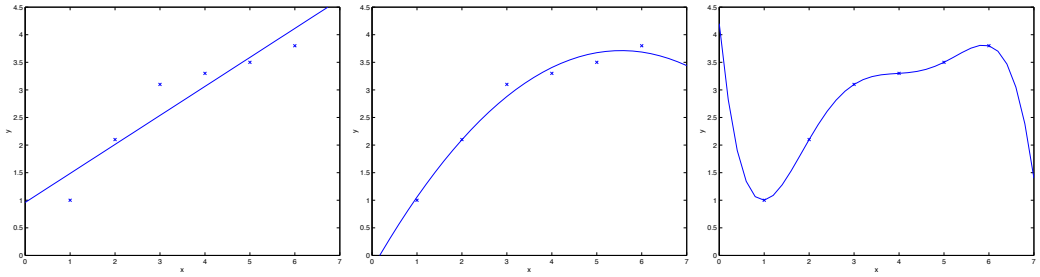
Part VI
Learning Theory

1 Bias/variance tradeoff

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 saw the following example:



Fitting a 5th order polynomial to the data (rightmost figure) did not result in a good model. Specifically, even though the 5th order polynomial did a very good job predicting y (say, prices of houses) from x (say, living area) for the examples in the training set, we do not expect the model shown to be a good one for predicting the prices of houses not in the training set. In other words, what's has been learned from the training set does not *generalize* well to other houses. The **generalization error** (which will be made formal shortly) of a hypothesis is its expected error on examples not necessarily in the training set.

Both the models in the leftmost and the rightmost figures above have large generalization error. However, the problems that the two models suffer from are very different. If the relationship between y and x is not linear,

then even if we were fitting a linear model to a very large amount of training data, the linear model would still fail to accurately capture the structure in the data. Informally, we define the **bias** of a model to be the expected generalization error even if we were to fit it to a very (say, infinitely) large training set. Thus, for the problem above, the linear model suffers from large bias, and may underfit (i.e., fail to capture structure exhibited by) the data.

Apart from bias, there's a second component to the generalization error, consisting of the **variance** of a model fitting procedure. Specifically, when fitting a 5th order polynomial as in the rightmost figure, 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. This could be, say, because in the training set we just happened by chance to get a slightly more-expensive-than-average house here, and a slightly less-expensive-than-average house there, and so on. By fitting these "spurious" patterns in the training set, we might again obtain a model with large generalization error. In this case, we say the model has large variance.[1]

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); if it is too "complex" and has very many parameters, then it may suffer from large variance (but have smaller bias). In the example above, fitting a quadratic function does better than either of the extremes of a first or a fifth order polynomial.

2 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? The 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

[1] In these notes, we will not try to formalize the definitions of bias and variance beyond this discussion. While bias and variance are straightforward to define formally for, e.g., linear regression, there have been several proposals for the definitions of bias and variance for classification, and there is as yet no agreement on what is the "right" and/or the most useful formalism.

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 sums of the probabilities of the k different events.

Lemma. (Hoeffding inequality) Let $Z_1, \ldots, Z_m$ be m 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 $\hat{\phi} = (1/m) \sum_{i=1}^{m} Z_i$ be the mean of these random variables, and let any $\gamma > 0$ be fixed. Then

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

Chebyshev inequality (For X random variable with E(X) and VAR(X)): P(|X-E(X)|>= lambda) <= var(X)/lambda^2

This lemma (which in learning theory is also called the **Chernoff bound**) says that if we take $\hat{\phi}$—the average of m 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 m 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 m times and calculate the fraction of times that it came up heads, that will be a good estimate of ϕ with high probability (if m 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, including regression and multi-class classification, problems.

We assume we are given a training set $S = \{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$ of size m, 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

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

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.[2]

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 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, $\mathcal{H} = \{h_\theta : h_\theta(x) = 1\{\theta^T x \geq 0\}, \theta \in \mathbb{R}^{n+1}\}$ 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:

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

[2]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.

