Chapter 8

Generalization

This chapter discusses tools to analyze and understand the generalization of machine learning models, i.e, their performances on unseen test examples. Recall that for supervised learning problems, given a training dataset $\{(x^{(i)}, y^{(i)})\}_{i=1}^n$, we typically learn a model h_θ by minimizing a loss/cost function $J(\theta)$, which encourages h_θ to fit the data. E.g., when the loss function is the least square loss (aka mean squared error), we have $J(\theta) = \frac{1}{n}\sum_{i=1}^n (y^{(i)} - h_\theta(x^{(i)}))^2$. This loss function for training purposes is oftentimes referred to as the **training** loss/error/cost.

However, minimizing the training loss is **not** our ultimate goal—it is merely our approach towards the goal of learning a predictive model. The most important evaluation metric of a model is the loss on unseen test examples, which is oftentimes referred to as the test error. Formally, we sample a test example (x, y) from the so-called test distribution $\mathcal{D}$, and measure the model's error on it, by, e.g., the mean squared error, $(h_\theta(x) - y)^2$. The expected loss/error over the randomness of the test example is called the test loss/error,[1]

$$L(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[(y - h_\theta(x))^2] \tag{8.1}$$

Note that the measurement of the error involves computing the expectation, and in practice, it can be approximated by the average error on many sampled test examples, which are referred to as the test dataset. Note that the key difference here between training and test datasets is that the test examples

[1]In theoretical and statistical literature, we oftentimes call the uniform distribution over the training set $\{(x^{(i)}, y^{(i)})\}_{i=1}^n$, denoted by $\widehat{\mathcal{D}}$, an empirical distribution, and call $\mathcal{D}$ the population distribution. Partly because of this, the training loss is also referred to as the empirical loss/risk/error, and the test loss is also referred to as the population loss/risk/error.

are *unseen*, in the sense that the training procedure has not used the test examples. In classical statistical learning settings, the training examples are also drawn from the same distribution as the test distribution $\mathcal{D}$, but still the test examples are unseen by the learning procedure whereas the training examples are seen.[2]

Because of this key difference between training and test datasets, even if they are both drawn from the same distribution $\mathcal{D}$, the test error is not necessarily always close to the training error.[3] As a result, successfully minimizing the training error may not always lead to a small test error. We typically say the model **overfits** the data if the model predicts accurately on the training dataset but doesn't generalize well to other test examples, that is, if the training error is small but the test error is large. We say the model **underfits** the data if the training error is relatively large[4] (and in this case, typically the test error is also relatively large.)

This chapter studies how the test error is influenced by the learning procedure, especially the choice of model parameterizations. We will decompose the test error into "bias" and "variance" terms and study how each of them is affected by the choice of model parameterizations and their tradeoffs. Using the bias-variance tradeoff, we will discuss when overfitting and underfitting will occur and be avoided. We will also discuss the double descent phenomenon in Section 8.2 and some classical theoretical results in Section 8.3.

[2]These days, researchers have increasingly been more interested in the setting with "domain shift", that is, the training distribution and test distribution are different.

[3]the difference between test error and training error is often referred to as the generalization gap. The term *generalization error* in some literature means the test error, and in some other literature means the generalization gap.

[4]e.g., larger than the intrinsic noise level of the data in regression problems.

8.1 Bias-variance tradeoff

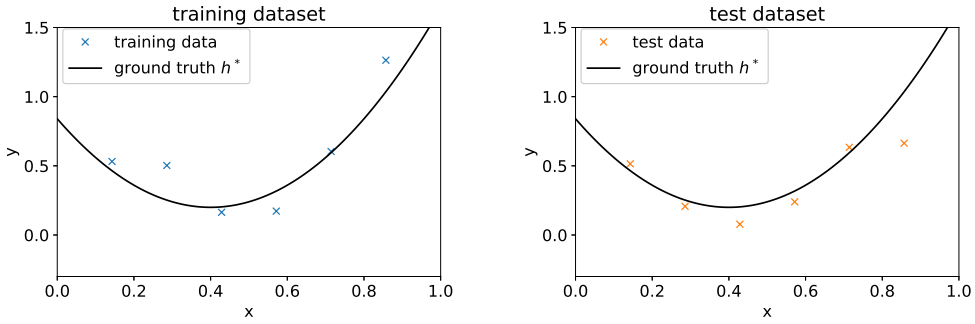


Figure 8.1: A running example of training and test dataset for this section.

As an illustrating example, we consider the following training dataset and test dataset, which are also shown in Figure 8.1. The training inputs $x^{(i)}$'s are randomly chosen and the outputs $y^{(i)}$ are generated by $y^{(i)} = h^\star(x^{(i)}) + \xi^{(i)}$ where the function $h^\star(\cdot)$ is a quadratic function and is shown in Figure 8.1 as the solid line, and $\xi^{(i)}$ is the a observation noise assumed to be generated from $\sim N(0, \sigma^2)$. A test example (x, y) also has the same input-output relationship $y = h^\star(x) + \xi$ where $\xi \sim N(0, \sigma^2)$. It's impossible to predict the noise ξ, and therefore essentially our goal is to recover the function $h^\star(\cdot)$.

We will consider the test error of learning various types of models. When talking about linear regression, we discussed the problem of whether to fit a "simple" model such as the linear "$y = \theta_0 + \theta_1 x$," or a more "complex" model such as the polynomial "$y = \theta_0 + \theta_1 x + \cdots \theta_5 x^5$."

We start with fitting a linear model, as shown in Figure 8.2. The best fitted linear model cannot predict y from x accurately even on the training dataset, let alone on the test dataset. This is because the true relationship between y and x is not linear—any linear model is far away from the true function $h^\star(\cdot)$. As a result, the training error is large and this is a typical situation of *underfitting*.

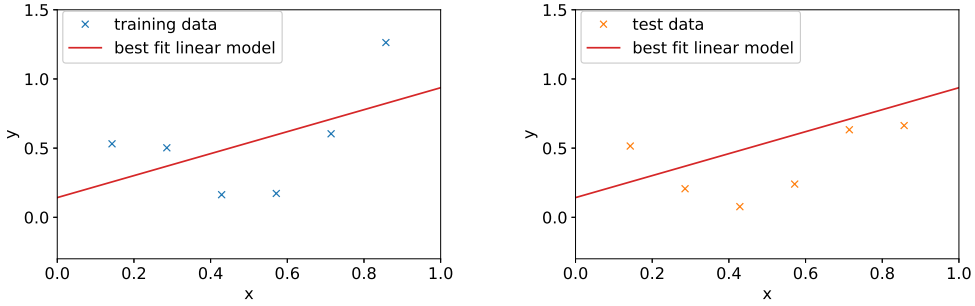


Figure 8.2: The best fit linear model has large training and test errors.

