Chapter 2

Training versus Testing

Before the final exam, a professor may hand out some practice problems and solutions to the class. Although these problems are not the exact ones that will appear on the exam, studying them will help you do better. They are the 'training set' in your learning.

If the professor's goal is to help you do better in the exam, why not give out the exam problems themselves? Well, nice try ☺. Doing well in the exam is not the goal in and of itself. The goal is for you to learn the course material. The exam is merely a way to gauge how well you have learned the material. If the exam problems are known ahead of time, your performance on them will no longer accurately gauge how well you have learned.

The same distinction between training and testing happens in learning from data. In this chapter, we will develop a mathematical theory that characterizes this distinction. We will also discuss the conceptual and practical implications of the contrast between training and testing.

2.1 Theory of Generalization

The out-of-sample error E_{out} measures how well our training on $\mathcal{D}$ has *generalized* to data that we have not seen before. E_{out} is based on the performance over the entire input space $\mathcal{X}$. Intuitively, if we want to estimate the value of E_{out} using a sample of data points, these points must be 'fresh' test points that have not been used for training, similar to the questions on the final exam that have not been used for practice.

The in-sample error E_{in}, by contrast, is based on data points that have been used for training. It expressly measures training performance, similar to your performance on the practice problems that you got before the final exam. Such performance has the benefit of looking at the solutions and adjusting accordingly, and may not reflect the ultimate performance in a real test. We began the analysis of in-sample error in Chapter 1, and we will extend this

analysis to the general case in this chapter. We will also make the contrast between a training set and a test set more precise.

A word of warning: this chapter is the heaviest in this book in terms of mathematical abstraction. To make it easier on the not-so-mathematically inclined, we will tell you which part you can safely skip without 'losing the plot'. The mathematical results provide fundamental insights into learning from data, and we will interpret these results in practical terms.

Generalization error. We have already discussed how the value of E_{in} does not always generalize to a similar value of E_{out}. Generalization is a key issue in learning. One can define the *generalization error* as the discrepancy between E_{in} and E_{out}.[1] The Hoeffding Inequality (1.6) provides a way to characterize the generalization error with a probabilistic bound,

$$\mathbb{P}\left[|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon\right] \leq 2Me^{-2\epsilon^2 N},$$

for any $\epsilon > 0$. This can be rephrased as follows. Pick a tolerance level δ, for example $\delta = 0.05$, and assert with probability at least $1 - \delta$ that

$$E_{\text{out}}(g) \leq E_{\text{in}}(g) + \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}. \tag{2.1}$$

We refer to the type of inequality in (2.1) as a *generalization bound* because it bounds E_{out} in terms of E_{in}. To see that the Hoeffding Inequality implies this generalization bound, we rewrite (1.6) as follows: with probability at least $1 - 2Me^{-2N\epsilon^2}$, $|E_{\text{out}} - E_{\text{in}}| \leq \epsilon$, which implies $E_{\text{out}} \leq E_{\text{in}} + \epsilon$. We may now identify $\delta = 2Me^{-2N\epsilon^2}$, from which $\epsilon = \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$, and (2.1) follows.

Notice that the other side of $|E_{\text{out}} - E_{\text{in}}| \leq \epsilon$ also holds, that is, $E_{\text{out}} \geq E_{\text{in}} - \epsilon$ for all $h \in \mathcal{H}$. This is important for learning, but in a more subtle way. Not only do we want to know that the hypothesis g that we choose (say the one with the best training error) will continue to do well out of sample (i.e., $E_{\text{out}} \leq E_{\text{in}} + \epsilon$), but we also want to be sure that we did the best we could with our $\mathcal{H}$ (no other hypothesis $h \in \mathcal{H}$ has $E_{\text{out}}(h)$ significantly better than $E_{\text{out}}(g)$). The $E_{\text{out}}(h) \geq E_{\text{in}}(h) - \epsilon$ direction of the bound assures us that we couldn't do much better because every hypothesis with a higher E_{in} than the g we have chosen will have a comparably higher E_{out}.

The error bound $\sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$ in (2.1), or 'error bar' if you will, depends on M, the size of the hypothesis set $\mathcal{H}$. If $\mathcal{H}$ is an infinite set, the bound goes to infinity and becomes meaningless. Unfortunately, almost all interesting learning models have infinite $\mathcal{H}$, including the simple perceptron which we discussed in Chapter 1.

In order to study generalization in such models, we need to derive a counterpart to (2.1) that deals with infinite $\mathcal{H}$. We would like to replace M with

[1] Sometimes 'generalization error' is used as another name for E_{out}, but not in this book.

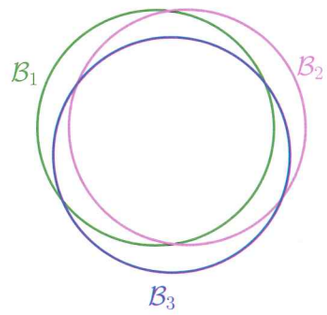
something finite, so that the bound is meaningful. To do this, we notice that the way we got the M factor in the first place was by taking the disjunction of events:

$$\begin{aligned} &\text{“}|E_{\text{in}}(h_1) - E_{\text{out}}(h_1)| > \epsilon\text{”} \ \textbf{or} \\ &\text{“}|E_{\text{in}}(h_2) - E_{\text{out}}(h_2)| > \epsilon\text{”} \ \textbf{or} \\ &\qquad\vdots \\ &\text{“}|E_{\text{in}}(h_M) - E_{\text{out}}(h_M)| > \epsilon\text{”}\ , \end{aligned} \tag{2.2}$$

which is guaranteed to include the event “$|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon$” since g is always one of the hypotheses in $\mathcal{H}$. We then over-estimated the probability using the union bound. Let $\mathcal{B}_m$ be the ($\mathcal{B}$ad) event that “$|E_{\text{in}}(h_m) - E_{\text{out}}(h_m)| > \epsilon$”. Then,

$$\mathbb{P}[\mathcal{B}_1 \textbf{ or } \mathcal{B}_2 \textbf{ or } \cdots \textbf{ or } \mathcal{B}_M] \leq \mathbb{P}[\mathcal{B}_1] + \mathbb{P}[\mathcal{B}_2] + \cdots + \mathbb{P}[\mathcal{B}_M].$$

If the events $\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_M$ are strongly overlapping, the union bound becomes particularly loose as illustrated in the figure to the right for an example with 3 hypotheses; the areas of different events correspond to their probabilities. The union bound says that the total area covered by $\mathcal{B}_1$, $\mathcal{B}_2$, or $\mathcal{B}_3$ is smaller than the sum of the individual areas, which is true but is a gross overestimate when the areas overlap heavily as in this example. The events “$|E_{\text{in}}(h_m) - E_{\text{out}}(h_m)| > \epsilon$”; $m = 1, \cdots, M$, are often strongly overlapping. If h_1 is very similar to h_2 for instance, the two events “$|E_{\text{in}}(h_1) - E_{\text{out}}(h_1)| > \epsilon$” and “$|E_{\text{in}}(h_2) - E_{\text{out}}(h_2)| > \epsilon$” are likely to coincide for most data sets. In a typical learning model, many hypotheses are indeed very similar. If you take the perceptron model for instance, as you slowly vary the weight vector $\mathbf{w}$, you get infinitely many hypotheses that differ from each other only infinitesimally.



The mathematical theory of generalization hinges on this observation. Once we properly account for the overlaps of the different hypotheses, we will be able to replace the number of hypotheses M in (2.1) by an effective number which is finite even when M is infinite, and establish a more useful condition under which E_{out} is close to E_{in}.

