MSRI's *Mathematics of Machine Learning* Summer School

Nico Courts

July 29-Aug 9, 2019

Introduction

These notes were the ones I took while attending MSRI's "Mathematics of Machine Learning" summer school at the University of Washington during the two weeks above.

1 Statistical Learning

This series of lectures was given by Robert Shapire from Microsoft Research.

The topic of the talks here are somewhere at hte nexus of supervised and statistical learning. The big idea concerning what we want o do is to learn how to do better in the future based on experience from the past. The hope is that these methods can be fully automated. This is a proper subset of AI, but definitely a core area of how we do it.

We are looking for easy to use, flexible algorithms that give us useful results. Ideally they would be interpretable, but this is usually a secondary result.

He believes that ML can help us to understand how learning happens in non-machine entities (cool!). For instance, one can consider the question of nature vs nuture and to ask the question "what is simplicity and why is it useful?"

1.1 The Problem

In this course, we will be focusing on a single kind of (simple) learning problem: learning from example. Given a set of examples, we are looking for an algorithm that can classify instances given objects.

For instance, character recognition. We have a set of handwritten characters and the letter it depicts. Then we want to feed this into a learning algorithm that gives us a prediction rule, that given a handwritten character, outputs the predicted character.

We were then given a couple of examples to try. The particulars weren't important, but the takeaway here is

- We needed enough data to say something meaningful.

- We looked for a rule that fit the observations. We want the rule to be consistent or to contain few mistakes.
- We were looking for rules that were "simple." Importantly, our notion of simplicity changed when we converted one set of numbers to binary.

The first condition is always the case: "more is more." But the latter two fit into a tradeoff between fit and simplicity.

The main questions we will be trying to answer:

- How much data do we need?
- How do we define simplicity?
- How much complexity do we need to represent our problem?
- What is the tradeoff mentioned above?

1.2 Studying the Problem Mathematically

We have to come up with a formal model for the problem so that we can do mathematics with it. The learning model should answer some basic questions: *What is the goal of learning?* and *How is the learning happening?*.

We begin the description of the model:

1.2.1 Definition: An **instance** (or sometimes informally **example**) x is an object in a space called the **instance space** or **domain** X.

1.2.2 Remark: To each instance we assign a **label** or **class**. In this course we will be simplifying to only consider two classes, denoted 0 and 1 or $+$ and $-$.

1.2.3 Remark: We work under an assumption that there is an underlying function

$$c : X \to \{0, 1\}$$

called the (target) **concept** that we are trying to learn. Later we will relax this assumption.

1.2.4 Definition: We define the **hypothesis** to be (similar to the concept) a map

$$h : X \to \{0, 1\}$$

that holds our current beliefs.

We write $\mathcal{C}$ to denote the **concept class**, the class of all possible functions we are exploring (we assume we know a class to which c belongs). We will further assume that our data were generated *independently* and at random. It is limiting to make an assumption about the distribution from which these were drawn. So we will just say there is one.

1.2.5 Definition: We define

$$\text{err}_D(h) = \Pr_{x\sim D}[h(x) \neq c(x)]$$

to be the (generalization) **error**.

Then we obviously ask that $\text{err}_D(h) < \varepsilon$ for some small ε, but furthermore we want that this happens with high probability:

$$\Pr[\text{err}_D(h) \leq \varepsilon] \geq 1 - \delta$$

and in this case, we say that that the rule is probably approximately correct (PAC).

1.2.6 Definition: The class $\mathcal{C}$ is **PAC-learnable** by $\mathcal{H}$ (a hypothesis space) if there is an algorithm A such that for all $c \in \mathcal{C}$ and for all distributions D and for all positive ε and δ, then A takes "polynomial many" random variables $X_i \sim D$ which outputs $h \in \mathcal{H}$ such that

$$\Pr[\text{err}_D(h) \leq \varepsilon] \geq 1 - \delta$$

1.2.7 Remark: This is not an easy problem, per se, but it is one of the most simple classes of algorithms to work with.

Example 1.1
Let $X = \mathbb{R}$ and let $\mathcal{C}$ be of the form c_r for $r \in \mathbb{R}$ where

$$c(x) = \begin{cases} 1, & x \geq r \\ 0 & \text{otherwise} \end{cases}$$

so $\mathcal{C}$ is the *set of positive half lines*. Then if we have some data, we can set our hypothesis $h = c_b$ where b is the leftmost positive value. Then if $c = c_r$, then $\text{err}_D(h)$ has to do with the interval $[r, b]$. Let $b - r = \varepsilon$.

Now restrict attention to $X = [0, 1]$ and notice that if we got a training point within ε of c (to the right), then we would have gotten a smaller difference. Thus

$$\Pr[\text{err}_D(h) < \varepsilon] \leq \Pr[\text{no } x_i \text{ are in } [b, b + \varepsilon]]$$

$$
\begin{aligned}
&= \prod_1^m \Pr[x_i \in R] \\
&= (1-\varepsilon)^m \\
&\leq e^{-\varepsilon m}
\end{aligned}
$$

using the fact that $1 + x \leq e^x$ for all x.

1.3 Sufficient conditions for learning

Now we consider the case when our hypothesis space $\mathcal{H}$ is finite.

1.3.1 Theorem
Let A be an algorithm that finds a hypothesis $h_A \in \mathcal{H}$ which is consistent with m random training examples (as before) where

$$m \geq \frac{1}{\varepsilon}(\ln|\mathcal{H}| + \ln\frac{1}{\delta})$$

then $\Pr[\text{err}_D(h_A) > \varepsilon] \leq \delta$.

1.3.2 Remark: Note that we have completely done away with trying to get a handle on the concept class here.

1.3.3 Remark: Equivalently, with probability of $1-\delta$,

$$\text{err}_D(h_A) \leq \frac{\ln|\mathcal{H}| + \ln|\frac{1}{\delta}|}{m}$$

The idea to take away here is that $\ln|\mathcal{H}|$ is in some way measuring "description length" of a hypothesis: you have to describe how to distinguish the correct one from the wrong ones. Think popping these in a binary tree. Then the m bound gives you some sort of way to manage the complexity of the algorithm.

