15-850(C) Machine Learning Theory

Lecturer: Avrim Blum
Scribe: Avrim Blum

Lecture 2: September 21, 1995

---

1 The consistency model, continued

Last time we saw algorithms for learning conjunctions, k-CNF formulas and k-DNF formulas in the consistency model. We also noted that the consistency problem for k-term DNF formulas is NP-hard, even though the set of functions expressible as k-term DNFs is a subset of the set of functions expressible as k-CNFs. In what sense are k-term DNFs hard to learn? Really, what's going on is not that they can represent especially complex phemonena, but rather that forcing the learner's representation into that form can be difficult.

General DNF formulas are easy for finding consistent hypotheses. They also can express anything expressible by the other representations (in fact, they can express any boolean function). Does that mean that this learner is more powerful? What is going on?

The problem is that finding a consistent hypothesis may not have anything to do with the goal of predicting well on new examples.

A secondary problem is the consistency model doesn't handle noise very well. We could improve this to the *max consistency model*, by asking the learner to provide the concept c in C that maximizes the number of examples in S consistent with c. Or we could ask the learner to approximately maximize this quantity. Unfortunately, without any noise assumptions, many of these questions are NP-hard.

Open problem (weak agnostic learning of conjunctions): Is there a polynomial time algorithm that given a set S of labeled examples over $\{0,1\}^n$ and a guarantee that there *exists* some conjunction consistent with at least 90% of them, will produce a conjunction consistent with at least a $\frac{1}{2} + 1/poly(n)$ fraction?

2 The PAC model

The basic idea of the PAC model is to assume that examples are being provided from a fixed (but perhaps unknown) distribution over the instance space. The assumption of a fixed distribution gives us hope that what we learn based on some training data will carry over to new test data we haven't seen yet. Another nice feature of this assumption is that it provides us a well-defined notion of the error of a hypothesis with respect to target concept.

Definition 1 *Given a example distribution $\mathcal{D}$, the* error *of a hypothesis h with respect to a target concept c is* $\mathbf{Prob}_{x\in\mathcal{D}}[h(x) \neq c(x)]$. *($\mathbf{Prob}_{x\in\mathcal{D}}(A)$ means the probability of event A given that x is selected according to distribution $\mathcal{D}$.)*

Suppose we are seeing examples labeled according to some target concept in class C. What kind of guarantee could we hope to make?

- We converge quickly to the target concept (or equivalent). But, what if our distribution places low weight on some part of X?
- We converge quickly to an approximation of the target concept. But, what if the examples we see don't correctly reflect the distribution?
- With high probability we converge to an approximation of the target concept. This is the idea of **P**robably **A**pproximately **C**orrect learning.

In the following definition, "n" denotes the size of an example.

Definition 2 *An algorithm $\mathcal{A}$ PAC-learns concept class C by hypothesis class H if for any $c \in C$, any distribution D over the instance space, any $\epsilon, \delta > 0$, and for some polynomial p, the following is true. Algorithm $\mathcal{A}$, with access to labeled examples of c drawn from distribution $\mathcal{D}$ produces with probability at least $1-\delta$ a hypothesis $h \in H$ with error at most ϵ. In addition, $\mathcal{A}$ should do so after running for time (and number of examples seen) at most $p(n, 1/\epsilon, 1/\delta, size(c))$.*

The quantity ϵ is usually called the accuracy parameter and δ is called the confidence parameter. A hypothesis with error at most ϵ is often called "ϵ-good."

This definition allows us to make statements such as: "the class of k-term DNF formulas is learnable by the hypothesis class of k-CNF formulas."

Remark 1: If we require $H = C$, then this is typically called "proper PAC learning". If we allow H to be the class of polynomial time programs (i.e., we don't care what representation the learner uses so long as it can predict well) then this is typically called "PAC prediction". I will usually say: "concept class C is PAC-learnable" to mean that C is learnable in the PAC-prediction sense.

Remark 2: One nice extension of this model is instead of requiring the error of h by at most ϵ to just require that the error be at most $\frac{1}{2} - 1/poly(n)$. This is called *weak learning* and we will talk more about this later.

Remark 3: Another nice extension is to the case where H is not necessarily a superset of C. In this case, let ϵ_H be the least possible error using hypotheses from H. Now, we relax the goal to having the error of h be at most $\epsilon + \epsilon_H$. If we let C be the set of all concepts (and we remove "$size(c)$" from the set of parameters we are allowed to be polynomial in), then this is often called the *agnostic model* of learning: we simply want to find the (approximately) best $h \in H$ we can, without any prior assumptions on the target concept.

2.1 Conjunctions

Let's show that the above algorithm for learning conjunctions in the consistency model is a PAC learning algorithm for this class. There are many ways of showing this fact. Here is one particularly nice way.

1. Consider some specific "bad" conjunction whose error is at least ϵ. The probability that this bad conjunction is consistent with m examples drawn from $\mathcal{D}$ is at most $(1-\epsilon)^m$.

2. Notice that there are (only) 3^n conjunctions over n variables.

3. (1) and (2) imply that given m examples drawn from $\mathcal{D}$, the probability *there exists* a bad conjunction consistent with all of them is at most $3^n(1-\epsilon)^m$. Suppose that m is sufficiently large so that this quantity is at most δ. That means that with probability $(1-\delta)$ there are *no* consistent conjunctions whose error is more than ϵ. Since our algorithm finds a consistent conjunction whenever one exists (and we are assuming one exists here), this means that with probability at least $1-\delta$ our algorithm finds a hypothesis with error at most ϵ.

4. The final step is to calculate the value m needed to make sure that it is polynomial in our parameters (from the description of the algorithm, it is clear that its running time is polynomial so long as m is). We can do this as follows:

   $$3^n(1-\epsilon)^m \leq \delta \quad \text{is equivalent to} \quad m\ln(1-\epsilon) + n\ln(3) \leq \ln(\delta)$$

   which is equivalent to

   $$m \geq [n\ln(3) + \ln(1/\delta)]/[-\ln(1-\epsilon)]$$

   which (simplifying using $-\ln(1-x) = x + x^2/2 + x^3/3 + ... > x$) is implied by

   $$m \geq \frac{1}{\epsilon}[n\ln(3) + \ln(1/\delta)].$$

   Since this is polynomial in our parameters, we have a PAC-learning algorithm. ■

2.1.1 A lower bound

How tight is this analysis? Here we give a specific situation showing that in fact it's tight up to constant factors. This situation holds even for monotone conjunctions.

Consider the case where the target concept is the "null" conjunction that says everything is positive (if you don't like this, then let x_0 be a special variable, let the target concept be

x_0, and set x_0 to 1 in every example listed below). Consider the following distribution on examples:

Example	Probability
$1111\ldots11$	$1-2\epsilon$
$0111\ldots11$	$2\epsilon/n$
$1011\ldots11$	$2\epsilon/n$
$\vdots$	$\vdots$
$1111\ldots10$	$2\epsilon/n$

Suppose we see only $\frac{n}{4\epsilon}$ examples. In that case, the expected number of examples we see that are not the all ones example is just $n/2$. Therefore, we expect to throw out at most $n/2$ of the variables (perhaps fewer if we see the same example twice). This means the the expected error of our algorithm is at least ϵ.

Notes and Acknowledgements

The PAC model was initially proposed by Valiant [2] and a good discussion appears in [1].

References

[1] Michael Kearns, Ming Li, Leonard Pitt, and Leslie Valiant. Recent results on boolean concept learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 337–352, University of California, Irvine, June 1987.

[2] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.