3 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

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

Thus, $\hat{\varepsilon}(h_i)$ is exactly the mean of the m 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

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

This shows that, for our particular h_i, training error will be close to generalization error with high probability, assuming m 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 for 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 show that, for any particular A_i, it holds true that $P(A_i) \leq 2 \exp(-2\gamma^2 m)$. Thus, using the union bound, we

have that

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

If we subtract both sides from 1, we find that

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

(The "$\neg$" symbol means "not.") So, with probability at least $1-2k\exp(-2\gamma^2 m)$, 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 m 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: m, 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 m be before we can guarantee that with probability at least $1 - \delta$, training error will be within γ of generalization error? By setting $\delta = 2k\exp(-2\gamma^2 m)$ and solving for m, [you should convince yourself this is the right thing to do!], we find that if

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

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 m 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 m 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 $h \in \mathcal{H}$,

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

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 m, δ be fixed. Then with probability at least $1-\delta$, we have that

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

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 m 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

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

4 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 $m \geq O\left(\frac{1}{\gamma^2} \log \frac{2^{64d}}{\delta}\right) = O\left(\frac{d}{\gamma^2} \log \frac{1}{\delta}\right) = O_{\gamma,\delta}(d)$. (The γ, δ subscripts are to 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're going to do is try to 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.)

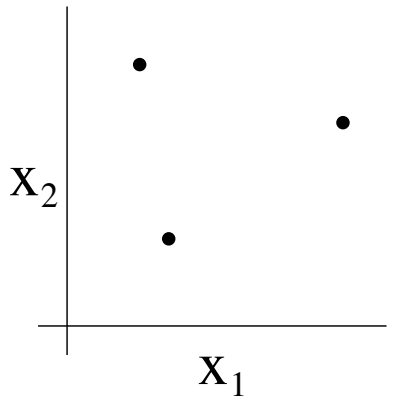
The other part of our previous argument that's slightly unsatisfying is that it relies on the parameterization of $\mathcal{H}$. Intuitively, this doesn't seem like it should matter: We had written the class of linear classifiers as $h_\theta(x) = 1\{\theta_0 + \theta_1 x_1 + \cdots \theta_n x_n \geq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

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

Given a set $S = \{x^{(i)}, \ldots, x^{(d)}\}$ (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 $\{y^{(1)}, \ldots, y^{(d)}\}$, there exists some $h \in \mathcal{H}$ so that $h(x^{(i)}) = y^{(i)}$ for all $i = 1, \ldots d$.

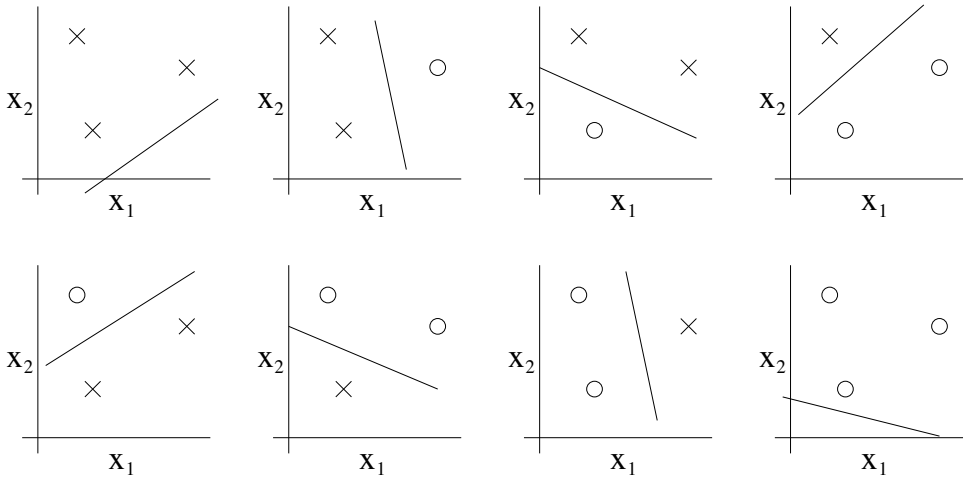
Given a hypothesis class $\mathcal{H}$, we then define its **Vapnik-Chervonenkis dimension**, written $\text{VC}(\mathcal{H})$, to be the size of the largest set that is shattered by $\mathcal{H}$. (If $\mathcal{H}$ can shatter arbitrarily large sets, then $\text{VC}(\mathcal{H}) = \infty$.)