2.1.1 Effective Number of Hypotheses

We now introduce the *growth function*, the quantity that will formalize the effective number of hypotheses. The growth function is what will replace M

in the generalization bound (2.1). It is a combinatorial quantity that captures how different the hypotheses in $\mathcal{H}$ are, and hence how much overlap the different events in (2.2) have.

We will start by defining the growth function and studying its basic properties. Next, we will show how we can bound the value of the growth function. Finally, we will show that we can replace M in the generalization bound with the growth function. These three steps will yield the generalization bound that we need, which applies to infinite $\mathcal{H}$. We will focus on binary target functions for the purpose of this analysis, so each $h \in \mathcal{H}$ maps $\mathcal{X}$ to $\{-1, +1\}$.

The definition of the growth function is based on the number of different hypotheses that $\mathcal{H}$ can implement, but only over a finite sample of points rather than over the entire input space $\mathcal{X}$. If $h \in \mathcal{H}$ is applied to a finite sample $\mathbf{x}_1, \cdots, \mathbf{x}_N \in \mathcal{X}$, we get an N-tuple $h(\mathbf{x}_1), \cdots, h(\mathbf{x}_N)$ of ± 1's. Such an N-tuple is called a *dichotomy* since it splits $\mathbf{x}_1, \cdots, \mathbf{x}_N$ into two groups: those points for which h is -1 and those for which h is $+1$. Each $h \in \mathcal{H}$ generates a dichotomy on $\mathbf{x}_1, \cdots, \mathbf{x}_N$, but two different h's may generate the same dichotomy if they happen to give the same pattern of ± 1's on this particular sample.

Definition 2.1. *Let $\mathbf{x}_1, \cdots, \mathbf{x}_N \in \mathcal{X}$. The dichotomies generated by $\mathcal{H}$ on these points are defined by*

$$\mathcal{H}(\mathbf{x}_1, \cdots, \mathbf{x}_N) = \{ (h(\mathbf{x}_1), \cdots, h(\mathbf{x}_N)) \mid h \in \mathcal{H} \}. \tag{2.3}$$

One can think of the dichotomies $\mathcal{H}(\mathbf{x}_1, \cdots, \mathbf{x}_N)$ as a set of hypotheses just like $\mathcal{H}$ is, except that the hypotheses are seen through the eyes of N points only. A larger $\mathcal{H}(\mathbf{x}_1, \cdots, \mathbf{x}_N)$ means $\mathcal{H}$ is more 'diverse' – generating more dichotomies on $\mathbf{x}_1, \cdots, \mathbf{x}_N$. The growth function is based on the number of dichotomies.

Definition 2.2. *The growth function is defined for a hypothesis set $\mathcal{H}$ by*

$$m_{\mathcal{H}}(N) = \max_{\mathbf{x}_1, \cdots, \mathbf{x}_N \in \mathcal{X}} |\mathcal{H}(\mathbf{x}_1, \cdots, \mathbf{x}_N)|,$$

where $|\cdot|$ denotes the cardinality (number of elements) of a set.

In words, $m_{\mathcal{H}}(N)$ is the maximum number of dichotomies that can be generated by $\mathcal{H}$ on any N points. To compute $m_{\mathcal{H}}(N)$, we consider all possible choices of N points $\mathbf{x}_1, \cdots, \mathbf{x}_N$ from $\mathcal{X}$ and pick the one that gives us the most dichotomies. Like M, $m_{\mathcal{H}}(N)$ is a measure of the number of hypotheses in $\mathcal{H}$, except that a hypothesis is now considered on N points instead of the entire $\mathcal{X}$. For any $\mathcal{H}$, since $\mathcal{H}(\mathbf{x}_1, \cdots, \mathbf{x}_N) \subseteq \{-1, +1\}^N$ (the set of all possible dichotomies on any N points), the value of $m_{\mathcal{H}}(N)$ is at most $|\{-1, +1\}^N|$, hence

$$m_{\mathcal{H}}(N) \leq 2^N.$$

If $\mathcal{H}$ is capable of generating all possible dichotomies on $\mathbf{x}_1, \cdots, \mathbf{x}_N$, then $\mathcal{H}(\mathbf{x}_1, \cdots, \mathbf{x}_N) = \{-1, +1\}^N$ and we say that $\mathcal{H}$ can *shatter* $\mathbf{x}_1, \cdots, \mathbf{x}_N$. This signifies that $\mathcal{H}$ is as diverse as can be on this particular sample.

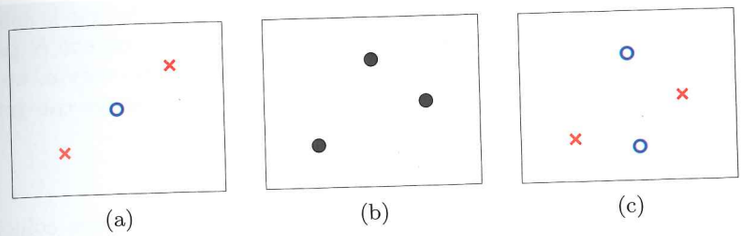


Figure 2.1: Illustration of the growth function for a two-dimensional perceptron. The dichotomy of red versus blue on the 3 colinear points in part (a) cannot be generated by a perceptron, but all 8 dichotomies on the 3 points in part (b) can. By contrast, the dichotomy of red versus blue on the 4 points in part (c) cannot be generated by a perceptron. At most 14 out of the possible 16 dichotomies on any 4 points can be generated.

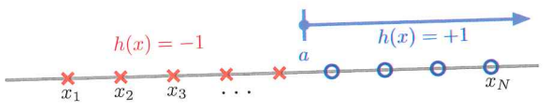
Example 2.1. If $\mathcal{X}$ is a Euclidean plane and $\mathcal{H}$ is a two-dimensional perceptron, what are $m_{\mathcal{H}}(3)$ and $m_{\mathcal{H}}(4)$? Figure 2.1(a) shows a dichotomy on 3 points that the perceptron cannot generate, while Figure 2.1(b) shows another 3 points that the perceptron can shatter, generating all $2^3 = 8$ dichotomies. Because the definition of $m_{\mathcal{H}}(N)$ is based on the maximum number of dichotomies, $m_{\mathcal{H}}(3) = 8$ in spite of the case in Figure 2.1(a).

In the case of 4 points, Figure 2.1(c) shows a dichotomy that the perceptron cannot generate. One can verify that there are no 4 points that the perceptron can shatter. The most a perceptron can do on any 4 points is 14 dichotomies out of the possible 16, where the 2 missing dichotomies are as depicted in Figure 2.1(c) with blue and red corresponding to -1, $+1$ or to $+1$, -1. Hence, $m_{\mathcal{H}}(4) = 14$. □

Let us now illustrate how to compute $m_{\mathcal{H}}(N)$ for some simple hypothesis sets. These examples will confirm the intuition that $m_{\mathcal{H}}(N)$ grows faster when the hypothesis set $\mathcal{H}$ becomes more complex. This is what we expect of a quantity that is meant to replace M in the generalization bound (2.1).

Example 2.2. Let us find a formula for $m_{\mathcal{H}}(N)$ in each of the following cases.

1. *Positive rays*: $\mathcal{H}$ consists of all hypotheses $h \colon \mathbb{R} \to \{-1, +1\}$ of the form $h(x) = \text{sign}(x - a)$, i.e., the hypotheses are defined in a one-dimensional input space, and they return -1 to the left of some value a and $+1$ to the right of a.