1.3.4 Theorem
Assume we have $m \geq \frac{1}{\varepsilon}(\ln|\mathcal{H}| + \ln|\frac{1}{\delta}|)$ examples. Then with probability $1 \geq 1-\delta$, for all $h \in \mathcal{H}$ if h is consistent then $\text{err}_D(h) \leq \varepsilon$ (here we often write "h is ε-good.")

Proof
We prove the converse: the probability of there being an $h \in \mathcal{H}$ that is consistent and ε-bad is less than δ. Notice that whether h is consistent with the training set is a random variable (since it depends on choice of training set). Whether it is ε-bad is not random. So if $\mathcal{B}$ is the set of ε-bad hypotheses, then we really want

$$\Pr[\exists h \in \mathcal{B} : h \text{ is consistent}] \leq \sum_{g \in \mathcal{B}} \Pr[h \text{ is consistent}]$$

using the union bound.

Now fix $g \in \mathcal{B}$ and we want to compute the probability of it being consistent. That is, $h(x_i) = c(x_i)$ for all i. Since the samples are independent, this probability is

$$\prod_1^m \Pr[h(x_i) = c(x_i)] \leq (1-\varepsilon)^n$$

Therefore the original probability is

$$|\mathcal{B}|(1-\varepsilon)^m \leq |\mathcal{H}|e^{-\varepsilon m} \leq \delta$$ ♠

1.4 When does this work when we have infinite hypothesis classes?

We saw in the first example that infinite classes can still be PAC-learnable, but where is the line drawn?

To see this, consider classifications of finite subsets. But any choice of threshold between any two points on $\mathbb{R}$ will give the same classification! So really we only have something like 5 different hypothesis (in the form of a threshold function at least). Compare that with 2^m behaviors one could consider in general!

More formally, if we are given a set $S = \langle x_1, \ldots, x_m \rangle$ of instances, then we can consider the collection of all $\langle h(x_1), \ldots, h(x_m) \rangle$ for all $h \in \mathcal{H}$. We call this set $\Pi_{\mathcal{H}}(S)$ and define

$$\Pi_{\mathcal{H}}(m) = \max_{|S|=m} |\Pi_{\mathcal{H}}(S)|$$

and call this the **growth function.** We can use this function to get some sort of handle on the complexity of the problem, and in fact it plays a similar role in an analogous theorem to the one we saw earlier:

1.4.1 Proposition
Give m training examples, with probability $1-\delta$ for all $h \in \mathcal{H}$ if h is consistent then

$$\text{err}_D(h) \leq \mathcal{O}\left(\frac{\ln \Pi_{\mathcal{H}}(2m) + \ln \frac{1}{\delta}}{m}\right)$$

1.4.2 Remark: The proof of this result can't quite continue in the same way as before because we need to select a training set *before we can know the hypothesis space*, so the dependence gets in the way. There is a fancier proof but we won't do it here (although there will be another proof later that is similar).

1.4.3 Remark: Notice that if $\Pi_{\mathcal{H}}(m)$ is polynomial, this reduces nicely and as m gets larger, the error drops. Thus learning is possible.

1.4.1 More on the growth function

There are at least some cases when $\Pi_{\mathcal{H}}(m) \in \mathcal{O}(m^d)$ for some constant d. In the worst case, $\Pi_{\mathcal{H}}(m) = 2^m$. In fact, these are the only two cases that can happen! Furthermore, it ends up that these two cases correspond exactly to when learning is possible!

To see why, we need a new definition:

1.4.4 Definition: A sample S of size m is **shattered** by $\mathcal{H}$ if all possible behaviors/labelings are possible: that is, $\Pi_{\mathcal{H}}(S) = 2^m$.

Example 1.2
Given $\mathcal{H}$ to be the set of all closed intervals $[a, b] \subseteq \mathbb{R}$, any set of two points in $\mathbb{R}$ is shattered by $\mathcal{H}$. But you cannot shatter a set of three points in $\mathbb{R}$.

1.4.5 Definition: The **Vapnik-Chervinenkis (or VC) dimension** is

$$\text{VCdim}(\mathcal{H}) = \max\{|S| : S \text{ is shattered by } \mathcal{H}\}$$

1.4.6 Remark: So in the exercise above, VCdim(intervals) = 2.

Another set of hypotheses are the linear threshold functions in $\mathbb{R}^n$, and the VC dimension here is $n + 1$. If you fix your planes to be subspaces, the VC dimension is n.

1.4.7 Remark: A good heuristic measure for VC dimension is the number of parameters. There are pathological examples, but they are truly that.

We have a nice little lemma:

1.4.8 Lemma (Sauer)
If $d = \text{VCdim}(\mathcal{H})$, then

$$\Pi_{\mathcal{H}}(m) \leq \sum_{0}^{d} \binom{m}{i}.$$

1.4.9 Remark: A "nicer" upper bound leverages

$$\sum \binom{m}{i} \leq \left(\frac{em}{d}\right)^d$$

whenever $m \geq d \geq 1$.

Plugging this into the bound on the error of a hypothesis, we get that (forgetting the relatively small log term) that the VC dimension d gives us a complexity measure—that is, it gives a bound on the error of a consistent hypothesis.

1.5 Non-consistent Hypotheses

The results so far have been great, but has always assumed that there is a consistent hypothesis to find. But what if our data is noisy? Then learning the concept is no longer about finding the concept that matches *all* the given data.

Now we consider data $(x, y) \sim D$ to be a pair jointly sampled from some distribution D. We do basically the same thing to compute error:

$$\operatorname{err}_D(h) = \Pr_{(x,y)\sim D}[h(x) \neq y].$$

We are still trying to minimize the generalization error $\operatorname{err}(h)$ over $\mathcal{H}$ given some sample (x_i, y_i) sampled from D.