For instance, consider the following set of three points:



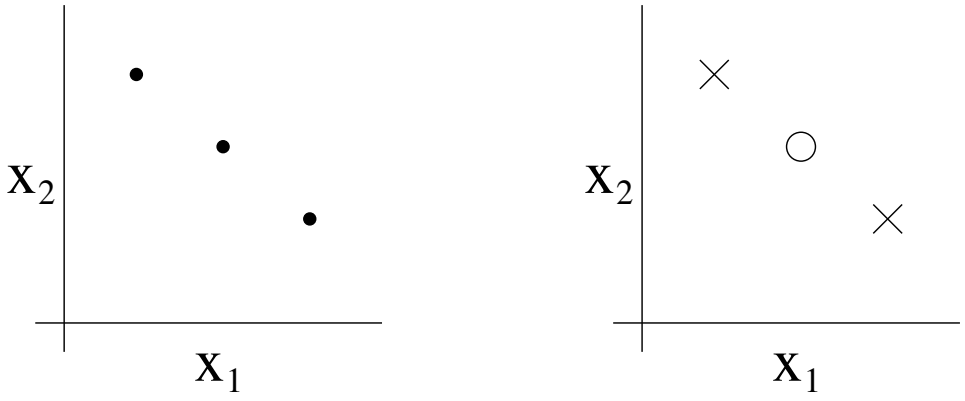
Can the set $\mathcal{H}$ of linear classifiers in two dimensions ($h(x) = 1\{\theta_0 + \theta_1 x_1 + \theta_2 x_2 \geq 0\}$) can shatter the set above? The answer is yes. Specifically, we

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 $\text{VC}(\mathcal{H})$ is at least d, we need to show only that there's at least *one* set of size 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 $d = \mathrm{VC}(\mathcal{H})$. Then with probability at least $1 - \delta$, we have that for all $h \in \mathcal{H}$,

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

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

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

In other words, if a hypothesis class has finite VC dimension, then uniform convergence occurs as m 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 $m = O_{\gamma,\delta}(d)$.

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 an algorithm that tries to minimize training error) the number of training examples needed is usually roughly linear in the number of parameters of $\mathcal{H}$.

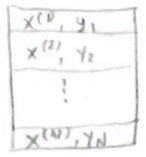
LEARNING THEORY

For algorithms we will cover, we will operate under assumptions:

① ∃ data distribution

$(x^{(i)}, y_i) \sim D$ → train / test } sample is iid

② Each $(x^{(i)}, y_i)$ are independent



$\sim D \Rightarrow$ Learning Algorithm $\Rightarrow \hat{h}$ or $\hat{\theta}$

- S is a random variable
- A deterministic function
- In statistics, the learning algorithm is the estimator
- random variable
- In statistic, sampling distr

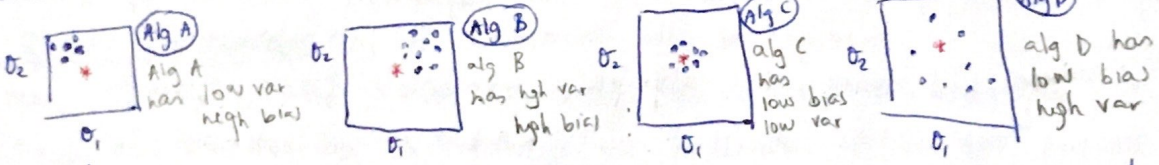
The true parameter is θ^* or h^*, which is unknown constant.