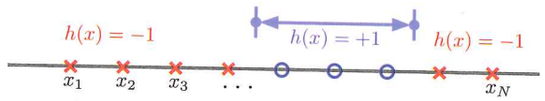


To compute $m_{\mathcal{H}}(N)$, we notice that given N points, the line is split by the points into $N+1$ regions. The dichotomy we get on the N points is decided by which region contains the value a. As we vary a, we will get $N+1$ different dichotomies. Since this is the most we can get for any N points, the growth function is

$$m_{\mathcal{H}}(N) = N + 1.$$

Notice that if we picked N points where some of the points coincided (which is allowed), we will get less than $N+1$ dichotomies. This does not affect the value of $m_{\mathcal{H}}(N)$ since it is defined based on the maximum number of dichotomies.

2. *Positive intervals*: $\mathcal{H}$ consists of all hypotheses in one dimension that return $+1$ within some interval and -1 otherwise. Each hypothesis is specified by the two end values of that interval.



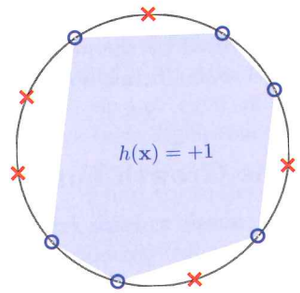
To compute $m_{\mathcal{H}}(N)$, we notice that given N points, the line is again split by the points into $N+1$ regions. The dichotomy we get is decided by which two regions contain the end values of the interval, resulting in $\binom{N+1}{2}$ different dichotomies. If both end values fall in the same region, the resulting hypothesis is the constant -1 regardless of which region it is. Adding up these possibilities, we get

$$m_{\mathcal{H}}(N) = \binom{N+1}{2} + 1 = \frac{1}{2}N^2 + \frac{1}{2}N + 1.$$

Notice that $m_{\mathcal{H}}(N)$ grows as the square of N, faster than the linear $m_{\mathcal{H}}(N)$ of the 'simpler' positive ray case.

3. *Convex sets*: $\mathcal{H}$ consists of all hypotheses in two dimensions $h\colon \mathbb{R}^2 \to \{-1, +1\}$ that are positive inside some convex set and negative elsewhere (a set is convex if the line segment connecting any two points in the set lies entirely within the set).

To compute $m_{\mathcal{H}}(N)$ in this case, we need to choose the N points carefully. Per the next figure, choose N points on the perimeter of a circle. Now consider any dichotomy on these points, assigning an arbitrary pattern of ± 1's to the N points. If you connect the $+1$ points with a polygon, the hypothesis made up of the closed interior of the polygon (which has to be convex since its vertices are on the perimeter of a circle) agrees with the dichotomy on all N points. For the dichotomies that have less than three $+1$ points, the convex set will be a line segment, a point, or an empty set.



This means that any dichotomy on these N points can be realized using a convex hypothesis, so $\mathcal{H}$ manages to shatter these points and the growth function has the maximum possible value

$$m_{\mathcal{H}}(N) = 2^N.$$

Notice that if the N points were chosen at random in the plane rather than on the perimeter of a circle, many of the points would be 'internal' and we wouldn't be able to shatter all the points with convex hypotheses as we did for the perimeter points. However, this doesn't matter as far as $m_{\mathcal{H}}(N)$ is concerned, since it is defined based on the maximum (2^N in this case). □

It is not practical to try to compute $m_{\mathcal{H}}(N)$ for every hypothesis set we use. Fortunately, we don't have to. Since $m_{\mathcal{H}}(N)$ is meant to replace M in (2.1), we can use an upper bound on $m_{\mathcal{H}}(N)$ instead of the exact value, and the inequality in (2.1) will still hold. Getting a good bound on $m_{\mathcal{H}}(N)$ will prove much easier than computing $m_{\mathcal{H}}(N)$ itself, thanks to the notion of a *break point*.

Definition 2.3. *If no data set of size k can be shattered by $\mathcal{H}$, then k is said to be a break point for $\mathcal{H}$.*

If k is a break point, then $m_{\mathcal{H}}(k) < 2^k$. Example 2.1 shows that $k = 4$ is a break point for two-dimensional perceptrons. In general, it is easier to find a break point for $\mathcal{H}$ than to compute the full growth function for that $\mathcal{H}$.

Exercise 2.1

By inspection, find a break point k for each hypothesis set in Example 2.2 (if there is one). Verify that $m_{\mathcal{H}}(k) < 2^k$ using the formulas derived in that Example.

We now use the break point k to derive a bound on the growth function $m_{\mathcal{H}}(N)$ for all values of N. For example, the fact that no 4 points can be shattered by

the two-dimensional perceptron puts a significant constraint on the number of dichotomies that can be realized by the perceptron on 5 or more points. We will exploit this idea to get a significant bound on $m_{\mathcal{H}}(N)$ in general.

2.1.2 Bounding the Growth Function

The most important fact about growth functions is that if the condition $m_{\mathcal{H}}(N) = 2^N$ breaks at any point, we can bound $m_{\mathcal{H}}(N)$ for all values of N by a simple polynomial based on this break point. The fact that the bound is polynomial is crucial. Absent a break point (as is the case in the convex hypothesis example), $m_{\mathcal{H}}(N) = 2^N$ for all N. If $m_{\mathcal{H}}(N)$ replaced M in Equation (2.1), the bound $\sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$ on the generalization error would not go to zero regardless of how many training examples N we have. However, if $m_{\mathcal{H}}(N)$ can be bounded by a polynomial – any polynomial –, the generalization error will go to zero as $N \to \infty$. This means that we will generalize well given a sufficient number of examples.

Begin safe skip: If you trust our math, you can skip the following part without compromising the logical sequence. A similar green box will tell you when to rejoin.

To prove the polynomial bound, we will introduce a combinatorial quantity that counts the maximum number of dichotomies given that there is a break point, without having to assume any particular form of $\mathcal{H}$. This bound will therefore apply to any $\mathcal{H}$.

Definition 2.4. *$B(N, k)$ is the maximum number of dichotomies on N points such that no subset of size k of the N points can be shattered by these dichotomies.*

The definition of $B(N, k)$ assumes a break point k, then tries to find the most dichotomies on N points without imposing any further restrictions. Since $B(N, k)$ is defined as a maximum, it will serve as an upper bound for any $m_{\mathcal{H}}(N)$ that has a break point k;

$$m_{\mathcal{H}}(N) \leq B(N, k) \quad \text{if } k \text{ is a break point for } \mathcal{H}.$$

The notation B comes from 'Binomial' and the reason will become clear shortly. To evaluate $B(N, k)$, we start with the two boundary conditions $k = 1$ and $N = 1$.

$$\begin{aligned} B(N, 1) &= 1 \\ B(1, k) &= 2 \text{ for } k > 1 \,. \end{aligned}$$

$B(N, 1) = 1$ for all N since if no subset of size 1 can be shattered, then only one dichotomy can be allowed. A second different dichotomy must differ on at least one point and then that subset of size 1 would be shattered. $B(1, k) = 2$ for $k > 1$ since in this case there do not even exist subsets of size k; the constraint is vacuously true and we have 2 possible dichotomies ($+1$ and -1) on the one point.