1.5.1 Definition: Write the **empirical error** as

$$\widehat{\operatorname{err}}(h) = \frac{1}{m}\sum_{1}^{m} \delta_{h(x_i)\neq y_i}$$

And then we can follow the route called *empirical risk minimization* (ERM) that finds an empirical solution:

$$\hat{h} = \operatorname{argmin}_{h\in\mathcal{H}} \widehat{\operatorname{err}}(h).$$

We want to get to the situation in which, with probability greater than $1-\delta$, for all $h \in \mathcal{H}$,

$$|\operatorname{err}(h) = \widehat{\operatorname{err}}(h)| \leq \varepsilon.$$

This result is a kind of **uniform convergence result** which is the kind of thing we are after. Obviously this is useful since it allows us to bridge the gap between imperfect information and the true concept.

Now $\delta_{h(x_i)\neq y_i} = 1$ with probability $\operatorname{err}(h)$ and zero otherwise.

Example 1.3
A quick aside: Consider IID variables $Z_1, \ldots, Z_m$ with $Z_i \in [0, 1]$. Let $p = \mathbb{E}[Z_i]$ and then let $\hat{p} = \frac{1}{m}\sum_i Z_i$ be the empirical mean.

Then there is a result called **Hoeffding's inequality**, which claims

$$\Pr[\hat{p} \geq p + \varepsilon] \leq e^{-2e\varepsilon^2 m}.$$

Notice that what this says is that the empirical average converges to its expected value. So we can think of this as a function converging

$$f(Z_1, \ldots, Z_m) \to \mathbb{E}[f(Z_1, \ldots, Z_m)].$$

This idea leads us to:

1.5.2 Lemma (McDiarmid's Inequality)
Suppose $f(z_1, \ldots, z_m)$ is real valued such that chainging z_i changes f by at most c_i, and furthermore the z_i are independent, but not necessarily identically distributed. Then you get a nice bound on the error (I didn't get it in time but I am sure it is online.)

So if we let $Z_i = \delta_{h(x_i)\neq y_i}$ and let $p = \mathbb{E}Z_i = \operatorname{err}(h)$ and $\hat{p} = \widehat{\operatorname{err}}(h)$, then for any particular $h \in \mathcal{H}$,

$$\Pr[|\operatorname{err}(h) - \widehat{\operatorname{err}}(h)| \geq \varepsilon] \leq 2e^{-2\varepsilon^2 m}.$$

So when $\mathcal{H}$ is finite, we can use the union bound to say

$$\Pr[\exists h \in \mathcal{H} : |\operatorname{err}(h) - \widehat{\operatorname{err}}(h)| \geq \varepsilon] \leq 2|\mathcal{H}|e^{-2\varepsilon^2 m}$$

then setting the RHS equal to δ, we get that with probability $1-\delta$, for all $h \in \mathcal{H}$,

$$|\operatorname{err}(h) - \widehat{\operatorname{err}}(h)| \leq \mathcal{O}\left(\sqrt{\frac{\ln|\mathcal{H}| + \ln\frac{1}{\delta}}{m}}\right)$$

or in another (slightly weaker but sometimes more suggestive) form:

$$\operatorname{err}(h) \leq \widehat{\operatorname{err}}(h) + \mathcal{O}\left(\sqrt{\frac{\ln|\mathcal{H}| + \ln\frac{1}{\delta}}{m}}\right)$$

1.5.3 Remark: Notice that this really (finally) ties together how several aspects impact learning: it relates the true error to the sample size, the complexity class of the concept, and the consistency of the hypothesis with the sample data.

1.5.4 Remark: Notice that as the complexity of your hypotheses increase, we expect the empirical error to decrease, but if we assume that the above is truly a good bound for $\operatorname{err}(h)$, we would expect that the true error will begin by dropping, but eventually the growth term from $\ln|\mathcal{H}|$ will overtake things and cause the error to rise again.

This is one way to think about the concept of **overfitting**: the complexity of your hypothesis functions can incorporate more and more pathological functions that are less likely to be realistic.

1.6 Another Complexity Measure

Suppose we have a labelled sample S consisting of (x_i, y_i), where we can now assume the label space is ± 1. Then if we have a hypothesis, we can compute the training error $\widehat{\operatorname{err}}(h)$:

$$\widehat{\operatorname{err}}(h) = \frac{1}{m}\sum_i \delta_{h(x_i)\neq y_i} = \frac{1}{m}\sum_i \frac{1 - y_i h(x_i)}{2} = \frac{1}{2} - \frac{1}{2}\left(\frac{1}{m}\sum_i y_i h(x_i)\right)$$

and the term in parentheses is another quantity we can consider which is just a constant away from training error.

So we can consider computing

$$\mathbb{E}_\sigma[\max_{h\in\mathcal{H}} \sum_i \sigma_i h(x_i)]$$

where σ_i are ± 1 equally likely. In the case that $|\mathcal{H}| = 1$, we get that this value is zero and when $\mathcal{H}$ shatters S (the set of all m-tuples of ± 1), the value is 1. These serve as the two extremes, obviously.

We would like to work with this towards describing convergence (and rates thereof) towards the generalization error. To do this, we need to broaden the context slightly. Let $f : Z \to \mathbb{R}$ (standing in for the hypothesis), a space $\mathcal{F}$ ($\mathcal{H}$), and define the **(empirical) Rademacher complexity** to be

$$\hat{R}_{\mathscr{F}}(S) = \mathbb{E}_\sigma \left[\sup_{f\in\mathscr{F}} \frac{1}{m} \sum_i \sigma_i f(z_i) \right]$$

and the expected Rademacher complexity is

$$R_{\mathscr{F}}(m) = \mathbb{E}_S[\hat{R}_{\mathscr{F}}(S)]$$

Then we want to show that for all $f \in \mathscr{F}$ that $\hat{\mathbb{E}}_S[f] \to \mathbb{E}[f]$. Specifically,

1.6.1 Theorem

Let $\mathscr{F}$ be a family of functions $f : Z \to [0, 1]$ and let S be a collection of z_i where $z_i \sim D$. Then with probability $1 - \delta$, and for all $f \in \mathscr{F}$,

$$\mathbb{E}[f] \leq \hat{\mathbb{E}}_S[f] + 2 \cdot \Psi + \mathcal{O}\left(\sqrt{\frac{\ln(1/\delta)}{m}}\right)$$

where Ψ is either $\hat{R}_{\mathscr{F}}(S)$ or $R_{\mathscr{F}}(m)$.