Bias and Variance

Data View

To get a more formal understanding of bias and variance:

Parameter View. Recal ∃ θ^*, *



Number of points • is the number of times we ran the learning algorithm

The above is purely statistics:

Bias: We have $\hat{\theta}$. We want an unbiased estimator $E(\hat{\theta}) = \theta$

Variance: We want $Var(\hat{\theta})$ to be "small"

As $N \to \infty$, $Var(\hat{\theta}) \to 0$. "statistical efficiency" } Nice properties for $\hat{\theta}$. Note MLEs have these properties. They are efficient and consistent

As $N \to \infty$, $\hat{\theta} \to \theta$. "statistical consistency" }

Asymptotic Evaluations

Consistency and Efficiency

"Letting the sample size increase without bound ("asymptopia") should not be ridiculed as merely a fanciful exercise. Rather, asymptotics cover the most fundamental properties of a procedure and give us a powerful and general evaluation tool" Casella & Berger

Consistency

This property requires that the estimator converge to the "correct value" as sample size becomes infinite.

Defn A sequence of estimators $W_n = W_n(X_1, \dots, X_n)$ is a consistent sequence of estimator of the parameter θ if $\forall \varepsilon > 0$ and $\theta \in \Theta$,

$$\lim_{n\to\infty} P(|W_n - \theta| < \varepsilon) = 1 \iff W_n \to \theta \text{ in probability}$$

Now, remember Markov's Inequality (If X is nonnegative r.v and $a > 0$ $P(X \geq a) \leq \frac{E(X)}{a}$)

$$P_\theta(|W_n - \theta| \geq \varepsilon) = P_\theta(|W_n - \theta|^2 \geq \varepsilon^2) \leq \frac{E((W_n - \theta)^2)}{\varepsilon^2}$$

So if $\lim_{n\to\infty} E((W_n - \theta)^2) = 0$ we have that W_n is consistent

But $\lim_{n\to\infty} E((W_n - \theta)^2) = 0 \Rightarrow \lim_{n\to\infty} \text{Var}(W_n) + \text{Bias}(W_n)^2 = 0$

Thm If W_n is a sequence of estimators of θ ∋

① $\lim_{n\to\infty} \text{Var } W_n = 0$ ② $\lim_{n\to\infty} \text{Bias } W_n = 0$ then W_n is consistent for θ

Efficiency

The property of consistency is concerned with the asymptotic accuracy of an estimator. Does it converge to the parameter it is estimating? Efficiency is concerned with the asymptotic variance of an estimator.

Defn The limiting variance or limit of the variances of an estimator T_n is τ^2 ∋ $\lim_{n\to\infty} T_n = \tau^2$

Defn The asymptotic variance or variance of the limit distribution of T_n is σ^2 where σ^2 is the variance of the limit distribution $k_n(T_n - \tau(\theta)) \to N(0, \sigma^2)$.

Note that the asymptotic variance ≠ limiting variance of sample means and others types of averages are equal. But this is not always the case

In the spirit of Cramér-Rao Lower Bound there is an optimal asymptotic variance

Defn A sequence of estimators W_n is asymptotically efficient for $\tau(\theta)$ is $\sqrt{n}(W_n - \tau(\theta)) \to N(0, v(\theta))$ in distr and

$$v(\theta) = \frac{(\tau'(\theta))^2}{E\left(\left(\frac{\partial}{\partial\theta}\log f(X|\theta)\right)^2\right)}$$

i.e, the asymptotic variance of W_n achieve the Cramér Rao Lower Bound

★ Under certain regularity conditions MLEs are consistent and efficient

Asymptotic Normality & consistency

Efficiency is only defined when the estimator is asymptotically normal and asymptotic normality implies consistency

Implications:

- An efficient estimator is asymptotically normal
- An efficient estimator is consistent
- If an estimator is consistent, we can't say anything about asymptotical normality or efficiency
- An estimator that is asymptotically normal is consistent