We now assume $N \geq 2$ and $k \geq 2$ and try to develop a recursion. Consider the $B(N, k)$ dichotomies in definition 2.4, where no k points can be shattered. We list these dichotomies in the following table,

	# of rows	$\mathbf{x}_1$	$\mathbf{x}_2$	$\cdots$	$\mathbf{x}_{N-1}$	$\mathbf{x}_N$
S_1	α	$+1$	$+1$	$\cdots$	$+1$	$+1$
		-1	$+1$	$\cdots$	$+1$	-1
		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
		$+1$	-1	$\cdots$	-1	-1
		-1	$+1$	$\cdots$	-1	$+1$
S_2	β	$+1$	-1	$\cdots$	$+1$	$+1$
S_2^+		-1	-1	$\cdots$	$+1$	$+1$
		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
		$+1$	-1	$\cdots$	$+1$	$+1$
		-1	-1	$\cdots$	-1	$+1$
	β	$+1$	-1	$\cdots$	$+1$	-1
S_2^-		-1	-1	$\cdots$	$+1$	-1
		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
		$+1$	-1	$\cdots$	$+1$	-1
		-1	-1	$\cdots$	-1	-1

where $\mathbf{x}_1, \cdots, \mathbf{x}_N$ in the table are labels for the N points of the dichotomy. We have chosen a convenient order in which to list the dichotomies, as follows. Consider the dichotomies on $\mathbf{x}_1, \cdots, \mathbf{x}_{N-1}$. Some dichotomies on these $N-1$ points appear only once (with either $+1$ or -1 in the $\mathbf{x}_N$ column, but not both). We collect these dichotomies in the set S_1. The remaining dichotomies on the first $N-1$ points appear twice, once with $+1$ and once with -1 in the $\mathbf{x}_N$ column. We collect these dichotomies in the set S_2 which can be divided into two equal parts, S_2^+ and S_2^- (with $+1$ and -1 in the $\mathbf{x}_N$ column, respectively). Let S_1 have α rows, and let S_2^+ and S_2^- have β rows each. Since the total number of rows in the table is $B(N, k)$ by construction, we have

$$B(N, k) = \alpha + 2\beta. \tag{2.4}$$

The total number of different dichotomies on the first $N-1$ points is given by $\alpha + \beta$; since S_2^+ and S_2^- are identical on these $N-1$ points, their dichotomies are redundant. Since no subset of k of these first $N-1$ points can

be shattered (since no k-subset of all N points can be shattered), we deduce that

$$\alpha + \beta \leq B(N-1, k) \tag{2.5}$$

by definition of B. Further, no subset of size $k-1$ of the first $N-1$ points can be shattered by the dichotomies in S_2^+. If there existed such a subset, then taking the corresponding set of dichotomies in S_2^- and adding $\mathbf{x}_N$ to the data points yields a subset of size k that is shattered, which we know cannot exist in this table by definition of $B(N,k)$. Therefore,

$$\beta \leq B(N-1, k-1). \tag{2.6}$$

Substituting the two Inequalities (2.5) and (2.6) into (2.4), we get

$$B(N,k) \leq B(N-1,k) + B(N-1,k-1). \tag{2.7}$$

We can use (2.7) to recursively compute a bound on $B(N,k)$, as shown in the following table.

	k						
N	1	2	3	4	5	6	...
1	1	2	2	2	2	2	...
2	1	3	4	4	4	4	...
3	1	4	7	8	8	8	...
4	1	5	11	...	...	...	...
5	1	6	⋮	⋱			
6	1	7	⋮		⋱		
⋮	⋮	⋮	⋮			⋱	

where the first row ($N = 1$) and the first column ($k = 1$) are the boundary conditions that we already calculated. We can also use the recursion to bound $B(N,k)$ analytically.

Lemma 2.3 (Sauer's Lemma)**.**

$$B(N,k) \leq \sum_{i=0}^{k-1} \binom{N}{i}$$

Proof. The statement is true whenever $k = 1$ or $N = 1$, by inspection. The proof is by induction on N. Assume the statement is true for all $N \leq N_o$ and all k. We need to prove the statement for $N = N_o + 1$ and all k. Since the statement is already true when $k = 1$ (for all values of N) by the initial condition, we only need to worry about $k \geq 2$. By (2.7),

$$B(N_o+1, k) \leq B(N_o, k) + B(N_o, k-1).$$

Applying the induction hypothesis to each term on the RHS, we get

$$\begin{aligned} B(N_o+1,k) &\leq \sum_{i=0}^{k-1} \binom{N_o}{i} + \sum_{i=0}^{k-2} \binom{N_o}{i} \\ &= 1 + \sum_{i=1}^{k-1} \binom{N_o}{i} + \sum_{i=1}^{k-1} \binom{N_o}{i-1} \\ &= 1 + \sum_{i=1}^{k-1} \left[\binom{N_o}{i} + \binom{N_o}{i-1} \right] \\ &= 1 + \sum_{i=1}^{k-1} \binom{N_o+1}{i} = \sum_{i=0}^{k-1} \binom{N_o+1}{i}, \end{aligned}$$

where the combinatorial identity $\binom{N_o+1}{i} = \binom{N_o}{i} + \binom{N_o}{i-1}$ has been used. This identity can be proved by noticing that to calculate the number of ways to pick i objects from N_o+1 distinct objects, either the first object is included, in $\binom{N_o}{i-1}$ ways, or the first object is not included, in $\binom{N_o}{i}$ ways. We have thus proved the induction step, so the statement is true for all N and k. ■

It turns out that $B(N,k)$ in fact equals $\sum_{i=0}^{k-1} \binom{N}{i}$ (see Problem 2.4), but we only need the inequality of Lemma 2.3 to bound the growth function. For a given break point k, the bound $\sum_{i=0}^{k-1} \binom{N}{i}$ is polynomial in N, as each term in the sum is polynomial (of degree $i \leq k-1$). Since $B(N,k)$ is an upper bound on any $m_{\mathcal{H}}(N)$ that has a break point k, we have proved

End safe skip: Those who skipped are now rejoining us. The next theorem states that any growth function $m_{\mathcal{H}}(N)$ with a break point is bounded by a polynomial.

Theorem 2.4. If $m_{\mathcal{H}}(k) < 2^k$ for some value k, then

$$m_{\mathcal{H}}(N) \leq \sum_{i=0}^{k-1} \binom{N}{i} \tag{2.8}$$

for all N. The RHS is polynomial in N of degree $k-1$.

The implication of Theorem 2.4 is that if $\mathcal{H}$ has a break point, we have what we want to ensure good generalization; a polynomial bound on $m_{\mathcal{H}}(N)$.

Exercise 2.2

(a) Verify the bound of Theorem 2.4 in the three cases of Example 2.2:

(i) Positive rays: $\mathcal{H}$ consists of all hypotheses in one dimension of the form $h(x) = \text{sign}(x - a)$.

(ii) Positive intervals: $\mathcal{H}$ consists of all hypotheses in one dimension that are positive within some interval and negative elsewhere.

(iii) Convex sets: $\mathcal{H}$ consists of all hypotheses in two dimensions that are positive inside some convex set and negative elsewhere.

(Note: you can use the break points you found in Exercise 2.1.)

(b) Does there exist a hypothesis set for which $m_{\mathcal{H}}(N) = N + 2^{\lfloor N/2 \rfloor}$ (where $\lfloor N/2 \rfloor$ is the largest integer $\leq N/2$)?

2.1.3 The VC Dimension

Theorem 2.4 bounds the entire growth function in terms of any break point. The smaller the break point, the better the bound. This leads us to the following definition of a single parameter that characterizes the growth function.

Definition 2.5. *The Vapnik-Chervonenkis dimension of a hypothesis set $\mathcal{H}$, denoted by $d_{\text{VC}}(\mathcal{H})$ or simply d_{VC}, is the largest value of N for which $m_{\mathcal{H}}(N) = 2^N$. If $m_{\mathcal{H}}(N) = 2^N$ for all N, then $d_{\text{VC}}(\mathcal{H}) = \infty$.*

If d_{VC} is the VC dimension of $\mathcal{H}$, then $k = d_{\text{VC}} + 1$ is a break point for $m_{\mathcal{H}}$ since $m_{\mathcal{H}}(N)$ cannot equal 2^N for any $N > d_{\text{VC}}$ by definition. It is easy to see that no smaller break point exists since $\mathcal{H}$ can shatter d_{VC} points, hence it can also shatter any subset of these points.

