7 Section Week 7 - Intro to Learning Theory

Based roughly on Kearns and Vazirani - Computational Learning Theory

First, some discussion since the theory is really meant to capture a real phenomenon. What does it mean to learn something?

Discuss

- Relationship between training data and test data? What if in CS121 we spent all of lecture talking about circuits and then asked about Turing Machines on the exams?
- How do we represent what we "think" we're learning? If we learn about biology from a textbook do we actually have a textbook with all the concepts in our mind?
- How can error occur?
- Some simple learning algorithms e.g. process of elimination?

We form a representation that we think behaves similarly to the concept we're trying to learn, based on our training data.

Some conclusions - the test data set has to come from the same distribution as the sample data set. Our representation for a concept we're learning doesn't have to be in exactly the same form. We need to account for error in two ways - sample data is unlucky and not representative and in all other cases, we are fairly close to correct.

7.1 First Example - Learning an interval

We introduce ϵ, δ type arguments, which in learning theory are always (something similar to) of the form:

Our algorithm learns the concept if using some m samples, with probability $1-\delta$ it finds a hypothesis which correctly classifies a test data point with error at most ϵ. An efficient algorithm requires that $m = poly(\frac{1}{\delta}, \frac{1}{\epsilon})$.

We consider some instance space X, which is just all objects in the world, and a concept c is an element of a concept class C and is a subset $c \subseteq X$. We receive samples $x \sim_{\mathcal{D}} X$ drawn from distribution $\mathcal{D}$ and are tested on samples also drawn from distribution $\mathcal{D}$. Our samples are labelled $(x, c(x))$ to tell us if they're in the concept or not. We use the notation $c(x) = 1 \iff x \in c$.

Big picture - a concept is analogous to a language, where we can think of it as a subset of the instance space or as a function that labels instances.

A learning algorithm outputs a hypothesis $h \subseteq X$ where $h(x) = c(x)$ "most of the time" (with the appropriate error parameters).

Draw symmetric difference Venn diagram

Precisely, we want symmetric difference to be less than ϵ with probability greater than $1-\delta$, taken over the randomness of the sampling and testing.

Let's see how this all comes together with a simple example: learning an interval.

The concept is some interval (a, b) of the real numbers, the instance space is the real line, the concept class is all intervals, and sample data is drawn from some arbitrary distribution $\mathcal{D}$ over $\mathbb{R}$. What are some potential learning algorithms?

Lesson - there are many possible learning algorithms, we just need to pick one and show that it learns and that suffices.

Algorithm: given samples $s_1, s_2, \ldots s_m$, output (s_i, s_j) where s_i is the smallest s which is in the interval and s_j is the largest s which is in the interval.

Theorem 1. *For every ϵ and δ, with probability at least $1-\delta$ this algorithm learns the interval with error less than ϵ.*

Proof. Draw diagram with points a, a', b', b in order on the number line

Let $b' < b$ be the value where the weight of $\mathcal{D}$ on the interval (b', b) is exactly $\frac{\epsilon}{2}$.

If we take m samples, the probability that none of them fall into (b', b) is $(1 - \frac{\epsilon}{2})^m$. Consider the interval (a, a') defined analogously on the other side of the interval (a, b). Then the probability that none of the training samples fall into either (a, a') or (b', b) is at most $2(1 - \frac{\epsilon}{2})^m$, and in this case we would have too high total error probability.

If neither side of our error has weight more than $\frac{\epsilon}{2}$ then certainly our total error is at most ϵ, so all we need is for m to be large enough so that $2(1-\frac{\epsilon}{2})^m < \delta$.

We use the inequality that $1 - x < e^{-x}$ to get

$$\begin{aligned} 2e^{-m\epsilon/2} &< \delta \\ -m\epsilon/2 &< \ln \delta/2 \\ m &> \frac{2}{\epsilon} \ln \frac{2}{\delta} \end{aligned}$$

as the number of samples we need. This is in poly of $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$ so it's fine.