Pf// Sps W_n is asymptotically normal, this implies,

$$\frac{\sqrt{n}(W_n - \mu)}{\sigma} \to Z \text{ in distribution}$$

By applying Slutsky's Theorem, $W_n - \mu = \frac{\sigma}{\sqrt{n}} \cdot \frac{\sqrt{n}(W_n - \mu)}{\sigma} \to N(0, (\frac{\sigma}{\sqrt{n}})^2)$ in distr for fixed n. Now letting $n \to \infty$ $W_n - \mu \to 0$ in distribution

Convergence in distribution to a point is equivalent to convergence in probability so W_n is a consistent estimator of μ. ▨

Some further implications:

As seen above, if W_n is an estimator, W_n is asymptotically normal if $\sqrt{n}\left(\frac{W_n - \mu}{V}\right) \to Z \Rightarrow W_n \to N\left(\mu, \left(\frac{V}{\sqrt{n}}\right)^2\right)$ and as seen above an asymptotically normal estimator is consistent and the asymptotic variance and limiting variance is 0.

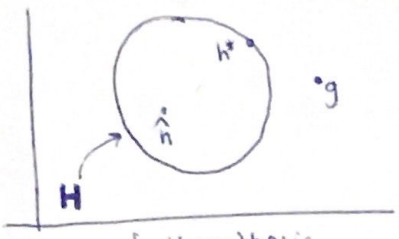
This for fixed finite n.

How to fight high variance in machine learning algorithms (estimators):

① Regularization

② Increase sample size

↳ However, this only makes sense if the estimator has a desirable property of either asymtotic normality, consistency, or efficiency. Then we can say that as $n \to \infty$ the asymptotic variance tends to 0.



space of Hypothesis

g - Best possible hypothesis

h^* - Best in class H

$\hat{h}$ - Learnt from finite data

$\varepsilon(h)$: risk / generalization error = $E_{(x,y)\sim D}[1\{h(x) \neq y\}]$

↑ thinking this from a classification perspective

↑ it generalizes (to training data)

$\hat{\varepsilon}(h)$: empirical risk / error = $\frac{1}{m}\sum_{i=1}^{m} 1\{h(x^{(i)}) \neq y_i\}$

$\varepsilon(g)$ = Bayes Error / Irreducible error : the error of best possible hypothesis

$\varepsilon(h^*) - \varepsilon(g)$ = Approximation error (the price we are paying for limiting ourselves to some class, diff b/w best possible error and best possible error subject to class)

$\varepsilon(\hat{h}) - \varepsilon(h^*)$ = Estimation error.

$$\varepsilon(\hat{h}) = \varepsilon(\hat{h}) - \varepsilon(h^*) + \varepsilon(h^*) - \varepsilon(g) + \varepsilon(g)$$

= estimation error + approximation error + irreducible error

estimation = variance + estimation = bias

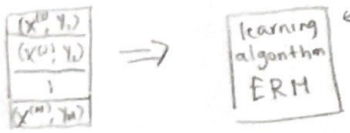
$$\varepsilon(\hat{h}) = \underbrace{\text{variance}}_{\text{estimation}} + \underbrace{\text{bias}}_{\text{estimation}} + \underbrace{\text{approximation error}} + \underbrace{\text{irreducible error}}_{\text{irreducible}}$$

How to fight high bias in Machine learning algorithms (estimator)

① Make **H** bigger.

ERM Emp Risk Minimization

- It is a learning algorithm



We are limiting ourselves to empirical risk minimizers.

$$\hat{h}_{ERM} = \arg\min_{h \in H} \hat{\varepsilon}(h) = \arg\min_{h \in H} \frac{1}{m}\sum_{i=1}^{m} 1\{h(x^{(i)}) \neq y_i\}$$

Uniform Convergence

① If we do empirical risk minimization, i.e, if we minimize training error, what does this say about generalization error $\hat{\varepsilon}(h)$ vs. $\varepsilon(h)$.