Exercise 2.3

Compute the VC dimension of $\mathcal{H}$ for the hypothesis sets in parts (i), (ii), (iii) of Exercise 2.2(a).

Since $k = d_{\text{VC}} + 1$ is a break point for $m_{\mathcal{H}}$, Theorem 2.4 can be rewritten in terms of the VC dimension:

$$m_{\mathcal{H}}(N) \leq \sum_{i=0}^{d_{\text{VC}}} \binom{N}{i}. \tag{2.9}$$

Therefore, the VC dimension is the order of the polynomial bound on $m_{\mathcal{H}}(N)$. It is also the best we can do using this line of reasoning, because no smaller break point than $k = d_{\text{VC}} + 1$ exists. The form of the polynomial bound can be further simplified to make the dependency on d_{VC} more salient. We state a useful form here, which can be proved by induction (Problem 2.5).

$$m_{\mathcal{H}}(N) \leq N^{d_{\text{VC}}} + 1. \tag{2.10}$$

Now that the growth function has been bounded in terms of the VC dimension, we have only one more step left in our analysis, which is to replace the number of hypotheses M in the generalization bound (2.1) with the growth function $m_{\mathcal{H}}(N)$. If we manage to do that, the VC dimension will play a pivotal role in the generalization question. If we were to directly replace M by $m_{\mathcal{H}}(N)$ in (2.1), we would get a bound of the form

$$E_{\text{out}} \overset{?}{\leq} E_{\text{in}} + \sqrt{\frac{1}{2N} \ln \frac{2m_{\mathcal{H}}(N)}{\delta}}.$$

Unless $d_{\text{VC}}(\mathcal{H}) = \infty$, we know that $m_{\mathcal{H}}(N)$ is bounded by a polynomial in N; thus, $\ln m_{\mathcal{H}}(N)$ grows logarithmically in N regardless of the order of the polynomial, and so it will be crushed by the $\frac{1}{N}$ factor. Therefore, for any fixed tolerance δ, the bound on E_{out} will be arbitrarily close to E_{in} for sufficiently large N.

Only if $d_{\text{VC}}(\mathcal{H}) = \infty$ will this argument fail, as the growth function in this case is exponential in N. For any finite value of d_{VC}, the error bar will converge to zero at a speed determined by d_{VC}, since d_{VC} is the order of the polynomial. The smaller d_{VC} is, the faster the convergence to zero.

It turns out that we cannot just replace M with $m_{\mathcal{H}}(N)$ in the generalization bound (2.1), but rather we need to make other adjustments as we will see shortly. However, the general idea above is correct, and d_{VC} will still play the role that we discussed here. One implication of this discussion is that there is a division of models into two classes. The 'good models' have finite d_{VC}, and for sufficiently large N, E_{in} will be close to E_{out}; for good models, the in-sample performance generalizes to out of sample. The 'bad models' have infinite d_{VC}. With a bad model, no matter how large the data set is, we cannot make generalization conclusions from E_{in} to E_{out} based on the VC analysis.[2]

Because of its significant role, it is worthwhile to try to gain some insight about the VC dimension before we proceed to the formalities of deriving the new generalization bound. One way to gain insight about d_{VC} is to try to compute it for learning models that we are familiar with. Perceptrons are one case where we can compute d_{VC} exactly. This is done in two steps. First, we show that d_{VC} is at least a certain value, then we show that it is at most the same value. There is a logical difference in arguing that d_{VC} is at least a certain value, as opposed to at most a certain value. This is because

$$d_{\text{VC}} \geq N \iff \text{there } \textbf{exists } \mathcal{D} \text{ of size } N \text{ such that } \mathcal{H} \text{ shatters } \mathcal{D},$$

hence we have different conclusions in the following cases.

1. There is a set of N points that can be shattered by $\mathcal{H}$. In this case, we can conclude that $d_{\text{VC}} \geq N$.

[2] In some cases with infinite d_{VC}, such as the convex sets that we discussed, alternative analysis based on an 'average' growth function can establish good generalization behavior.

2. Any set of N points can be shattered by $\mathcal{H}$. In this case, we have more than enough information to conclude that $d_{\text{vc}} \geq N$.

3. There is a set of N points that cannot be shattered by $\mathcal{H}$. Based only on this information, we cannot conclude anything about the value of d_{vc}.

4. No set of N points can be shattered by $\mathcal{H}$. In this case, we can conclude that $d_{\text{vc}} < N$.

Exercise 2.4

Consider the input space $\mathcal{X} = \{1\} \times \mathbb{R}^d$ (including the constant coordinate $x_0 = 1$). Show that the VC dimension of the perceptron (with $d+1$ parameters, counting w_0) is exactly $d+1$ by showing that it is at least $d+1$ and at most $d+1$, as follows.

(a) To show that $d_{\text{vc}} \geq d+1$, find $d+1$ points in $\mathcal{X}$ that the perceptron can shatter. *[Hint: Construct a nonsingular $(d+1) \times (d+1)$ matrix whose rows represent the $d+1$ points, then use the nonsingularity to argue that the perceptron can shatter these points.]*

(b) To show that $d_{\text{vc}} \leq d+1$, show that no set of $d+2$ points in $\mathcal{X}$ can be shattered by the perceptron. *[Hint: Represent each point in $\mathcal{X}$ as a vector of length $d+1$, then use the fact that any $d+2$ vectors of length $d+1$ have to be linearly dependent. This means that some vector is a linear combination of all the other vectors. Now, if you choose the class of these other vectors carefully, then the classification of the dependent vector will be dictated. Conclude that there is some dichotomy that cannot be implemented, and therefore that for $N \geq d+2$, $m_{\mathcal{H}}(N) < 2^N$.]*

The VC dimension of a d-dimensional perceptron[3] is indeed $d+1$. This is consistent with Figure 2.1 for the case $d = 2$, which shows a VC dimension of 3. The perceptron case provides a nice intuition about the VC dimension, since $d+1$ is also the number of parameters in this model. One can view the VC dimension as measuring the 'effective' number of parameters. The more parameters a model has, the more diverse its hypothesis set is, which is reflected in a larger value of the growth function $m_{\mathcal{H}}(N)$. In the case of perceptrons, the effective parameters correspond to explicit parameters in the model, namely $w_0, w_1, \cdots, w_d$. In other models, the effective parameters may be less obvious or implicit. The VC dimension measures these effective parameters or 'degrees of freedom' that enable the model to express a diverse set of hypotheses.

Diversity is not necessarily a good thing in the context of generalization. For example, the set of all possible hypotheses is as diverse as can be, so $m_{\mathcal{H}}(N) = 2^N$ for all N and $d_{\text{vc}}(\mathcal{H}) = \infty$. In this case, no generalization at all is to be expected, as the final version of the generalization bound will show.

[3] $\mathcal{X} = \{1\} \times \mathbb{R}^d$ is considered d-dimensional since the first coordinate $x_0 = 1$ is fixed.

2.1.4 The VC Generalization Bound

If we treated the growth function as an effective number of hypotheses, and replaced M in the generalization bound (2.1) with $m_{\mathcal{H}}(N)$, the resulting bound would be

$$E_{\text{out}}(g) \overset{?}{\leq} E_{\text{in}}(g) + \sqrt{\frac{1}{2N} \ln \frac{2m_{\mathcal{H}}(N)}{\delta}}. \tag{2.11}$$

It turns out that this is not exactly the form that will hold. The quantities in red need to be technically modified to make (2.11) true. The correct bound, which is called the VC generalization bound, is given in the following theorem; it holds for any binary target function f, any hypothesis set $\mathcal{H}$, any learning algorithm $\mathcal{A}$, and any input probability distribution P.