The issue cannot be mitigated with more training examples—even with a very large amount of, or even infinite training examples, the best fitted linear model is still inaccurate and fails to capture the structure of the data (Figure 8.3). Even if the noise is not present in the training data, the issue still occurs (Figure 8.4). Therefore, the fundamental bottleneck here is the linear model family's inability to capture the structure in the data—linear models cannot represent the true quadratic function $h^\star$—, but not the lack of the data. Informally, we define the **bias** of a model to be the test error even if we were to fit it to a very (say, infinitely) large training dataset. Thus, in this case, the linear model suffers from large bias, and underfits (i.e., fails to capture structure exhibited by) the data.

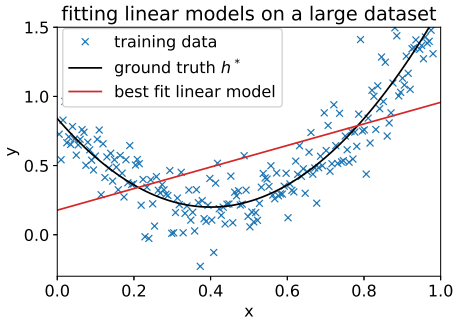


Figure 8.3: The best fit linear model on a much larger dataset still has a large training error.

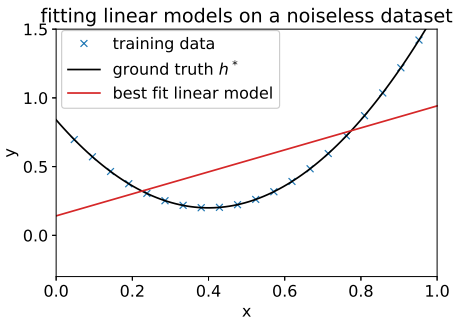


Figure 8.4: The best fit linear model on a noiseless dataset also has a large training/test error.

Next, we fit a 5th-degree polynomial to the data. Figure 8.5 shows that it fails to learn a good model either. However, the failure pattern is different from the linear model case. Specifically, even though the learnt 5th-degree

polynomial did a very good job predicting $y^{(i)}$'s from $x^{(i)}$'s for training examples, it does not work well on test examples (Figure 8.5). In other words, the model learnt from the training set does not *generalize* well to other test examples—the test error is high. Contrary to the behavior of linear models, the bias of the 5-th degree polynomials is small—if we were to fit a 5-th degree polynomial to an extremely large dataset, the resulting model would be close to a quadratic function and be accurate (Figure 8.6). This is because the family of 5-th degree polynomials contains all the quadratic functions (setting $\theta_5 = \theta_4 = \theta_3 = 0$ results in a quadratic function), and, therefore, 5-th degree polynomials are in principle capable of capturing the structure of the data.

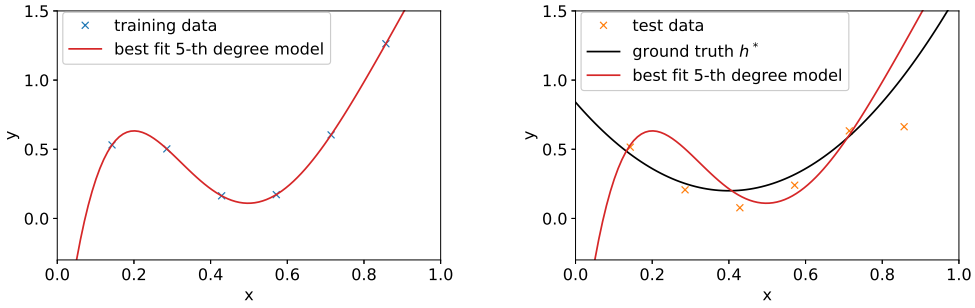


Figure 8.5: Best fit 5-th degree polynomial has zero training error, but still has a large test error and does not recover the the ground truth. This is a classic situation of overfitting.

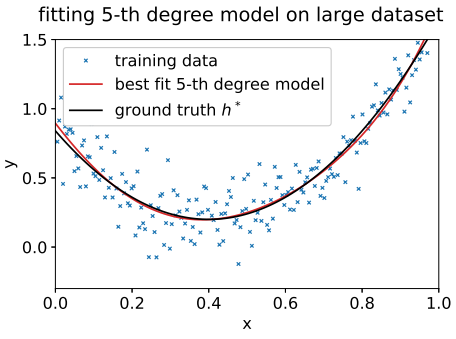


Figure 8.6: The best fit 5-th degree polynomial on a huge dataset nearly recovers the ground-truth—suggesting that the culprit in Figure 8.5 is the variance (or lack of data) but not bias.

The failure of fitting 5-th degree polynomials can be captured by another

component of the test error, called **variance** of a model fitting procedure. Specifically, when fitting a 5-th degree polynomial as in Figure 8.7, there is a large risk that we're fitting patterns in the data that happened to be present in our *small, finite* training set, but that do not reflect the wider pattern of the relationship between x and y. These "spurious" patterns in the training set are (mostly) due to the observation noise $\xi^{(i)}$, and fitting these spurious patters results in a model with large test error. In this case, we say the model has a large variance.

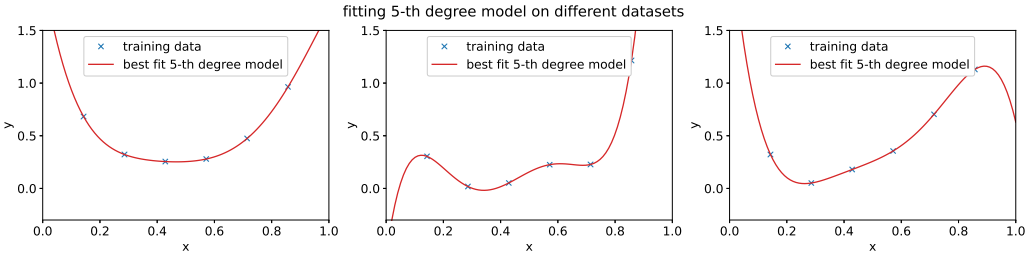


Figure 8.7: The best fit 5-th degree models on three different datasets generated from the same distribution behave quite differently, suggesting the existence of a large variance.

The variance can be intuitively (and mathematically, as shown in Section 8.1.1) characterized by the amount of variations across models learnt on multiple different training datasets (drawn from the same underlying distribution). The "spurious patterns" are specific to the randomness of the noise (and inputs) in a particular dataset, and thus are different across multiple training datasets. Therefore, overfitting to the "spurious patterns" of multiple datasets should result in very different models. Indeed, as shown in Figure 8.7, the models learned on the three different training datasets are quite different, overfitting to the "spurious patterns" of each datasets.

Often, there is a tradeoff between bias and variance. If our model is too "simple" and has very few parameters, then it may have large bias (but small variance), and it typically may suffer from underfittng. If it is too "complex" and has very many parameters, then it may suffer from large variance (but have smaller bias), and thus overfitting. See Figure 8.8 for a typical tradeoff between bias and variance.