Proof

Let $\Phi(S) = \sup_{f\in\mathscr{F}}(\mathbb{E}[f] = \mathbb{E}_S[f])$. Then first we show (via McDiarmid) that

$$\Phi(S) \leq \mathbb{E}_S[]\Phi(S)] + \mathcal{O}\left(\sqrt{\frac{\ln 1/\delta}{m}}\right).$$

So then we have reduced the problem with finding some bound for $\mathbb{E}_S[\Phi(S)]$.

Then let S' to be a hypothetical second sample (called the "ghost sample"). Compute

$$\mathbb{E}_S[\Phi(S)] = \mathbb{E}_S[\sup_f(\mathbb{E}[f] - \hat{\mathbb{E}}_S[f])] \leq \mathbb{E}_{S,S'}[\sup(\hat{\mathbb{E}}_{S'}[f] - \hat{\mathbb{E}}_S[f])]$$

There is something to prove in the last line there but he didn't show us. He said there was a slide somewhere. :)

For the next step, go through the two samples S and S' one index at a time and for each i with equal probability either swap z_i and z'_i or do nothing. Call the resulting samples T and T'. But then the following two expressions have the same distribution:

$$\hat{\mathbb{E}}_{S'}[f] - \hat{\mathbb{E}}_S[f] \quad \text{and} \quad \hat{\mathbb{E}}_{T'}[f] - \hat{\mathbb{E}}_T[f]$$

and so if we let σ_i denote the random variable for whether we swapped at the i^{th} step ($+1$ if we swapped z_i and z'_i and -1 otherwise), the latter is

$$\frac{1}{m}\sum_i \sigma_i f(f(z'_i) - f(z_i))$$

so we get

$$\mathbb{E}_{S,S'}[\sup_f(\hat{\mathbb{E}}_{S'} - \hat{\mathbb{E}}_S[f])] = \mathbb{E}_{S,S'}\left[\sup_f\left(\frac{1}{m}\sum_i \sigma_i f(f(z'_i) - f(z_i))\right)\right]$$

Finally we can show (another slide) that this value is at most $2R_{\mathscr{F}}(m)$ and then use McDiarmid's once more to show that the expected Rademacher complexity is close enough to that of the empirical RC. ♠

Now the whole point was to show that $\text{err}(h) \approx \widehat{\text{err}}(h)$. One can show the empirical Rademacher complexity of the class of loss functions comprised of indicator functions $l(x, y) = \delta_{h(x)\neq y}$ is twice that of $\mathcal{H}$ itself. You can easily plug this in to get $\mathbb{E}[l_h]$. Then you can get what you want.

Now if $|\mathcal{H}| < \infty$, (careful, he made a notation error earlier so I am switching subscripts and arguments in Rademacher complexities–they are still the same)

$$\hat{R}_S(\mathcal{H}) \leq \sqrt{\frac{2\ln|\mathcal{H}|}{m}}.$$

If the space is infinite, then we choose a representative hypothesis for each behavior seen in applying all hypotheses to S. Call this space $\mathcal{H}'$ and notice that $|\mathcal{H}'| = |\Pi_{\mathcal{H}}(S)|$. Then one can show that the empirical Rademacher complexity of $\mathcal{H}'$ is the same as that of $\mathcal{H}$!

1.7 Weak Learning

At the end of the last lecture we discussed the ideal of weak vs. strong learning. In weak learning, we only require that there is an algorithm yielding a hypothesis h such that $\text{err}_R(h) \leq \frac{1}{2} - \gamma$ for some $\gamma > 0$. But surprise!

1.7.1 Theorem

A class is strongly learnable if and only fif it is weakly learnable.

Today we will discuss why. We want to convert any algorithm that weakly learns a class into one that strongly learns the class. The idea here is that a class $\mathcal{C}$ is either completely (PAC) learnable or else there is no algorithm that learns the concept any better than random chance.

1.7.1 The Ada Boost Algorithm

We will assume we are given a weak learning algorithm A and data $(x_1, y_1), \ldots, (x_m, y_m) \sim \mathscr{D}$. We want to find a hypothesis H whose error $\text{err}_{\mathscr{D}}(H)$ is arbitrarily small.

1.7.2 Definition: A theorem that can take a weak learning algorithm and convert it into an ε-good hypothesis is called a **boosting algorithm.**

The basic idea is what one would expect; run A a total of T times, extracting weak hypotheses $h_1, \ldots, h_T$ and somehow combine these in a way to get a strong hypothesis H. There is a problem with just feeding the same data in to A over and over (e.g. A may be deterministic). So instead we need to find a way of creating distributions $D_1, \ldots, D_T$ that "focus in" on the bad/difficult examples that the weak hypotheses fail on.

1.7.3 Remark: Notice that the definition of learnability means that there has to be an algorithm that works **for all true distributions**. So for instance, the fact that we can learn something about the weather, but can't get it ε-good. This is because our algorithms only work well (enough) on a particular distribution.

Let $\varepsilon_t = \text{err}_{D_t}(h_t) = \frac{1}{2} - \gamma_t$ and notice $\gamma_t \geq \gamma > 0$. Then define the following distributions: D_1 will be the uniform distribution, so the weight on the i^{th} training datum: $D(i) = \frac{1}{m}$. Then define the rest iteratively:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t}, & h_t(x_i) = y_i \\ e^{\alpha t}, & \text{otherwise} \end{cases} = \frac{D_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

where Z_t is a normalization constant and

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right) > 0$$

Then we output H which is defined as

$$H(x) = \text{sign} \left(\sum_1^T \alpha_t h_t(x) \right).$$

That is, we let a weighted sum (according to the error on the provided distribution) tells us what to select.