□

7.2 Another Example - Learning a conjunction

A conjunction concept over n variables is of the form $c = x_1 \wedge \overline{x_2} \wedge x_5 \wedge \overline{x_7}$ etc. In other words, the AND of many literals.

Formally, let $I \subseteq [n]$ and $\ell(x_i)$ either be x_i or $\overline{x_i}$. A conjunction is of the form

$$c = \bigwedge_{i \in I} \ell(x_i)$$

In this case the instance space is $\{0, 1\}^n$, samples are of the form $x \in \{0, 1\}^n$

and $(x, c(x))$, where x is drawn from a distribution $\mathcal{D}$.

Suggest learning algorithms (hint - part of the purpose of this example is to illustrate that a learned hypothesis doesn't have to be exactly the concept).

Here's an algorithm: start with the hypothesis

$h(x) = x_1 \wedge \overline{x_1} \wedge x_2 \wedge \overline{x_2} \wedge x_3 \wedge \cdots \wedge \overline{x_n}$.

When we receive a sample of the form $(x, c(x) = 1)$, update the hypothesis h by removing each literal that evaluates to 0 on x.

Proof. The key observation here is that h will always contain a super set of the literals in c. Furthermore, if $c(x) = 1$, then we know that every literal in h that evaluates to 0 on x CANNOT be in c, else $c(x)$ would equal 0.

Now how do we analyze error? Consider some literal ℓ. We only miss the error in training if ℓ is not in c but we never saw a positive example x where $\ell = 0$, since that's the only way we remove ℓ from h.

Let $p(\ell)$ be the probability that we draw a sample x from $\mathcal{D}$ where $c(x) = 1$ and $\ell = 0$. The literal ℓ is "high error" if $p(\ell) > \frac{\epsilon}{2n}$, since if h has no "high error" literals then by the union bound our total error probability is at most $2n(\frac{\epsilon}{2n}) = \epsilon$. In other words, we don't care so much if we miss ℓ if it rarely causes problems.

Thus, we want to find m such that the probability any "high error" literal is in h after training is δ low. For a given "high error" literal ℓ, the probability that our m samples all miss the fact that ℓ is not supposed to be in h is at most $(1 - \frac{\epsilon}{2n})^m$. Union bounding over all at most $2n$ bad errors, we get that we want to find m such that

$$2n(1 - \tfrac{\epsilon}{2n}^m) < \delta$$

Doing similar analysis to before gets

$$m > \tfrac{2n}{\epsilon} \ln \tfrac{2n}{\delta}$$

Which is poly in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, $\frac{1}{n}$. $\square$

A meaningful observation here is that if x_1 is not in c but the distribution literally never picks any sample where $x_1 = 0$, then we're never going to eliminate the literal x_1. Thus, our hypothesis is going to be different from our concept, but over our training distribution this is totally ok.

Another interesting observation is that the form of m here is exactly the same as with learning an interval. This is because in both problems we union bound over some k symmetric bad results, where for intervals $k = 2$ and for conjunctions $k = 2n$.

7.3 Definition of PAC Learning and Others

The formal definition of PAC learnable (probably aprpoximately learnable) due to Les Valiant is:

A concept class C is PAC learnable if there exists an efficient algorithm L such that for any distribution $\mathcal{D}$ over the instance space, for every concept $c \in C$, and for any δ and ϵ, L takes some m samples and outputs a hypothesis h where with probability at least $1 - \delta$ the error of h is at most ϵ.

Another important term to know but we won't get into it:

The VC dimension is roughly as follows: given a set of n points, consider all 2^n possible labellings of those points. The VC dimension of our "range space" is the size of the largest set of points that can be classified in all possible ways.

For example, if our range space is $\mathbb{R}^2$ classified by half space partitions (a line dividing $\mathbb{R}^2$ into two parts), there exist sets of three points can be classified in any way but not any set of four points. Thus, since no set of four points can be classified in all possible ways, the VC dimension of $\mathbb{R}^2$ classified by half space partitions is 3.

For more on VC dimension and learning, check out chapter 14 of Probability and Computing by Mitzenmacher and Upfal.