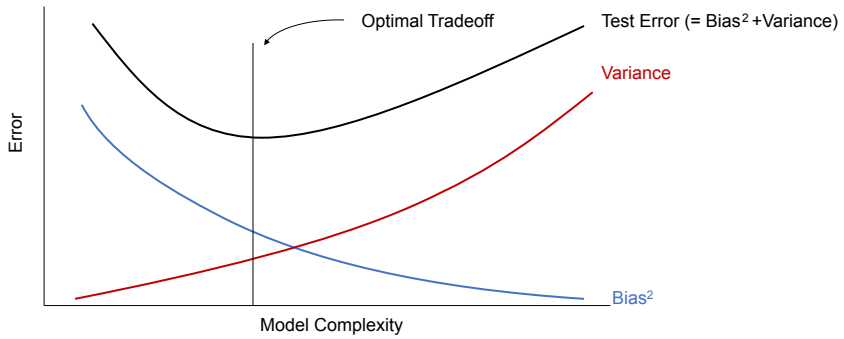


Figure 8.8: An illustration of the typical bias-variance tradeoff.

As we will see formally in Section 8.1.1, the test error can be decomposed as a summation of bias and variance. This means that the test error will have a convex curve as the model complexity increases, and in practice we should tune the model complexity to achieve the best tradeoff. For instance, in the example above, fitting a quadratic function does better than either of the extremes of a first or a 5-th degree polynomial, as shown in Figure 8.9.

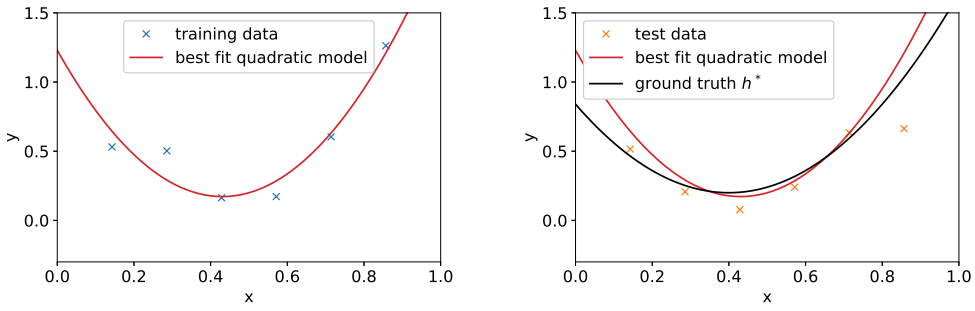


Figure 8.9: Best fit quadratic model has small training and test error because quadratic model achieves a better tradeoff.

Interestingly, the bias-variance tradeoff curves or the test error curves do not universally follow the shape in Figure 8.8, at least not universally when the model complexity is simply measured by the number of parameters. (We will discuss the so-called double descent phenomenon in Section 8.2.) Nevertheless, the principle of bias-variance tradeoff is perhaps still the first resort when analyzing and predicting the behavior of test errors.

8.1.1 A mathematical decomposition (for regression)

To formally state the bias-variance tradeoff for regression problems, we consider the following setup (which is an extension of the beginning paragraph of Section 8.1).

- Draw a training dataset $S = \{x^{(i)}, y^{(i)}\}_{i=1}^{n}$ such that $y^{(i)} = h^\star(x^{(i)}) + \xi^{(i)}$ where $\xi^{(i)} \in N(0, \sigma^2)$.
- Train a model on the dataset S, denoted by $\hat{h}_S$.
- Take a test example (x, y) such that $y = h^\star(x) + \xi$ where $\xi \sim N(0, \sigma^2)$, and measure the expected test error (averaged over the random draw of the training set S and the randomness of ξ)[5][6]

$$\mathrm{MSE}(x) = \mathbb{E}_{S,\xi}[(y - h_S(x))^2] \tag{8.2}$$

We will decompose the MSE into a bias and variance term. We start by stating a following simple mathematical tool that will be used twice below.

Claim 8.1.1: Suppose A and B are two independent real random variables and $\mathbb{E}[A] = 0$. Then, $\mathbb{E}[(A + B)^2] = \mathbb{E}[A^2] + \mathbb{E}[B^2]$.

As a corollary, because a random variable A is independent with a constant c, when $\mathbb{E}[A] = 0$, we have $\mathbb{E}[(A + c)^2] = \mathbb{E}[A^2] + c^2$.

The proof of the claim follows from expanding the square: $\mathbb{E}[(A+B)^2] = \mathbb{E}[A^2] + \mathbb{E}[B^2] + 2\mathbb{E}[AB] = \mathbb{E}[A^2] + \mathbb{E}[B^2]$. Here we used the independence to show that $\mathbb{E}[AB] = \mathbb{E}[A]\mathbb{E}[B] = 0$.

Using Claim 8.1.1 with $A = \xi$ and $B = h^\star(x) - \hat{h}_S(x)$, we have

$$\begin{aligned}
\mathrm{MSE}(x) = \mathbb{E}[(y - h_S(x))^2] &= \mathbb{E}[(\xi + (h^\star(x) - h_S(x)))^2] && (8.3)\\
&= \mathbb{E}[\xi^2] + \mathbb{E}[(h^\star(x) - h_S(x))^2] \quad \text{(by Claim 8.1.1)}\\
&= \sigma^2 + \mathbb{E}[(h^\star(x) - h_S(x))^2] && (8.4)
\end{aligned}$$

Then, let's define $h_{\mathrm{avg}}(x) = \mathbb{E}_S[h_S(x)]$ as the "average model"—the model obtained by drawing an infinite number of datasets, training on them, and averaging their predictions on x. Note that h_{avg} is a hypothetical model for analytical purposes that can not be obtained in reality (because we don't

[5] For simplicity, the test input x is considered to be fixed here, but the same conceptual message holds when we average over the choice of x's.

[6] The subscript under the expectation symbol is to emphasize the variables that are considered as random by the expectation operation.

have infinite number of datasets). It turns out that for many cases, h_{avg} is (approximately) equal to the the model obtained by training on a *single* dataset with infinite samples. Thus, we can also intuitively interpret h_{avg} this way, which is consistent with our intuitive definition of bias in the previous subsection.

We can further decompose $\text{MSE}(x)$ by letting $c = h^\star(x) - h_{\text{avg}}(x)$ (which is a constant that does not depend on the choice of S!) and $A = h_{\text{avg}}(x) - h_S(x)$ in the corollary part of Claim 8.1.1:

$$\begin{aligned}
\text{MSE}(x) &= \sigma^2 + \mathbb{E}[(h^\star(x) - h_S(x))^2] && (8.5)\\
&= \sigma^2 + (h^\star(x) - h_{\text{avg}}(x))^2 + \mathbb{E}[(h_{\text{avg}} - h_S(x))^2] && (8.6)\\
&= \underbrace{\sigma^2}_{\text{unavoidable}} + \underbrace{(h^\star(x) - h_{\text{avg}}(x))^2}_{\triangleq \text{ bias}^2} + \underbrace{\text{var}(h_S(x))}_{\triangleq \text{ variance}} && (8.7)
\end{aligned}$$

We call the second term the bias (square) and the third term the variance. As discussed before, the bias captures the part of the error that are introduced due to the lack of expressivity of the model. Recall that h_{avg} can be thought of as the best possible model learned even with infinite data. Thus, the bias is not due to the lack of data, but is rather caused by that the family of models fundamentally cannot approximate the $h^\star$. For example, in the illustrating example in Figure 8.2, because any linear model cannot approximate the true quadratic function $h^\star$, neither can h_{avg}, and thus the bias term has to be large.