Theorem 2.5 (VC generalization bound)**.** For any tolerance $\delta > 0$,

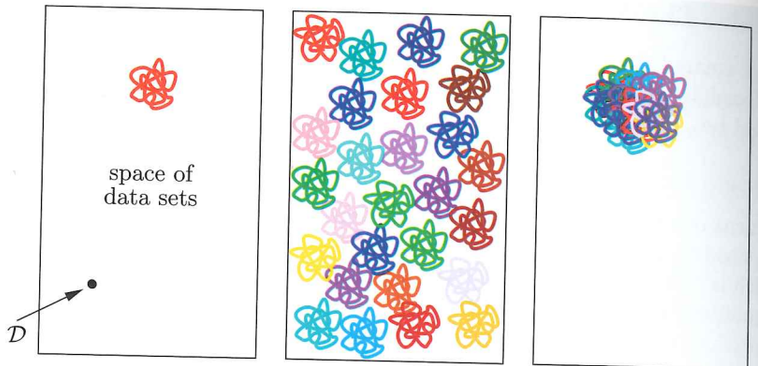
$$E_{\text{out}}(g) \leq E_{\text{in}}(g) + \sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}} \tag{2.12}$$

with probability $\geq 1 - \delta$.

If you compare the blue items in (2.12) to their red counterparts in (2.11), you notice that all the blue items move the bound in the weaker direction. However, as long as the VC dimension is finite, the error bar still converges to zero (albeit at a slower rate), since $m_{\mathcal{H}}(2N)$ is also polynomial of order d_{vc} in N, just like $m_{\mathcal{H}}(N)$. This means that, with enough data, *each and every hypothesis* in an infinite $\mathcal{H}$ with a finite VC dimension will generalize well from E_{in} to E_{out}. The key is that the effective number of hypotheses, represented by the finite growth function, has replaced the actual number of hypotheses in the bound.

The VC generalization bound is the most important mathematical result in the theory of learning. It establishes the feasibility of learning with infinite hypothesis sets. Since the formal proof is somewhat lengthy and technical, we illustrate the main ideas in a sketch of the proof, and include the formal proof as an appendix. There are two parts to the proof; the justification that the growth function can replace the number of hypotheses in the first place, and the reason why we had to change the red items in (2.11) into the blue items in (2.12).

Sketch of the proof. The data set $\mathcal{D}$ is the source of randomization in the original Hoeffding Inequality. Consider the space of all possible data sets. Let us think of this space as a 'canvas' (Figure 2.2(a)). Each $\mathcal{D}$ is a point on that canvas. The probability of a point is determined by which $\mathbf{x}_n$'s in $\mathcal{X}$ happen to be in that particular $\mathcal{D}$, and is calculated based on the distribution P over $\mathcal{X}$. Let's think of probabilities of different events as areas on that canvas, so the total area of the canvas is 1.



(a) Hoeffding Inequality (b) Union Bound (c) VC Bound

Figure 2.2: Illustration of the proof of the VC bound, where the 'canvas' represents the space of all data sets, with areas corresponding to probabilities. (a) For a given hypothesis, the colored points correspond to data sets where E_{in} does not generalize well to E_{out}. The Hoeffding Inequality guarantees a small colored area. (b) For several hypotheses, the union bound assumes no overlaps, so the total colored area is large. (c) The VC bound keeps track of overlaps, so it estimates the total area of bad generalization to be relatively small.

For a given hypothesis $h \in \mathcal{H}$, the event "$|E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon$" consists of all points $\mathcal{D}$ for which the statement is true. For a particular h, let us paint all these 'bad' points using one color. What the basic Hoeffding Inequality tells us is that the colored area on the canvas will be small (Figure 2.2(a)).

Now, if we take another $h \in \mathcal{H}$, the event "$|E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon$" may contain different points, since the event depends on h. Let us paint these points with a different color. The area covered by all the points we colored will be at most the sum of the two individual areas, which is the case only if the two areas have no points in common. This is the worst case that the union bound considers. If we keep throwing in a new colored area for each $h \in \mathcal{H}$, and never overlap with previous colors, the canvas will soon be mostly covered in color (Figure 2.2(b)). Even if each h contributed very little, the sheer number of hypotheses will eventually make the colored area cover the whole canvas. This was the problem with using the union bound in the Hoeffding Inequality (1.6), and not taking the overlaps of the colored areas into consideration.

The bulk of the VC proof deals with how to account for the overlaps. Here is the idea. If you were told that the hypotheses in $\mathcal{H}$ are such that each point on the canvas that is colored will be colored 100 times (because of 100 different h's), then the total colored area is now 1/100 of what it would have been if the colored points had not overlapped at all. This is the essence of the VC bound as illustrated in (Figure 2.2(c)). The argument goes as follows.

Many hypotheses share the same dichotomy on a given $\mathcal{D}$, since there are finitely many dichotomies even with an infinite number of hypotheses. Any statement based on $\mathcal{D}$ alone will be simultaneously true or simultaneously false for all the hypotheses that look the same on that particular $\mathcal{D}$. What the growth function enables us to do is to account for this kind of hypothesis redundancy in a precise way, so we can get a factor similar to the '100' in the above example.

When $\mathcal{H}$ is infinite, the redundancy factor will also be infinite since the hypotheses will be divided among a finite number of dichotomies. Therefore, the reduction in the total colored area when we take the redundancy into consideration will be dramatic. If it happens that the number of dichotomies is only a polynomial, the reduction will be so dramatic as to bring the total probability down to a very small value. This is the essence of the proof of Theorem 2.5.

The reason $m_{\mathcal{H}}(2N)$ appears in the VC bound instead of $m_{\mathcal{H}}(N)$ is that the proof uses a sample of $2N$ points instead of N points. Why do we need $2N$ points? The event "$|E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon$" depends not only on $\mathcal{D}$, but also on the entire $\mathcal{X}$ because $E_{\text{out}}(h)$ is based on $\mathcal{X}$. This breaks the main premise of grouping h's based on their behavior on $\mathcal{D}$, since aspects of each h outside of $\mathcal{D}$ affect the truth of "$|E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon$." To remedy that, we consider the artificial event "$|E_{\text{in}}(h) - E'_{\text{in}}(h)| > \epsilon$" instead, where E_{in} and E'_{in} are based on two samples $\mathcal{D}$ and $\mathcal{D}'$ each of size N. This is where the $2N$ comes from. It accounts for the total size of the two samples $\mathcal{D}$ and $\mathcal{D}'$. Now, the truth of the statement "$|E_{\text{in}}(h) - E'_{\text{in}}(h)| > \epsilon$" depends exclusively on the total sample of size $2N$, and the above redundancy argument will hold.

Of course we have to justify why the two-sample condition "$|E_{\text{in}}(h) - E'_{\text{in}}(h)| > \epsilon$" can replace the original condition "$|E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon$." In doing so, we end up having to shrink the ϵ's by a factor of 4, and also end up with a factor of 2 in the estimate of the overall probability. This accounts for the $\frac{8}{N}$ instead of $\frac{1}{2N}$ in the VC bound and for having 4 instead of 2 as the multiplicative factor of the growth function. When you put all this together, you get the formula in (2.12). □

2.2 Interpreting the Generalization Bound

The VC generalization bound (2.12) is a universal result in the sense that it applies to all hypothesis sets, learning algorithms, input spaces, probability distributions, and binary target functions. It can be extended to other types of target functions as well. Given the generality of the result, one would suspect that the bound it provides may not be particularly tight in any given case, since the same bound has to cover a lot of different cases. Indeed, the bound is quite loose.