1.7.4 Theorem

$$\widehat{\text{err}}(h) \leq \prod_1^T 2\sqrt{\varepsilon_t(1 - \varepsilon_t)} = \prod \sqrt{1 - 4\gamma_t^2} \leq \exp\left(-2 \sum \gamma_t^2 \right)$$

So if $\gamma_t > \gamma$, we get

$$\widehat{\text{err}}(h) \leq e^{-2\gamma^2 T}$$

Proof
He went too fast. Slides are somewhere supposedly. ♠

So we get a bound on training error, great! Let's turn it into something about generalization error: if $h_t \in \mathcal{H}$ and $d = \text{VCdim}(\mathcal{H})$, then with probability $1 - \delta$ (see problem session)

$$\text{err}(H) \leq \widehat{\text{err}}(H) + \widetilde{\mathcal{O}}\left(\sqrt{\frac{Td + \ln 1/\delta}{m}}\right)$$

where $\widetilde{\mathcal{O}}$ means that we ignore log factors.

Then something that may be surprising is that as T grows, we may expect that eventually we will overfit our training data, but in practice you can often run this very far out and just keeping better results! in fact, your output H will continue to get better even after it achieves zero error on the training set (since the underlying h_t are still incorrect).

But this points to the issue: the test error doesn't tell the whole story. Instead we are interested in measuring the *confidence* the algorithm has in the combined classifier. So how do we do that? Recall that to determine H we held an "election" for the best output. What we are interested in here is the **margin** (the difference between the fraction of votes for and against an output): we call the margin the *weighted fraction of the h_t's correct minus the weighted fraction of the h_t's incorrect.*

Doing some math, you get

$$\text{margin} = y \cdot \frac{\sum \alpha_t h_t(x)}{\sum \alpha_t}$$

Since we are about out of time, we are sprinting to the finish line: Doing an analysis, we can show that this algorithm tends to push the margins towards the positive. Then you prove that this phenomenon tends to create better generalization performance. Note that we can actually remove the dependence on T by doing the following:

$$\text{err}(H) \leq S + \widetilde{\mathcal{O}}\left(\frac{d/\theta^2 + \ln 1.\delta}{m}\right)$$

where S is the fraction of training examples with margin $\leq \theta$.

2 Convex Optimization

This series was given by Sebastien Bubeck from Microsoft Research.

2.1 Fundamentals

Let $K \subseteq \mathbb{R}^n$ be a convex set: that is one such that the straight line segment between two points in K is contained entirely in K.

2.1.1 Definition: The map $f : K \to \mathbb{R}$ is called **a convex function** if

$$f((1-\gamma)x + \gamma y) \leq (1-\gamma)f(x) + \gamma f(y).$$

2.1.2 Remark: Then if f is differentiable, this implies

$$\frac{f(x+\gamma(y-x)) - f(x)}{\gamma} \leq f(y) - f(x)$$

and as $\gamma \to 0$, the LHS tends to $\nabla f(x) \cdot (y-x)$. Thus

$$f(y) \geq f(x) + \nabla f(x) \cdot (y-x).$$

Then the goal of this course is to "find" $\operatorname{argmin}_{x \in K} f(x)$. A question we ask ourselves: how are f and K specified? Mostly in these lectures we will be focusing on *simple* K and functions f such that $\nabla f(x)$ can be computed (for any x).

2.2 Examples in machine learning

2.2.1 Regression

Here we have a dataset: $(a_i, y_i) \in \mathbb{R}^n \times \mathbb{R}$ are the elements. Then we want to find a rule that assigns y to $\mathbf{a}$. We focus on *linear* rules. That is, we are looking for a linear space in $\mathbb{R}^{n+1}$ that approximates the "true" values.

Tentatively we let our function be $a \mapsto x \cdot a$, parameterized by $x \in \mathbb{R}^n$. Then for each x we can evaluate how the tentative function fits our data. More specifically:

$$\frac{1}{m} \sum_{1}^{m} l(x \cdot a_i, y_i)$$

where l is some loss function.

There are several loss functions one might consider:
Least Squares:

$$l(u, v) = (u - v)^2$$

The upshot here, however, is that the evaluation function above in this case is convex! The idea here is that we are applying a linear function to a convex function. The other nice thing is that if you make a modeling assumption about the distribution underlying the label distribution, nice things happen.

Specifically, assume that $y \sim \mathcal{N}(a \cdot \underline{x}, \sigma^2)$ where $\underline{x}$ is the *true value*. Notice we haven't put any assumption on the draw distribution, just the relationship between the (fixed) a_i and y given the truth x. One can compute

$$\frac{1}{2\pi\sigma^2} \exp(-\frac{1}{2\sigma}(y_i - a_i \cdot x)^2).$$

Thus the likelihood of the entire data set is the product:

$$\frac{1}{(2\pi\sigma^2)^{m/2}}\exp(-\frac{1}{2\sigma^2}\sum_1^m(y_i - a_i\cdot x)^2)$$

and so we recover the least squares loss function as the objective we want to minimize. That is, the maximum likelihood estimator minimizes least squares fit.

2.2.2 Classification

Now restrict $y \in \{\pm 1\}$. A natural loss is

$$l(u,v) = \delta_{\text{sign}(u)\neq\text{sign}(v)}.$$

This is a problem for this class since it is non-convex.

So instead we define **support vector machines** using the loss

$$l(u,v) = \max(0, 1-uv)$$

which is convex

Another example is the **logistic loss**

$$l(u,v) = \log(1-e^{-uv})$$

which you can see is a smooth upper bound on the 0-1 loss we started with. Minimizing this function is equivalent to minimizing the MLE for the *logistic model:*

Denote $p(a) = \mathbb{P}(+1|a)$. Then we assume that the "scale" of our probability is given by a linear function:

$$\log\left(\frac{p(a)}{1-p(a)}\right) = \underline{x}\cdot a \Leftrightarrow p(a) = \frac{1}{1+e^{-\underline{x}\cdot a}}$$

and then the **logistic loss** is the negative log-likelihood of the logistic model.

2.2.3 Graphical Models

Given $a_1,\ldots,a_m \in \mathbb{R}^n$, we want to infer the correlation structure of the variables (how are they all related).