The variance term captures how the random nature of the finite dataset introduces errors in the learned model. It measures the sensitivity of the learned model to the randomness in the dataset. It often decreases as the size of the dataset increases.

There is nothing we can do about the first term σ^2 as we can not predict the noise ξ by definition.

Finally, we note that the bias-variance decomposition for classification is much less clear than for regression problems. There have been several proposals, but there is as yet no agreement on what is the "right" and/or the most useful formalism.

8.2 The double descent phenomenon

Model-wise double descent. Recent works have demonstrated that the test error can present a "double descent" phenomenon in a range of machine

learning models including linear models and deep neural networks.[7] The conventional wisdom, as discussed in Section 8.1, is that as we increase the model complexity, the test error first decreases and then increases, as illustrated in Figure 8.8. However, in many cases, we empirically observe that the test error can have a second descent—it first decreases, then increases to a peak around when the model size is large enough to fit all the training data very well, and then decreases again in the so-called overparameterized regime, where the number of parameters is larger than the number of data points. See Figure 8.10 for an illustration of the typical curves of test errors against model complexity (measured by the number of parameters). To some extent, the overparameterized regime with the second descent is considered as new to the machine learning community—partly because lightly-regularized, overparameterized models are only extensively used in the deep learning era. A practical implication of the phenomenon is that one should not hold back from scaling into and experimenting with over-parametrized models because the test error may well decrease again to a level even smaller than the previous lowest point. Actually, in many cases, larger overparameterized models always lead to a better test performance (meaning there won't be a second ascent after the second descent).

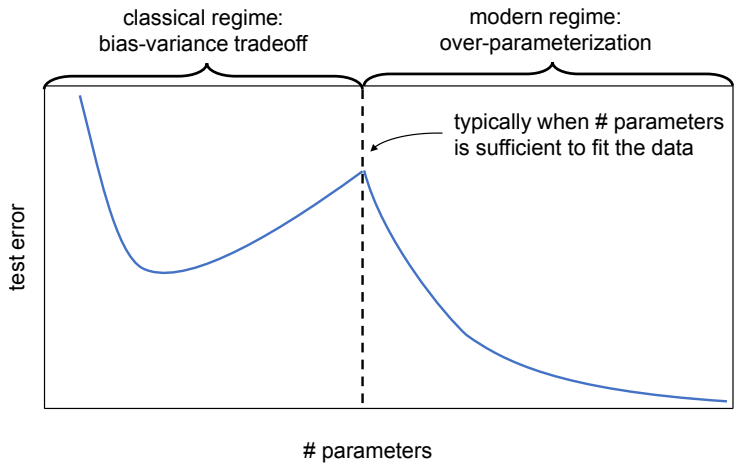


Figure 8.10: A typical model-wise double descent phenomenon. As the number of parameters increases, the test error first decreases when the number of parameters is smaller than the training data. Then in the overparameterized regime, the test error decreases again.

[7]The discovery of the phenomenon perhaps dates back to Opper [1995, 2001], and has been recently popularized by Belkin et al. [2020], Hastie et al. [2019], etc.

Sample-wise double descent. A priori, we would expect that more training examples always lead to smaller test errors—more samples give strictly more information for the algorithm to learn from. However, recent work [Nakkiran, 2019] observes that the test error is not monotonically decreasing as we increase the sample size. Instead, as shown in Figure 8.11, the test error decreases, and then increases and peaks around when the number of examples (denoted by n) is similar to the number of parameters (denoted by d), and then decreases again. We refer to this as the sample-wise double descent phenomenon. To some extent, sample-wise double descent and model-wise double descent are essentially describing similar phenomena—the test error is peaked when $n \approx d$.

Explanation and mitigation strategy. The sample-wise double descent, or, in particular, the peak of test error at $n \approx d$, suggests that the existing training algorithms evaluated in these experiments are far from optimal when $n \approx d$. We will be better off by tossing away some examples and run the algorithms with a smaller sample size to steer clear of the peak. In other words, in principle, there are other algorithms that can achieve smaller test error when $n \approx d$, but the algorithms evaluated in these experiments fail to do so. The sub-optimality of the learning procedure appears to be the culprit of the peak in both sample-wise and model-wise double descent.

Indeed, with an optimally-tuned regularization (which will be discussed more in Section 9), the test error in the $n \approx d$ regime can be dramatically improved, and the model-wise and sample-wise double descent are both mitigated. See Figure 8.11.

The intuition above only explains the peak in the model-wise and sample-wise double descent, but does not explain the second descent in the model-wise double descent—why overparameterized models are able to generalize so well. The theoretical understanding of overparameterized models is an active research area with many recent advances. A typical explanation is that the commonly-used optimizers such as gradient descent provide an implicit regularization effect (which will be discussed in more detail in Section 9.2). In other words, even in the overparameterized regime and with an unregularized loss function, the model is still implicitly regularized, and thus exhibits a better test performance than an arbitrary solution that fits the data. For example, for linear models, when $n \ll d$, the gradient descent optimizer with zero initialization finds the *minimum norm* solution that fits the data (instead of an arbitrary solution that fits the data), and the minimum norm regularizer turns out to be a sufficiently good for the overparameterized regime (but it's not a good regularizer when $n \approx d$, resulting in the peak of test

error).

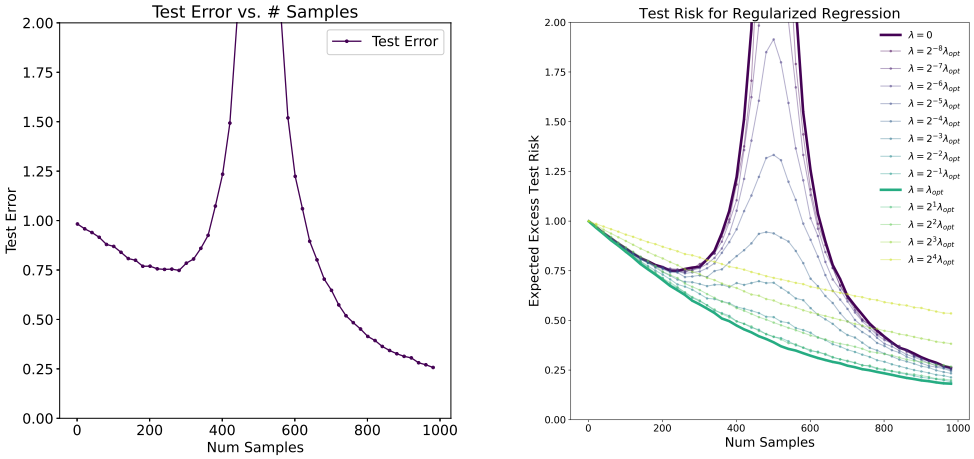


Figure 8.11: **Left:** The sample-wise double descent phenomenon for linear models. **Right:** The sample-wise double descent with different regularization strength for linear models. Using the optimal regularization parameter λ (optimally tuned for each n, shown in green solid curve) mitigates double descent. **Setup:** The data distribution of (x, y) is $x \sim \mathcal{N}(0, I_d)$ and $y \sim x^\top \beta + \mathcal{N}(0, \sigma^2)$ where $d = 500, \sigma = 0.5$ and $\|\beta\|_2 = 1$.[8]