Exercise 2.5

Suppose we have a simple learning model whose growth function is $m_{\mathcal{H}}(N) = N + 1$, hence $d_{\text{vc}} = 1$. Use the VC bound (2.12) to estimate the probability that E_{out} will be within 0.1 of E_{in} given 100 training examples. *[Hint: The estimate will be ridiculous.]*

Why is the VC bound so loose? The slack in the bound can be attributed to a number of technical factors. Among them,

1. The basic Hoeffding Inequality used in the proof already has a slack. The inequality gives the same bound whether E_{out} is close to 0.5 or close to zero. However, the variance of E_{in} is quite different in these two cases. Therefore, having one bound capture both cases will result in some slack.

2. Using $m_{\mathcal{H}}(N)$ to quantify the number of dichotomies on N points, regardless of which N points are in the data set, gives us a worst-case estimate. This does allow the bound to be independent of the probability distribution P over $\mathcal{X}$. However, we would get a more tuned bound if we considered specific $\mathbf{x}_1, \cdots, \mathbf{x}_N$ and used $|\mathcal{H}(\mathbf{x}_1, \cdots, \mathbf{x}_N)|$ or its expected value instead of the upper bound $m_{\mathcal{H}}(N)$. For instance, in the case of convex sets in two dimensions, which we examined in Example 2.2, if you pick N points at random in the plane, they will likely have far fewer dichotomies than 2^N, while $m_{\mathcal{H}}(N) = 2^N$.

3. Bounding $m_{\mathcal{H}}(N)$ by a simple polynomial of order d_{vc}, as given in (2.10), will contribute further slack to the VC bound.

Some effort could be put into tightening the VC bound, but many highly technical attempts in the literature have resulted in only diminishing returns. The reality is that the VC line of analysis leads to a very loose bound. Why did we bother to go through the analysis then? Two reasons. First, the VC analysis is what establishes the feasibility of learning for infinite hypothesis sets, the only kind we use in practice. Second, although the bound is loose, it tends to be equally loose for different learning models, and hence is useful for comparing the generalization performance of these models. This is an observation from practical experience, not a mathematical statement. In real applications, learning models with lower d_{vc} tend to generalize better than those with higher d_{vc}. Because of this observation, the VC analysis proves useful in practice, and some rules of thumb have emerged in terms of the VC dimension. For instance, requiring that N be at least $10 \times d_{\text{vc}}$ to get decent generalization is a popular rule of thumb.

Thus, the VC bound can be used as a guideline for generalization, relatively if not absolutely. With this understanding, let us look at the different ways the bound is used in practice.

2.2.1 Sample Complexity

The sample complexity denotes how many training examples N are needed to achieve a certain generalization performance. The performance is specified by two parameters, ϵ and δ. The error tolerance ϵ determines the allowed generalization error, and the confidence parameter δ determines how often the error tolerance ϵ is violated. How fast N grows as ϵ and δ become smaller[4] indicates how much data is needed to get good generalization.

We can use the VC bound to estimate the sample complexity for a given learning model. Fix $\delta > 0$, and suppose we want the generalization error to be at most ϵ. From Equation (2.12), the generalization error is bounded by $\sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}}$, and so it suffices to make $\sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}} \leq \epsilon$. It follows that

$$N \geq \frac{8}{\epsilon^2} \ln \left(\frac{4m_{\mathcal{H}}(2N)}{\delta} \right)$$

suffices to obtain generalization error at most ϵ (with probability at least $1-\delta$). This gives an implicit bound for the sample complexity N, since N appears on both sides of the inequality. If we replace $m_{\mathcal{H}}(2N)$ in (2.12) by its polynomial upper bound in (2.10) which is based on the the VC dimension, we get a similar bound

$$N \geq \frac{8}{\epsilon^2} \ln \left(\frac{4((2N)^{d_{\text{vc}}} + 1)}{\delta} \right), \tag{2.13}$$

which is again implicit in N. We can obtain a numerical value for N using simple iterative methods.

Example 2.6. Suppose that we have a learning model with $d_{\text{vc}} = 3$ and would like the generalization error to be at most 0.1 with confidence 90% (so $\epsilon = 0.1$ and $\delta = 0.1$). How big a data set do we need? Using (2.13), we need

$$N \geq \frac{8}{0.1^2} \ln \left(\frac{4(2N)^3 + 4}{0.1} \right).$$

Trying an initial guess of $N = 1,000$ in the RHS, we get

$$N \geq \frac{8}{0.1^2} \ln \left(\frac{4(2 \times 1000)^3 + 4}{0.1} \right) \approx 21,193.$$

We then try the new value $N = 21,193$ in the RHS and continue this iterative process, rapidly converging to an estimate of $N \approx 30,000$. If d_{vc} were 4, a similar calculation will find that $N \approx 40,000$. For $d_{\text{vc}} = 5$, we get $N \approx 50,000$. You can see that the inequality suggests that the number of examples needed is approximately proportional to the VC dimension, as has been observed in practice. The constant of proportionality it suggests is 10,000, which is a *gross* overestimate; a more practical constant of proportionality is closer to 10. □

[4] The term 'complexity' comes from a similar metaphor in computational complexity.

2.2.2 Penalty for Model Complexity

Sample complexity fixes the performance parameters ϵ (generalization error) and δ (confidence parameter) and estimates how many examples N are needed. In most practical situations, however, we are given a fixed data set $\mathcal{D}$, so N is also fixed. In this case, the relevant question is what performance can we expect given this particular N. The bound in (2.12) answers this question: with probability at least $1-\delta$,

$$E_{\text{out}}(g) \leq E_{\text{in}}(g) + \sqrt{\frac{8}{N}\ln\left(\frac{4m_{\mathcal{H}}(2N)}{\delta}\right)}.$$

If we use the polynomial bound based on d_{vc} instead of $m_{\mathcal{H}}(2N)$, we get another valid bound on the out-of-sample error,

$$E_{\text{out}}(g) \leq E_{\text{in}}(g) + \sqrt{\frac{8}{N}\ln\left(\frac{4((2N)^{d_{\text{vc}}}+1)}{\delta}\right)}. \tag{2.14}$$

Example 2.7. Suppose that $N = 100$ and we have a 90% confidence requirement ($\delta = 0.1$). We could ask what error bar can we offer with this confidence, if $\mathcal{H}$ has $d_{\text{vc}} = 1$. Using (2.14), we have

$$E_{\text{out}}(g) \leq E_{\text{in}}(g) + \sqrt{\frac{8}{100}\ln\left(\frac{4(201)}{0.1}\right)} \approx E_{\text{in}}(g) + 0.848 \tag{2.15}$$

with confidence $\geq 90\%$. This is a pretty poor bound on E_{out}. Even if $E_{\text{in}} = 0$, E_{out} may still be close to 1. If $N = 1,000$, then we get $E_{\text{out}}(g) \leq E_{\text{in}}(g)+0.301$, a somewhat more respectable bound. □

Let us look more closely at the two parts that make up the bound on E_{out} in (2.12). The first part is E_{in}, and the second part is a term that increases as the VC dimension of $\mathcal{H}$ increases.

$$E_{\text{out}}(g) \leq E_{\text{in}}(g) + \Omega(N, \mathcal{H}, \delta), \tag{2.16}$$

where

$$\begin{aligned}\Omega(N, \mathcal{H}, \delta) &= \sqrt{\frac{8}{N}\ln\left(\frac{4m_{\mathcal{H}}(2N)}{\delta}\right)} \\ &\leq \sqrt{\frac{8}{N}\ln\left(\frac{4((2N)^{d_{\text{vc}}}+1)}{\delta}\right)}.\end{aligned}$$

