\mathcal{D}} : (0,1) \times (0,1) \to \mathbb{N}$ such that for every $\epsilon, \delta > 0$ if $m > m_{\mathcal{D}}(\epsilon, \delta)$ then

$$\mathop{\mathbb{P}}_{S \sim \mathcal{D}^m} [\mathcal{D}(\{x : x \notin S\}) > \epsilon] < \delta.$$

5. Prove that `Memorize` is a consistent learner for every class of (binary-valued) functions over any countable domain.