Finally, we also remark that the double descent phenomenon has been mostly observed when the model complexity is measured by the number of parameters. It is unclear if and when the number of parameters is the best complexity measure of a model. For example, in many situations, the norm of the models is used as a complexity measure. As shown in Figure 8.12 right, for a particular linear case, if we plot the test error against the norm of the learnt model, the double descent phenomenon no longer occurs. This is partly because the norm of the learned model is also peaked around $n \approx d$ (See Figure 8.12 (middle) or Belkin et al. [2019], Mei and Montanari [2022], and discussions in Section 10.8 of James et al. [2021]). For deep neural networks, the correct complexity measure is even more elusive. The study of double descent phenomenon is an active research topic.

[8]The figure is reproduced from Figure 1 of Nakkiran et al. [2020]. Similar phenomenon are also observed in Hastie et al. [2022], Mei and Montanari [2022]

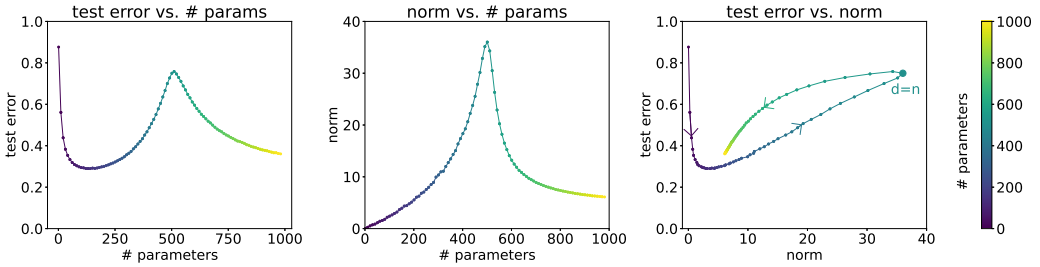


Figure 8.12: **Left:** The double descent phenomenon, where the number of parameters is used as the model complexity. **Middle:** The norm of the learned model is peaked around $n \approx d$. **Right:** The test error against the norm of the learnt model. The color bar indicate the number of parameters and the arrows indicates the direction of increasing model size. Their relationship are closer to the convention wisdom than to a double descent. **Setup:** We consider a linear regression with a fixed dataset of size $n = 500$. The input x is a random ReLU feature on Fashion-MNIST, and output $y \in \mathbb{R}^{10}$ is the one-hot label. This is the same setting as in Section 5.2 of Nakkiran et al. [2020].

8.3 Sample complexity bounds (optional readings)

8.3.1 Preliminaries

In this set of notes, we begin our foray into learning theory. Apart from being interesting and enlightening in its own right, this discussion will also help us hone our intuitions and derive rules of thumb about how to best apply learning algorithms in different settings. We will also seek to answer a few questions: First, can we make formal the bias/variance tradeoff that was just discussed? This will also eventually lead us to talk about model selection methods, which can, for instance, automatically decide what order polynomial to fit to a training set. Second, in machine learning it's really generalization error that we care about, but most learning algorithms fit their models to the training set. Why should doing well on the training set tell us anything about generalization error? Specifically, can we relate error on the training set to generalization error? Third and finally, are there conditions under which we can actually prove that learning algorithms will work well?

We start with two simple but very useful lemmas.

Lemma. (The union bound). Let $A_1, A_2, \ldots, A_k$ be k different events (that may not be independent). Then

$$P(A_1 \cup \cdots \cup A_k) \leq P(A_1) + \ldots + P(A_k).$$

In probability theory, the union bound is usually stated as an axiom (and thus we won't try to prove it), but it also makes intuitive sense: The probability of any one of k events happening is at most the sum of the probabilities of the k different events.

Lemma. (Hoeffding inequality) Let $Z_1, \ldots, Z_n$ be n independent and identically distributed (iid) random variables drawn from a Bernoulli(ϕ) distribution. I.e., $P(Z_i = 1) = \phi$, and $P(Z_i = 0) = 1 - \phi$. Let $\hat{\phi} = (1/n)\sum_{i=1}^{n} Z_i$ be the mean of these random variables, and let any $\gamma > 0$ be fixed. Then

$$P(|\phi - \hat{\phi}| > \gamma) \leq 2\exp(-2\gamma^2 n)$$

This lemma (which in learning theory is also called the **Chernoff bound**) says that if we take $\hat{\phi}$—the average of n Bernoulli(ϕ) random variables—to be our estimate of ϕ, then the probability of our being far from the true value is small, so long as n is large. Another way of saying this is that if you have a biased coin whose chance of landing on heads is ϕ, then if you toss it n

times and calculate the fraction of times that it came up heads, that will be a good estimate of ϕ with high probability (if n is large).

Using just these two lemmas, we will be able to prove some of the deepest and most important results in learning theory.

To simplify our exposition, let's restrict our attention to binary classification in which the labels are $y \in \{0, 1\}$. Everything we'll say here generalizes to other problems, including regression and multi-class classification.

We assume we are given a training set $S = \{(x^{(i)}, y^{(i)}); i = 1, \ldots, n\}$ of size n, where the training examples $(x^{(i)}, y^{(i)})$ are drawn iid from some probability distribution $\mathcal{D}$. For a hypothesis h, we define the **training error** (also called the **empirical risk** or **empirical error** in learning theory) to be

$$\hat{\varepsilon}(h) = \frac{1}{n} \sum_{i=1}^{n} 1\{h(x^{(i)}) \neq y^{(i)}\}.$$

This is just the fraction of training examples that h misclassifies. When we want to make explicit the dependence of $\hat{\varepsilon}(h)$ on the training set S, we may also write this a $\hat{\varepsilon}_S(h)$. We also define the generalization error to be

$$\varepsilon(h) = P_{(x,y)\sim\mathcal{D}}(h(x) \neq y).$$

I.e. this is the probability that, if we now draw a new example (x, y) from the distribution $\mathcal{D}$, h will misclassify it.

