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.

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.