Let's start from a (Gaussian) model: we assume we are drawing IID samples from $\mathcal{N}(\mu,\underline{\Sigma})$. A fact: if $(\Sigma^{-1})_{ij} = 0$, then x_i and x_j are independent, conditioned on the rest. Thus the goal is to estimate Σ^{-1}. So if the truth is Σ, the density of $a_1,\ldots,a_n$ is

$$\frac{1}{\sqrt{(2\pi)^n\det(\Sigma)}}\exp(-\frac{1}{2}(a_i-\mu)^T\Sigma^{-1}(a_i-\mu))$$

and by taking the negative log likelihood:

$$c + \frac{m}{2}\log\det\Sigma + \frac{1}{2}\sum_1^m(a_i-\mu)^T\Sigma^{-1}(a_i-\mu) = C + \frac{m}{2}\left(-\log\det\Sigma^{-1} + \text{tr}(\Sigma^{-1}S)\right)$$

where c is some constant and S is the sample covariance matrix. Then to estimate Σ^{-1} via MLE, we want to solve

$$\text{argmin}_{X>0} - \log\det(X) + \text{tr}(XS)$$

where this is convex in X (this is not obvious but we will see it in the problem session).

If you know that the true matrix is sparse, you may add a penalty term $\lambda\|X\|_1$. Similarly in regression you may add either $\|x\|_1$ or $\|x\|_2^2$. In either case, these add curvature to your objective function. This may speed up the optimization process.

2.2.4 Unsupervised Learning

Finally we want to talk about *clustering*. Notice that we have been slowly changing our dataset. First we dropped labels and now we are going to represent our dataset as a graph $G = (V, E)$ of interactions. We would like to infer some partitioning of the vertices.

As a modelling assumption, we use the stochastic block model. There is a hidden partitioning $\sigma \in \{-1, 1\}^{|V|}$. G is generated from the distribution

$$(i,j) \in E \text{ with probability } \begin{cases} p & \sigma_i = \sigma j \\ q, \text{otherwise} \end{cases}$$

where we assume $q < p$. Furthermore we assume all connections are independent.

The the likelihood of $x \in \{-1, 1\}^n$ is given as follows: let $A = (a_{ij})$ be the adjacency matrix of G. Then the likelihood is

$$\prod_{ij}\left[\frac{1+x_ix_j}{2}\left(pa_{ij} + (1-p)(1-a_{ij}) + \frac{1-x_ix_j}{2}(qa_{ij}(1-q)(1-a_{ij}))\right)\right]$$

this is non-convex! But we can look at a convex upper bound and we're good! Next time we'll talk about gradient descent.

2.3 Gradient Descent

This is the most simple algorithm we will study (also one of the most used ones). It is an iterative method where we define

$$x_{t+1} = x_t - \eta\nabla f(x_t)$$

which dates back to the mid 1800's (likely by Cauchy). Notice that this is similar to Newton's method, but the latter requires inverting an $n \times n$ matrix and this was an attempt to avoid that difficult computation.

Why does this work? By Taylor's theorem, $f(y) \approx f(x) + \nabla f(x) \cdot (y - x)$. The idea here is that the negative $\nabla f(x)$ direction is in the direction of the minimum, so we just follow that.

Now if we have a convex function, where $f(x) - f(y) \leq \nabla f(x) \cdot (x - y)$, we can set $y = x^*$ and define

$$\delta(x) := f(x) = f(x^*) \leq \nabla f(x) \cdot (x^* - x) = -\nabla f(x) \cdot (x^* - x)$$

then the idea is that progress in the $x^* - x$ direction (that is towards the target) is bounded below by $\delta(x)$. That is to say: *the rate of decrease of the instance to OPT is lower bounded by the suboptimal gap.*

There is a small problem, namely that we also take a (hopefully small) step away from the direct-line path from x to x^*. To fix this, we use continuous time analysis:

$$\frac{d}{dt}x(t) = -\nabla f(x(t))$$

which we call the **gradient flow.** Then

$$\frac{d}{dt}\frac{1}{2}\|x(t) - x^*\|^2 = \left(\frac{d}{dt}(x(t) - x^*)\right) \cdot (x(t) = x^*) = -\nabla f(x(t)) \cdot (x(t) - x^*) \leq -\delta(x(t))$$

and integrating over time:

$$\int_0^\tau \frac{1}{2}\|x(t) - x^*\|^2 \mathrm{d}t \leq \int_0^\tau \delta(x(t))\mathrm{d}t$$

and using the FTOC and dropping the τ term (and normalizing):

$$\frac{1}{\tau}\int_0^\tau \delta(x(t))\mathrm{d}t \leq \frac{1}{2\tau}\|x(0) - x^*\|^2 := \frac{r_0^2}{2\tau}$$

and so finally

$$\delta\left(\frac{1}{\tau}\int_0^\tau x(t)\mathrm{d}t\right) \leq \frac{r_0^2}{2\tau}.$$

Of course this isn't a discrete process, so it's useless for coding! So we want to run gradient descent with step size η for T steps and approximate this integral with a finite sum from 1 to T. Then if we further impose that f is Lipshitz ($\|\nabla f(x)\| \leq L$) then the error in this approximation is at worst ηL^2. So if we are trying to measure the total error of our discretized algorithm from the true value, we get an error bound

$$\frac{r_0^2}{T\eta} + \eta L^2 \leq \frac{r_0 L}{\sqrt{T}}$$

for a "good" choice of $\eta = \frac{r_0}{L\sqrt{T}}$.

2.3.1 Discrete Time Analysis

2.3.1 Theorem

Let $g_1, \ldots, g_T$ be arbitrary vectors in $\mathbb{R}^n$ and that $x_{t+1} = x_t - \eta g_t$. Then for any $x \in \mathbb{R}^n$,

$$\sum_1^T g_t(x_t - x)le\frac{\|x_1 - x\|^2}{2\eta} + \eta\sum_1^T \|g_T\|^2$$

2.3.2 Corollary
If $g_t = \nabla f(x_t)$ (f convex) and $\|nablaf(x_t)\| \leq L$, then

$$f\left(\frac{1}{T}\sum_1^T x_t\right) - f(x^*) \leq \frac{1}{T}\sum_1^T (f(x_t) - f(x^*)) \leq \frac{1}{T}\sum \nabla f(x_t)(x_t - x^*) \leq \frac{r_n^2}{2\eta T} + \eta L^2$$

The proof here is really just one line of computation but I missed it. Interestingly, the proof gives us *equality* rather than inequality. I suppose that it's nice to use it as if it were a bound.