Note that we have assumed that the training data was drawn from the *same* distribution $\mathcal{D}$ with which we're going to evaluate our hypotheses (in the definition of generalization error). This is sometimes also referred to as one of the **PAC** assumptions.[9]

Consider the setting of linear classification, and let $h_\theta(x) = 1\{\theta^T x \geq 0\}$. What's a reasonable way of fitting the parameters θ? One approach is to try to minimize the training error, and pick

$$\hat{\theta} = \arg\min_{\theta} \hat{\varepsilon}(h_\theta).$$

We call this process **empirical risk minimization** (ERM), and the resulting hypothesis output by the learning algorithm is $\hat{h} = h_{\hat{\theta}}$. We think of ERM as the most "basic" learning algorithm, and it will be this algorithm that we

[9]PAC stands for "probably approximately correct," which is a framework and set of assumptions under which numerous results on learning theory were proved. Of these, the assumption of training and testing on the same distribution, and the assumption of the independently drawn training examples, were the most important.

focus on in these notes. (Algorithms such as logistic regression can also be viewed as approximations to empirical risk minimization.)

In our study of learning theory, it will be useful to abstract away from the specific parameterization of hypotheses and from issues such as whether we're using a linear classifier. We define the **hypothesis class** $\mathcal{H}$ used by a learning algorithm to be the set of all classifiers considered by it. For linear classification, $\mathcal{H} = \{h_\theta : h_\theta(x) = 1\{\theta^T x \geq 0\}, \theta \in \mathbb{R}^{d+1}\}$ is thus the set of all classifiers over $\mathcal{X}$ (the domain of the inputs) where the decision boundary is linear. More broadly, if we were studying, say, neural networks, then we could let $\mathcal{H}$ be the set of all classifiers representable by some neural network architecture.

Empirical risk minimization can now be thought of as a minimization over the class of functions $\mathcal{H}$, in which the learning algorithm picks the hypothesis:

$$\hat{h} = \arg\min_{h \in \mathcal{H}} \hat{\varepsilon}(h)$$

8.3.2 The case of finite $\mathcal{H}$

Let's start by considering a learning problem in which we have a finite hypothesis class $\mathcal{H} = \{h_1, \ldots, h_k\}$ consisting of k hypotheses. Thus, $\mathcal{H}$ is just a set of k functions mapping from $\mathcal{X}$ to $\{0, 1\}$, and empirical risk minimization selects $\hat{h}$ to be whichever of these k functions has the smallest training error.

We would like to give guarantees on the generalization error of $\hat{h}$. Our strategy for doing so will be in two parts: First, we will show that $\hat{\varepsilon}(h)$ is a reliable estimate of $\varepsilon(h)$ for all h. Second, we will show that this implies an upper-bound on the generalization error of $\hat{h}$.

Take any one, fixed, $h_i \in \mathcal{H}$. Consider a Bernoulli random variable Z whose distribution is defined as follows. We're going to sample $(x, y) \sim \mathcal{D}$. Then, we set $Z = 1\{h_i(x) \neq y\}$. I.e., we're going to draw one example, and let Z indicate whether h_i misclassifies it. Similarly, we also define $Z_j = 1\{h_i(x^{(j)}) \neq y^{(j)}\}$. Since our training set was drawn iid from $\mathcal{D}$, Z and the Z_j's have the same distribution.

We see that the misclassification probability on a randomly drawn example—that is, $\varepsilon(h)$—is exactly the expected value of Z (and Z_j). Moreover, the training error can be written

$$\hat{\varepsilon}(h_i) = \frac{1}{n} \sum_{j=1}^{n} Z_j.$$

Thus, $\hat{\varepsilon}(h_i)$ is exactly the mean of the n random variables Z_j that are drawn iid from a Bernoulli distribution with mean $\varepsilon(h_i)$. Hence, we can apply the

Hoeffding inequality, and obtain

$$P(|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) \leq 2\exp(-2\gamma^2 n).$$

This shows that, for our particular h_i, training error will be close to generalization error with high probability, assuming n is large. But we don't just want to guarantee that $\varepsilon(h_i)$ will be close to $\hat{\varepsilon}(h_i)$ (with high probability) for just only one particular h_i. We want to prove that this will be true simultaneously for *all* $h \in \mathcal{H}$. To do so, let A_i denote the event that $|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma$. We've already shown that, for any particular A_i, it holds true that $P(A_i) \leq 2\exp(-2\gamma^2 n)$. Thus, using the union bound, we have that

$$\begin{aligned} P(\exists\, h \in \mathcal{H}.|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) &= P(A_1 \cup \cdots \cup A_k) \\ &\leq \sum_{i=1}^{k} P(A_i) \\ &\leq \sum_{i=1}^{k} 2\exp(-2\gamma^2 n) \\ &= 2k\exp(-2\gamma^2 n) \end{aligned}$$

If we subtract both sides from 1, we find that

$$\begin{aligned} P(\neg\exists\, h \in \mathcal{H}.|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) &= P(\forall h \in \mathcal{H}.|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| \leq \gamma) \\ &\geq 1 - 2k\exp(-2\gamma^2 n) \end{aligned}$$

(The "$\neg$" symbol means "not.") So, with probability at least $1 - 2k\exp(-2\gamma^2 n)$, we have that $\varepsilon(h)$ will be within γ of $\hat{\varepsilon}(h)$ for all $h \in \mathcal{H}$. This is called a *uniform convergence* result, because this is a bound that holds simultaneously for all (as opposed to just one) $h \in \mathcal{H}$.

In the discussion above, what we did was, for particular values of n and γ, give a bound on the probability that for some $h \in \mathcal{H}$, $|\varepsilon(h) - \hat{\varepsilon}(h)| > \gamma$. There are three quantities of interest here: n, γ, and the probability of error; we can bound either one in terms of the other two.

For instance, we can ask the following question: Given γ and some $\delta > 0$, how large must n be before we can guarantee that with probability at least $1 - \delta$, training error will be within γ of generalization error? By setting $\delta = 2k\exp(-2\gamma^2 n)$ and solving for n, [you should convince yourself this is the right thing to do!], we find that if

$$n \geq \frac{1}{2\gamma^2}\log\frac{2k}{\delta},$$

then with probability at least $1-\delta$, we have that $|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$ for all $h \in \mathcal{H}$. (Equivalently, this shows that the probability that $|\varepsilon(h) - \hat{\varepsilon}(h)| > \gamma$ for some $h \in \mathcal{H}$ is at most δ.) This bound tells us how many training examples we need in order make a guarantee. The training set size n that a certain method or algorithm requires in order to achieve a certain level of performance is also called the algorithm's **sample complexity**.

The key property of the bound above is that the number of training examples needed to make this guarantee is only *logarithmic* in k, the number of hypotheses in $\mathcal{H}$. This will be important later.