One way to think of $\Omega(N, \mathcal{H}, \delta)$ is that it is a penalty for model complexity. It penalizes us by worsening the bound on E_{out} when we use a more complex $\mathcal{H}$ (larger d_{vc}). If someone manages to fit a simpler model with the same training

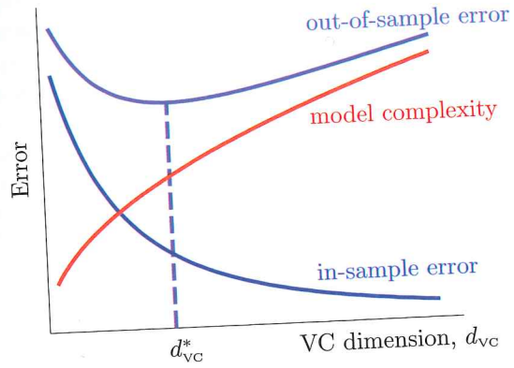


Figure 2.3: When we use a more complex learning model, one that has higher VC dimension d_{vc}, we are likely to fit the training data better resulting in a lower in-sample error, but we pay a higher penalty for model complexity. A combination of the two, which estimates the out-of-sample error, thus attains a minimum at some intermediate d^*_{vc}.

error, they will get a more favorable estimate for E_{out}. The penalty $\Omega(N, \mathcal{H}, \delta)$ gets worse if we insist on higher confidence (lower δ), and it gets better when we have more training examples, as we would expect.

Although $\Omega(N, \mathcal{H}, \delta)$ goes up when $\mathcal{H}$ has a higher VC dimension, E_{in} is likely to go down with a higher VC dimension as we have more choices within $\mathcal{H}$ to fit the data. Therefore, we have a tradeoff: more complex models help E_{in} and hurt $\Omega(N, \mathcal{H}, \delta)$. The optimal model is a compromise that minimizes a combination of the two terms, as illustrated informally in Figure 2.3.

2.2.3 The Test Set

As we have seen, the generalization bound gives us a loose estimate of the out-of-sample error E_{out} based on E_{in}. While the estimate can be useful as a guideline for the training process, it is next to useless if the goal is to get an accurate forecast of E_{out}. If you are developing a system for a customer, you need a more accurate estimate so that your customer knows how well the system is expected to perform.

An alternative approach that we alluded to in the beginning of this chapter is to estimate E_{out} by using a *test set*, a data set that was not involved in the training process. The final hypothesis g is evaluated on the test set, and the result is taken as an estimate of E_{out}. We would like to now take a closer look at this approach.

Let us call the error we get on the test set E_{test}. When we report E_{test} as our estimate of E_{out}, we are in fact asserting that E_{test} generalizes very well to E_{out}. After all, E_{test} is just a sample estimate like E_{in}. How do we know

that E_{test} generalizes well? We can answer this question with authority now that we have developed the theory of generalization in concrete mathematical terms.

The effective number of hypotheses that matters in the generalization behavior of E_{test} is 1. There is only one hypothesis as far as the test set is concerned, and that's the final hypothesis g that the training phase produced. This hypothesis would not change if we used a different test set as it would if we used a different training set. Therefore, the simple Hoeffding Inequality is valid in the case of a test set. Had the choice of g been affected by the test set in any shape or form, it wouldn't be considered a *test* set any more and the simple Hoeffding Inequality would not apply.

Therefore, the generalization bound that applies to E_{test} is the simple Hoeffding Inequality with one hypothesis. This is a much tighter bound than the VC bound. For example, if you have $1,000$ data points in the test set, E_{test} will be within $\pm 5\%$ of E_{out} with probability $\geq 98\%$. The bigger the test set you use, the more accurate E_{test} will be as an estimate of E_{out}.

Exercise 2.6

A data set has 600 examples. To properly test the performance of the final hypothesis, you set aside a randomly selected subset of 200 examples which are never used in the training phase; these form a test set. You use a learning model with $1,000$ hypotheses and select the final hypothesis g based on the 400 training examples. We wish to estimate $E_{\text{out}}(g)$. We have access to two estimates: $E_{\text{in}}(g)$, the in-sample error on the 400 training examples; and, $E_{\text{test}}(g)$, the test error on the 200 test examples that were set aside.

(a) Using a 5% error tolerance ($\delta = 0.05$), which estimate has the higher 'error bar'?

(b) Is there any reason why you shouldn't reserve even more examples for testing?

Another aspect that distinguishes the test set from the training set is that the test set is not biased. Both sets are finite samples that are bound to have some variance due to sample size, but the test set doesn't have an optimistic or pessimistic bias in its estimate of E_{out}. The training set has an optimistic bias, since it was used to choose a hypothesis that looked good *on it*. The VC generalization bound implicitly takes that bias into consideration, and that's why it gives a huge error bar. The test set just has straight finite-sample variance, but no bias. When you report the value of E_{test} to your customer and they try your system on new data, they are as likely to be pleasantly surprised as unpleasantly surprised, though quite likely not to be surprised at all.

There is a price to be paid for having a test set. The test set does not affect the outcome of our learning process, which only uses the training set. The test set just tells us how well we did. Therefore, if we set aside some

of the data points provided by the customer as a test set, we end up using fewer examples for training. Since the training set is used to select one of the hypotheses in $\mathcal{H}$, training examples are essential to finding a good hypothesis. If we take a big chunk of the data for testing and end up with too few examples for training, we may not get a good hypothesis from the training part even if we can reliably evaluate it in the testing part. We may end up reporting to the customer, with high confidence mind you, that the g we are delivering is terrible ☺. There is thus a tradeoff to setting aside test examples. We will address that tradeoff in more detail and learn some clever tricks to get around it in Chapter 4.

In some of the learning literature, E_{test} is used as synonymous with E_{out}. When we report experimental results in this book, we will often treat E_{test} based on a large test set as if it was E_{out} because of the closeness of the two quantities.

2.2.4 Other Target Types

Although the VC analysis was based on binary target functions, it can be extended to real-valued functions, as well as to other types of functions. The proofs in those cases are quite technical, and they do not add to the insight that the VC analysis of binary functions provides. Therefore, we will introduce an alternative approach that covers real-valued functions and provides new insights into generalization. The approach is based on bias-variance analysis, and will be discussed in the next section.

In order to deal with real-valued functions, we need to adapt the definitions of E_{in} and E_{out} that have so far been based on binary functions. We defined E_{in} and E_{out} in terms of binary error; either $h(\mathbf{x}) = f(\mathbf{x})$ or else $h(\mathbf{x}) \neq f(\mathbf{x})$. If f and h are real-valued, a more appropriate error measure would gauge how far $f(\mathbf{x})$ and $h(\mathbf{x})$ are from each other, rather than just whether their values are exactly the same.

An error measure that is commonly used in this case is the squared error $\mathsf{e}(h(\mathbf{x}), f(\mathbf{x})) = (h(\mathbf{x}) - f(\mathbf{x}))^2$. We can define in-sample and out-of-sample versions of this error measure. The out-of-sample error is based on the expected value of the error measure over the entire input space $\mathcal{X}$,

$$E_{\text{out}}(h) = \mathbb{E}\left[(h(\mathbf{x}) - f(\mathbf{x}))^2\right],$$

while the in-sample error is based on averaging the error measure over the data set,

$$E_{\text{in}}(h) = \frac{1}{N}\sum_{n=1}^{N}(h(\mathbf{x}_n) - f(\mathbf{x}_n))^2.$$

These definitions make E_{in} a sample estimate of E_{out} just as it was in the case of binary functions. In fact, the error measure used for binary functions can also be expressed as a squared error.