2.3.2 Projective Gradient Descent

A generalization is optimization with some constraint. So if we have a bounded convex set K, we just follow each step in gradient descent with a projection onto the closest point in K (should we leave it). This is only better for us since

2.3.3 Lemma
$\|P_K(x) - z\| \leq \|x - z\}$ for any $z \in K$.

2.3.4 Remark: Notice that we are assuming here that both K and f are convex. Without convexity we definitely don't have this.

2.4 Stochastic Gradient Descent

We begin with a corollary of the theorem we just proved for standard GD:

2.4.1 Corollary
Say that g_t is a random variable such that $\mathbb{E}[g_t|x_t] = \nabla x_t$ and that $\mathbb{E}[\|g_t\|^2] < B^2$. Then

$$\mathbb{E}[f\left(\frac{1}{T}\sum_1^T x_t\right) - f(x^*)] \leq \frac{r_1^2}{2\eta} + \frac{\eta}{2}TB^2 \leq \frac{r_1 B}{\sqrt{T}}$$

given an optimal choice of η.

So then we can use this result to define SGD, Stochastic Gradient Descent. Let $f(x) = \mathbb{E}_{(a,y)}l(x,(a,y))$. The idea here is that we don't have the actual gradient of the loss function, but we have access (under the statistical learning framework) to a continuous stream of IID data. So we define the **stochastic gradient**:

$$g_t = \nabla_x l(x_t, (a, t, y_t))$$

that is, compute the x gradient of $l(x,(a_t,y_t))$ and evaluate at x_t. Then basically this pos right into the above corollary.

2.4.2 Remark: here the notation varies from statistical learning, so think of x as our hypothesis (the thing we're optimizing over) and (a,y) denotes a sample from the dataset.

Note that here you only get one pass on the data (although here you can directly control the generalization error as opposed to bounding it with testing error in the other class).

Instead, we can do multi-pass SGD where at each step we sample uniformly data points from your dataset (possibly having much more steps than there are data points) but then your function converges to an unbiased estimator for the test error rather than the generalization error.

3 Bandits

This series was given by Kevin Jamieson, a professor in the CSE department here at UW.

3.1 Introduction

The core idea here (motivated very well by his description of the WSJ front page (ads, content, etc) and google maps (that does bandit-like testing to find new good routes) that balances the exploration (data-gathering) phase and the application phase and the transiton between them. It more resembles a conversation than a one-time transaction.

The name comes from slot machines, obviously. :) We are thinking of these problems as having a slot machine with n "arms", which can be pulled to yield some data. So the input to the algorithm is the number n and for $t = 1, 2, \ldots$, the algorithm pulls some arm $I_t \in [n]$ and nature reveals a corresponding reward $r_t \sim P_{I_t}$ where $\mathbb{E}[r_t | I_t] = \theta^*_{I_t}$.

3.1.1 Regret Minimization

We define the **regret** as follows:

3.1.1 Definition: After T timesteps, the regret is

$$R_T = \max_i \theta^*_i T - \mathbb{E}\left[\sum_1^T r_t\right]$$

Then the goal is to achieve "sublinear regret", that is get that $R_T \in \mathbf{o}(T)$.

3.1.2 Lemma

$$\begin{aligned} R_T &= \max_i \theta^*_i T - \mathbb{E}[\sum_1^T \sum_1^n r_t \delta_{I_t=i}] \\ &= \max \theta^*_i T - \sum_1^n \mathbb{E}[\sum +1^T \theta^*_i \delta_{I_t=i}] \end{aligned}$$

$$
\begin{aligned}
&= \max \theta_i^* T - \sum_1^n \theta_i^* \mathbb{E}[T_i] \\
&= \sum_2^n (\theta_1^* - \theta_i^*)\mathbb{E}[T_i] \\
&= \sum_2^n \Delta_i \mathbb{E}[T_i].
\end{aligned}
$$

3.2 Another game

This leads us to the concept of **best-arm identification**: we fix some $\delta \in (0,1)$ and we identify $\operatorname{argmax}_i \theta_i^*$ with probability $1 = \delta$ with as few as possible total pulls. This time we don't put an upper bound on the time you're given, and instead your job is to identify what the best strategy is.

3.2.1 Definition: A random variable X is **R-sub-Gaussian** if

$$\mathbb{E}[\exp(\lambda(x))] \leq \exp(\lambda R^2/2)$$

3.2.2 Remark: Then by Hoeffding, if $X \in [a,b]$ and letting $R^2 = \frac{(b-a)^2}{8}$ and notice when $X \sim \mathcal{N}(0,1)$, we get $R^2 = 1$.

Then we can develop the Chernoff bound: Suppose X_i are IID and 1-sub-Gaussian. Then

$$
\begin{aligned}
\mathbb{P}(\sum_1^n X_i > \varepsilon) &= \mathbb{P}(\exp(\lambda \sum X_i) > \exp(\lambda\varepsilon)) \\
&\leq e^{-\lambda\varepsilon}\mathbb{E}[\exp(\lambda \sum X_i)] \\
&= e^{-\lambda\varepsilon} \prod \mathbb{E}[e^{\lambda X_i}] \\
&\leq e^{-\lambda\varepsilon}(e^{\lambda^2/2})^n \\
&\leq e^{-n\varepsilon^2/2}
\end{aligned}
$$

where the last step "comes from calculus". So by letting $\delta = e^{-n\varepsilon^2/2}$, one can solve for epsilon and get that with probability $1-\delta$, $\frac{1}{n}\sum X_i \leq \sqrt{\frac{2\ln 1/\delta}{n}}$.