Similarly, we can also hold n and δ fixed and solve for γ in the previous equation, and show [again, convince yourself that this is right!] that with probability $1-\delta$, we have that for all $h \in \mathcal{H}$,

$$|\hat{\varepsilon}(h) - \varepsilon(h)| \leq \sqrt{\frac{1}{2n} \log \frac{2k}{\delta}}.$$

Now, let's assume that uniform convergence holds, i.e., that $|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$ for all $h \in \mathcal{H}$. What can we prove about the generalization of our learning algorithm that picked $\hat{h} = \arg\min_{h \in \mathcal{H}} \hat{\varepsilon}(h)$?

Define $h^* = \arg\min_{h \in \mathcal{H}} \varepsilon(h)$ to be the best possible hypothesis in $\mathcal{H}$. Note that h^* is the best that we could possibly do given that we are using $\mathcal{H}$, so it makes sense to compare our performance to that of h^*. We have:

$$\begin{aligned} \varepsilon(\hat{h}) &\leq \hat{\varepsilon}(\hat{h}) + \gamma \\ &\leq \hat{\varepsilon}(h^*) + \gamma \\ &\leq \varepsilon(h^*) + 2\gamma \end{aligned}$$

The first line used the fact that $|\varepsilon(\hat{h}) - \hat{\varepsilon}(\hat{h})| \leq \gamma$ (by our uniform convergence assumption). The second used the fact that $\hat{h}$ was chosen to minimize $\hat{\varepsilon}(h)$, and hence $\hat{\varepsilon}(\hat{h}) \leq \hat{\varepsilon}(h)$ for all h, and in particular $\hat{\varepsilon}(\hat{h}) \leq \hat{\varepsilon}(h^*)$. The third line used the uniform convergence assumption again, to show that $\hat{\varepsilon}(h^*) \leq \varepsilon(h^*) + \gamma$. So, what we've shown is the following: If uniform convergence occurs, then the generalization error of $\hat{h}$ is at most 2γ worse than the best possible hypothesis in $\mathcal{H}$!

Let's put all this together into a theorem.

Theorem. Let $|\mathcal{H}| = k$, and let any n, δ be fixed. Then with probability at least $1-\delta$, we have that

$$\varepsilon(\hat{h}) \leq \left(\min_{h \in \mathcal{H}} \varepsilon(h) \right) + 2\sqrt{\frac{1}{2n} \log \frac{2k}{\delta}}.$$

This is proved by letting γ equal the $\sqrt{\cdot}$ term, using our previous argument that uniform convergence occurs with probability at least $1-\delta$, and then noting that uniform convergence implies $\varepsilon(h)$ is at most 2γ higher than $\varepsilon(h^*) = \min_{h\in\mathcal{H}} \varepsilon(h)$ (as we showed previously).

This also quantifies what we were saying previously saying about the bias/variance tradeoff in model selection. Specifically, suppose we have some hypothesis class $\mathcal{H}$, and are considering switching to some much larger hypothesis class $\mathcal{H}' \supseteq \mathcal{H}$. If we switch to $\mathcal{H}'$, then the first term $\min_h \varepsilon(h)$ can only decrease (since we'd then be taking a min over a larger set of functions). Hence, by learning using a larger hypothesis class, our "bias" can only decrease. However, if k increases, then the second $2\sqrt{\cdot}$ term would also increase. This increase corresponds to our "variance" increasing when we use a larger hypothesis class.

By holding γ and δ fixed and solving for n like we did before, we can also obtain the following sample complexity bound:

Corollary. Let $|\mathcal{H}| = k$, and let any δ, γ be fixed. Then for $\varepsilon(\hat{h}) \leq \min_{h\in\mathcal{H}} \varepsilon(h) + 2\gamma$ to hold with probability at least $1-\delta$, it suffices that

$$\begin{aligned} n &\geq \frac{1}{2\gamma^2} \log \frac{2k}{\delta} \\ &= O\left(\frac{1}{\gamma^2} \log \frac{k}{\delta}\right), \end{aligned}$$

8.3.3 The case of infinite $\mathcal{H}$

We have proved some useful theorems for the case of finite hypothesis classes. But many hypothesis classes, including any parameterized by real numbers (as in linear classification) actually contain an infinite number of functions. Can we prove similar results for this setting?

Let's start by going through something that is *not* the "right" argument. *Better and more general arguments exist*, but this will be useful for honing our intuitions about the domain.

Suppose we have an $\mathcal{H}$ that is parameterized by d real numbers. Since we are using a computer to represent real numbers, and IEEE double-precision floating point (`double`'s in C) uses 64 bits to represent a floating point number, this means that our learning algorithm, assuming we're using double-precision floating point, is parameterized by $64d$ bits. Thus, our hypothesis class really consists of at most $k = 2^{64d}$ different hypotheses. From the Corollary at the end of the previous section, we therefore find that, to guarantee

$\varepsilon(\hat{h}) \leq \varepsilon(h^*) + 2\gamma$, with to hold with probability at least $1-\delta$, it suffices that $n \geq O\left(\frac{1}{\gamma^2} \log \frac{2^{64d}}{\delta}\right) = O\left(\frac{d}{\gamma^2} \log \frac{1}{\delta}\right) = O_{\gamma,\delta}(d)$. (The γ, δ subscripts indicate that the last big-O is hiding constants that may depend on γ and δ.) Thus, the number of training examples needed is at most *linear* in the parameters of the model.

The fact that we relied on 64-bit floating point makes this argument not entirely satisfying, but the conclusion is nonetheless roughly correct: If what we try to do is minimize training error, then in order to learn "well" using a hypothesis class that has d parameters, generally we're going to need on the order of a linear number of training examples in d.

(At this point, it's worth noting that these results were proved for an algorithm that uses empirical risk minimization. Thus, while the linear dependence of sample complexity on d does generally hold for most discriminative learning algorithms that try to minimize training error or some approximation to training error, these conclusions do not always apply as readily to discriminative learning algorithms. Giving good theoretical guarantees on many non-ERM learning algorithms is still an area of active research.)

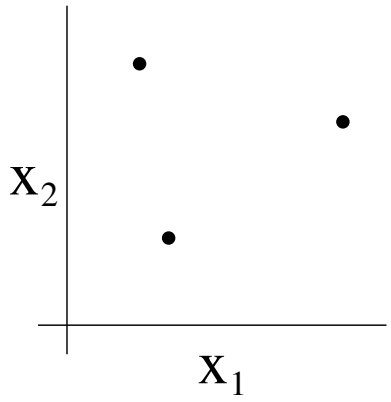
The other part of our previous argument that's slightly unsatisfying is that it relies on the parameterization of $\mathcal{H}$. Intuitively, this doesn't seem like it should matter: We had written the class of linear classifiers as $h_\theta(x) = 1\{\theta_0 + \theta_1 x_1 + \cdots \theta_d x_d \geq 0\}$, with $n+1$ parameters $\theta_0, \ldots, \theta_d$. But it could also be written $h_{u,v}(x) = 1\{(u_0^2 - v_0^2) + (u_1^2 - v_1^2)x_1 + \cdots (u_d^2 - v_d^2)x_d \geq 0\}$ with $2d+2$ parameters u_i, v_i. Yet, both of these are just defining the same $\mathcal{H}$: The set of linear classifiers in d dimensions.