② How does the generalization error of our trained hypothesis compare to the generalization of the best hypothesis $\varepsilon(\hat{h})$ vs $\varepsilon(h^*)$

We will answer these questions using the following tools.

Tools

① subadditivity of measure

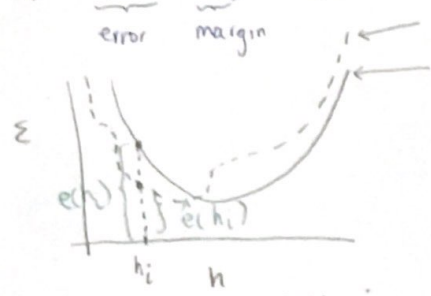
$$P\left(\bigcup_{i \in I} A_i\right) \leq \sum_{i \in I} P(A_i)$$

② Chernoff Inequality

Let $Z_1, Z_2, \dots, Z_m \sim \text{Ber}(\phi)$ i.i.d. Let $\hat{\phi} = \frac{1}{m}\sum_{i=1}^{m} Z_i$. Let $\gamma > 0$ [margin]

$$P_\phi\left(|\hat{\phi} - \phi| > \gamma\right) \leq 2e^{-2\gamma^2 m}$$

(error, margin)



$\hat{\varepsilon}_s(h)$, given sample data s, training error of every possible $h \in H$

$\varepsilon(h)$, the generalization error of every possible $h \in H$

Now Consider $|S| = m$, $(x^{(1)}, y_1), (x^{(2)}, y_2), \dots, (x^{(m)}, y_m) \in S$

Let $Z_i = 1\{h_i(x^{(i)}) \neq y_i\}$ for $i = 1, 2, \dots, m$ Then

$E(Z_i) = \varepsilon(h_i)$ and Z_i i.i.d $\text{Ber}(\varepsilon(h_i))$

Applying Chernoff's Inequality: $P\left(|\hat{\varepsilon}(h_i) - \varepsilon(h_i)| > \gamma\right) \leq 2e^{-2\gamma^2 m}$

As $m \to \infty$, $P\left(|\hat{\varepsilon}(h_i) - \varepsilon(h_i)| \leq \gamma\right) \approx 1$. we can let γ be very small

However, we don't just want to guarantee that $\varepsilon(h_i)$ close to $\hat{\varepsilon}(h_i)$ with high prob for just one h_i but $\forall\, h \in H$

Finite Hypothesis case, $H = \{h_1, h_2, \dots, h_k\}$, $|H| = k$

$$P\left(\forall h \in H \mid |\hat{\varepsilon}(h) - \varepsilon(h)| < \gamma\right) \geq 1 - \underbrace{2k e^{-2\gamma^2 m}}_{\delta}$$

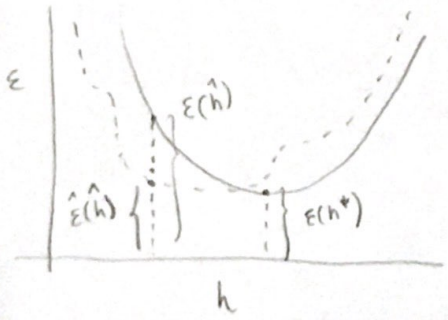
Let δ - prob of error, γ - margin of error, m - sample size.

So you can obtain relationships b/w δ, γ, m b/c of chernoff's Inequality

For ex: fix $\gamma, \delta > 0$, $m \geq \frac{1}{2\gamma^2}\log\left(\frac{2k}{\delta}\right)$. So with prob at least $1-\delta$ the margin of error will be less than γ as long as $m \geq \frac{1}{2\gamma^2}\log\frac{2k}{\delta}$

Called sample complexity

Now we understand $\hat{\varepsilon}(h)$ vs $\varepsilon(h)$. Let's turn to $\varepsilon(\hat{h})$ vs $\varepsilon(h^*)$



Notes