If $n = 2$, say $\theta_1^*\theta_2^*$ and $\Delta = \theta_1^* - \theta_2^*$. If both arms are pulled τ times, let the event

$$\mathcal{E}_i = \left\{ \left| \theta_i^* - \frac{1}{\tau}\sum_{t=1}^{\tau} X_{i,t} \right| \leq \sqrt{\frac{2\ln(4/\delta)}{\tau}} \right\}$$

for $i = 1, 2$ (these are the "good events"), then $\mathbb{P}(\mathcal{E}_i^C) \leq \delta/2$, so

$$\mathbb{P}(\mathcal{E}_1^C \cup \mathcal{E}_2^C) \leq \sum \mathbb{P}(\mathcal{E}_i^C) \leq \delta.$$

Then if $\tau = 8\Delta^{-2}\ln(4/\delta)$, we can say

$$\hat{\theta}_2 - \hat{\theta}_2 = \hat{\theta}^1 - \theta_1^* - \hat{\theta}_2 + \theta_2^* + \theta_1^* - \theta_2^* > 0$$

with probability $1 - \delta$. This of course means that we have accurately determined which arm to pull.

Now consider the regret:

$$R_T = \theta_1^* t = \mathbb{E}[\theta_1^* T_1 + \theta_2^* T_2] = (\theta_1^* - \theta_2^*)\mathbb{E}[T_2] = \Delta\mathbb{E}[T_2]$$

from which we can compute

$$\Delta\mathbb{E}[T_2] = \Delta\mathbb{E}[T_2\delta_{\mathcal{E}_1\cap\mathcal{E}_2} + T_2\delta_{\mathcal{E}_1^C\cup\mathcal{E}_2^C}] \leq \Delta\tau + \Delta T\mathbb{P}[\mathcal{E}_1^C \cup \mathcal{E}_2^C]$$

and so if we set $\delta = \frac{1}{T}$,

$$R_T \leq 8\Delta^{-1}\log(8T) + \Delta$$

But notice that if we just always pick the bad arm, the worst regret we can get is $R_T \leq \Delta T$. So

$$R_T \leq 1 + \min 8\Delta^{-1}\log(8T), \Delta T \leq 1 + \sqrt{8T\ln(8T)}$$

If you were to graph the minimum function we had there, you'd notice that the regret tends towards zero near zero and towards infinity. The idea here is that if there is only a small difference between the optimal and suboptimal choice, then guessing randomly will only incur a tiny regret. Likewise, if the means are very far apart, you will quickly detect this through testing. There is a spike at about $1/\sqrt{T}$ that is the place where both ideas miss the mark.

3.3 Extending to more Arms

We are going to start by writing down an algorithm and then trying to analyze it:

3.3.1 Algorithm

We input n arms and confidence δ. Initialize $\hat{X}_1 = \{1, \ldots, n\}$. Then for $l = 1, 2, \ldots$ we let $\mathcal{E}_l = 2^{-l}$ and

$$\tau_l = \left\lceil 2\mathcal{E}_l^{-2}\ln\frac{4l^2 n}{\delta} \right\rceil$$

Then pull each arm in $\hat{X}_l$ τ_l times and compute $\hat{\theta}_{i,l}$. Then define

$$\hat{X}_{l+1} = \hat{X}_l \setminus \left\{ i \in \hat{X}_l : \max_{j\in\hat{X}_l} \hat{\theta}_{j,l} - \hat{\theta}_{i,l} > 2\varepsilon_l \right\}$$

3.3.2 Analysis

Define $\mathcal{E}_{i,l} = \{|\hat{\theta}_{i,l} - \theta_i^*| \leq \varepsilon_l\}$ and let $\mathcal{E} = \cap_{i,l}\mathcal{E}_{i,l}$. But then

$$\mathbb{P}(\mathcal{E}_{i,l}^C) \leq 2\mathbb{P}(\hat{\theta}_{i,l} - \theta_i^* > \varepsilon_l) \leq 2\exp -\tau_l\varepsilon_l^2/2) \leq \frac{\delta}{2l^2n}$$

and so $\mathbb{P}(\mathcal{E}^c) \leq \delta$.

3.3.1 Definition: Let $\Delta_i = |\theta_i^* - \theta_1^*|$ for all $i > 1$ and define $\Delta_1 = \Delta_2$ (the minimum (nonzero) gap), where we assume for notational simplicity that $1 \geq \theta_1^* > \theta_2^* \geq \cdots \geq \theta_n^* \geq 0$.

3.3.2 Lemma
With probability greater or equal to $1 - \delta$, we have $1 \in \hat{X}_l$ and

$$\max_{i \in \hat{X}_l} \Delta_i \leq 8\varepsilon_l$$

for all l.

3.3.3 Remark: The key thing to notice in the proof here is the fact that for any $j \in \hat{X}_l$,

$$\hat{\theta}_{j,l} - \hat{\theta}_{1,l} = (\hat{\theta}_{j,l} - \theta_j^*) - (\hat{\theta}_{1,l} - \theta_1^*) - \Delta_j \leq 2\varepsilon_l - \Delta_j \leq 2\varepsilon_l$$

Now let's talk about regret:

$$R_T = \mathbb{E}\left[\sum_1^n \Delta_i T_i\right]$$

and so for any $\nu > 0$,

$$\sum_1^n \Delta_i T_i \leq \nu T + \sum_{l:8\varepsilon_l \geq \nu} 8\varepsilon_l\tau_l|\hat{X}_l| \leq \nu T + \sum_{l-1}^{\lceil \lg \max(\Delta,\nu)^{-1}\rceil} 8\varepsilon_l\tau_l\delta_{\Delta_i \leq 8\varepsilon_l}$$

and then doing some more massaging, we get

$$\sum \Delta_i T_i \leq \nu T + \sum_1^n c(\max(\Delta_i,\nu)^{-1}) \log\left(\frac{n\log(\max(\Delta_i,\nu)^{-1})}{\delta}\right)$$

3.3.4 Remark: Notice there are some serious limitations here. In particular, it requires prior knowledge of T, which we don't always have.