To derive a more satisfying argument, let's define a few more things.

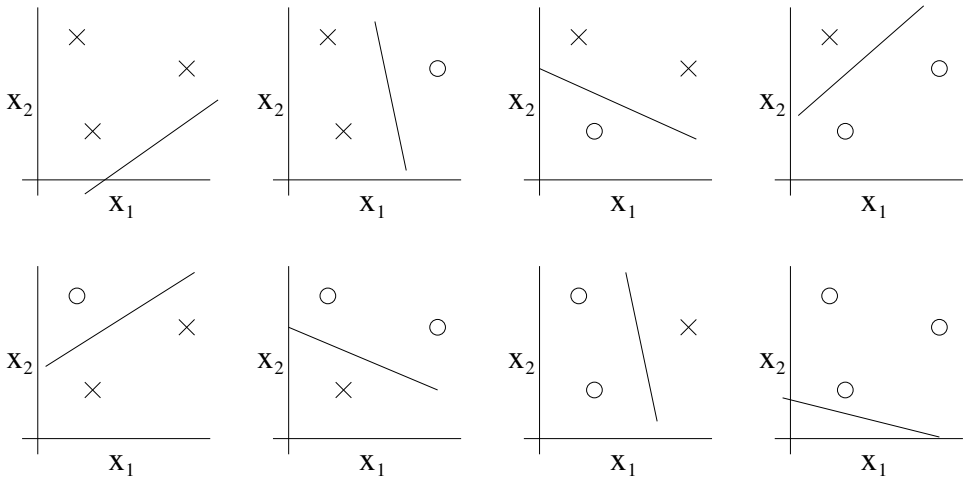
Given a set $S = \{x^{(i)}, \ldots, x^{(\mathbf{D})}\}$ (no relation to the training set) of points $x^{(i)} \in \mathcal{X}$, we say that $\mathcal{H}$ **shatters** S if $\mathcal{H}$ can realize any labeling on S. I.e., if for any set of labels $\{y^{(1)}, \ldots, y^{(\mathbf{D})}\}$, there exists some $h \in \mathcal{H}$ so that $h(x^{(i)}) = y^{(i)}$ for all $i = 1, \ldots \mathbf{D}$.

Given a hypothesis class $\mathcal{H}$, we then define its **Vapnik-Chervonenkis dimension**, written $\text{VC}(\mathcal{H})$, to be the size of the largest set that is shattered by $\mathcal{H}$. (If $\mathcal{H}$ can shatter arbitrarily large sets, then $\text{VC}(\mathcal{H}) = \infty$.)

For instance, consider the following set of three points:

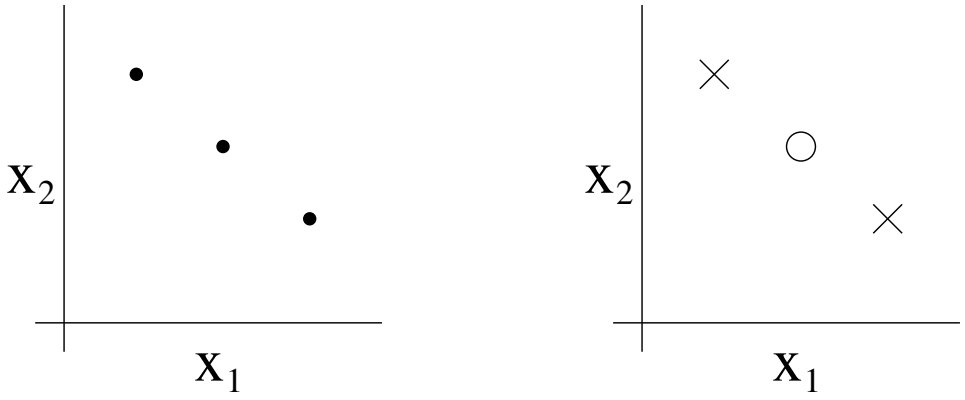


Can the set $\mathcal{H}$ of linear classifiers in two dimensions ($h(x) = 1\{\theta_0 + \theta_1 x_1 + \theta_2 x_2 \geq 0\}$) can shatter the set above? The answer is yes. Specifically, we see that, for any of the eight possible labelings of these points, we can find a linear classifier that obtains "zero training error" on them:



Moreover, it is possible to show that there is no set of 4 points that this hypothesis class can shatter. Thus, the largest set that $\mathcal{H}$ can shatter is of size 3, and hence $\text{VC}(\mathcal{H}) = 3$.

Note that the VC dimension of $\mathcal{H}$ here is 3 even though there may be sets of size 3 that it cannot shatter. For instance, if we had a set of three points lying in a straight line (left figure), then there is no way to find a linear separator for the labeling of the three points shown below (right figure):



In order words, under the definition of the VC dimension, in order to prove that VC($\mathcal{H}$) is at least $\mathbf{D}$, we need to show only that there's at least *one* set of size $\mathbf{D}$ that $\mathcal{H}$ can shatter.

The following theorem, due to Vapnik, can then be shown. (This is, many would argue, the most important theorem in all of learning theory.)

Theorem. Let $\mathcal{H}$ be given, and let $\mathbf{D} = \text{VC}(\mathcal{H})$. Then with probability at least $1 - \delta$, we have that for all $h \in \mathcal{H}$,

$$|\varepsilon(h) - \hat{\varepsilon}(h)| \leq O\left(\sqrt{\frac{\mathbf{D}}{n}\log\frac{n}{\mathbf{D}} + \frac{1}{n}\log\frac{1}{\delta}}\right).$$

Thus, with probability at least $1 - \delta$, we also have that:

$$\varepsilon(\hat{h}) \leq \varepsilon(h^*) + O\left(\sqrt{\frac{\mathbf{D}}{n}\log\frac{n}{\mathbf{D}} + \frac{1}{n}\log\frac{1}{\delta}}\right).$$

In other words, if a hypothesis class has finite VC dimension, then uniform convergence occurs as n becomes large. As before, this allows us to give a bound on $\varepsilon(h)$ in terms of $\varepsilon(h^*)$. We also have the following corollary:

Corollary. For $|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$ to hold for all $h \in \mathcal{H}$ (and hence $\varepsilon(\hat{h}) \leq \varepsilon(h^*) + 2\gamma$) with probability at least $1 - \delta$, it suffices that $n = O_{\gamma,\delta}(\mathbf{D})$.

In other words, the number of training examples needed to learn "well" using $\mathcal{H}$ is linear in the VC dimension of $\mathcal{H}$. It turns out that, for "most" hypothesis classes, the VC dimension (assuming a "reasonable" parameterization) is also roughly linear in the number of parameters. Putting these together, we conclude that for a given hypothesis class $\mathcal{H}$ (and for an algorithm that tries to minimize training error), the number of training examples needed to achieve generalization error close to that of the optimal classifier is usually roughly linear in the number of parameters of $\mathcal{H}$.