Machine Learning Theory (CSC 482A/581B) - Lecture 13

Nishant Mehta

1 Strong and Weak Learnability

Over the next few lectures, we will gain a deeper understanding of PAC learning by studying boosting and, in particular, AdaBoost.

Near the beginning of this course, we defined (efficient)[1] PAC learnability as follows:

> There is an algorithm $\mathcal{A}$ such that, for any $c \in \mathcal{C}$, any distribution P over $\mathcal{X}$, for all $\varepsilon > 0$, and all $\delta \in (0, 1)$, if $\mathcal{A}$ is given access to examples drawn from P and labeled according to c, then it outputs a hypothesis $\hat{f}$ for which, with probability at least $1 - \delta$,
>
> $$\Pr_{X \sim P}\left(\hat{f}(X) \neq c(X)\right) \leq \varepsilon;$$
>
> moreover, the algorithm's runtime must be polynomial in $\frac{1}{\varepsilon}$ and $\frac{1}{\delta}$ (and also polynomial in the dimension of the input d, though we'll forego mentioning this latter requirement as it is orthogonal to our current study).

We will use the term *strong learnability* to refer to the PAC learnability of a concept class in the above sense. Consider now the following two weakenings of the requirements of strong learnability:

1. Rather than needing to handle all ε, instead the algorithm only needs to obtain risk bounded by a constant ε_0;

2. Rather than needing to handle all δ, the algorithm only needs to succeed in satisfying the risk bound with probability at least δ_0 for some constant δ_0.

Under the above two weaker requirements, we now only require the algorithm's runtime to be polynomial in d (which, again, we will not discuss further).

Clearly, any algorithm can meet the first weakened requirement for $\varepsilon_0 = 1/2$ (just flip a fair coin to predict the label), and so to make the requirement meaningful, we assume that ε_0 is better than $1/2$ by some "edge" $\gamma > 0$, so that $\varepsilon_0 = \frac{1}{2} - \gamma$.

Formally, we say that a concept class $\mathcal{C}$ is *weakly learnable* if:

> There is an algorithm $\mathcal{A}$, $\gamma > 0$, and $\delta_0 \in (0, 1)$ such that, for any $c \in \mathcal{C}$ and any distribution P over $\mathcal{X}$, if $\mathcal{A}$ is given n_0 examples drawn from P and labeled according to c, then it outputs a hypothesis $\hat{f}$ for which, with probability at least δ_0,
>
> $$\Pr_{X \sim P}\left(\hat{f}(X) \neq c(X)\right) \leq \frac{1}{2} - \gamma;$$
>
> here, although n_0 may depend on γ and δ_0, the latter are considered constant and so n_0 itself is considered a constant.

[1] We will implicitly assume efficiency while studying boosting.

Michael Kearns and Leslie Valiant developed the concept of weak learnability, and they posed the question of whether weak learnability is equivalent to strong learnability: that is, given a weak learning algorithm $\mathcal{A}$ which just predicts better than chance with non-negligible probability, can such an algorithm be *boosted* so that with arbitrarily high probability it obtains arbitrarily small risk (with the usual polynomial sample complexity requirements in $\frac{1}{\varepsilon}$ and $\frac{1}{\delta}$). A boosting scheme proceeds by repeatedly running the weak learner on different training samples and somehow aggregating all of the learned "weak hypotheses".

In 1989 — not long after the question was posed — Robert Schapire answered this question in the affirmative: he devised the first boosting scheme which uses a number of samples polynomial in $\frac{1}{\varepsilon}$ and $\frac{1}{\delta}$ and which is efficient whenever the underlying weak learning algorithm is efficient. In 1990, Yoav Freund developed a drastically different and (more efficient) algorithm, boost-by-majority, which is of interest in its own right. Finally, in 1995, Freund and Schapire developed AdaBoost. Whereas the previous two algorithms could only be run with knowledge of the weak learner's edge γ, AdaBoost is *adaptive*: it can be run without knowledge of γ and actually adapts to the edge obtained in each call to the weak learner (which often exceeds γ). Freund and Schapire's work on AdaBoost ultimately won them the Gödel Prize.

2 Boosting the confidence

It turns out that all of the real work in developing a boosting algorithm is in boosting the *accuracy* from $\frac{1}{2} - \gamma$ to $1 - \varepsilon$ for arbitrarily small ε. Boosting the *confidence* of a learner that already obtains risk at most ε with some constant probability is much easier. To see this, suppose that we have a learning algorithm which with positive constant probability δ_0 obtains risk at most ε. Then via repeated calls to the weak learner on fresh samples, it is simple to boost the confidence of this guarantee to $1 - \delta$ for arbitrarily small δ with an at most $\log \frac{1}{\delta}$ blowup in the sample complexity and runtime of the algorithm. Working out the details will be left as an exercise in the next problem set, but the main idea is this:

> Let's call a hypothesis ε-GOOD if its risk is at most ε. Suppose that a learning algorithm $\mathcal{A}$, when called on a training sample of size n, returns an ε-GOOD hypothesis with probability at least δ_0. If $\mathcal{A}$ is called k times on k independent training samples (each of size n'), then the probability that none of the resulting k hypotheses is ε-GOOD is at most $(1 - \delta_0)^k$.

Therefore, in order to prove that weak learnability implies strong learnability, we may without loss of generality assume that the weak learner already succeeds in obtaining risk at most ε_0 with probability at least $1 - \delta$ for arbitrarily small δ (at the price of the runtime being at most polynomial in $\log \frac{1}{\delta}$). What about boosting the accuracy? This is where AdaBoost comes in.

3 Boosting the accuracy: AdaBoost

AdaBoost proceeds in rounds, constructing one weak hypothesis h_t in each of T rounds. As a precondition we assume that we are given:

- a training sample $(x_1, y_1), \ldots, (x_n, y_n)$;
- a weak learning algorithm $\mathcal{A}$ which weak learns $\mathcal{C}$ with edge γ and success probability δ_0.

In round t of AdaBoost, the weak learner $\mathcal{A}$ is called on a reweighted version of the training sample with weights specified by a distribution D_t over $\{1, \ldots, n\}$. The weights are chosen carefully to give more emphasis to examples which were misclassified in the past. By the weak learning assumption, the weak learner will return some hypothesis h_t for which

$$\varepsilon_t := \Pr_{j \sim D_t}(h_t(x_j) \neq y_j) = \frac{1}{2} - \gamma_t$$

for some $\gamma_t \geq \gamma$ (with probability at least δ_0). The algorithm is shown below.

Algorithm 1. AdaBoost

```
Set $D_1(j) = \frac{1}{n}$ for $j = 1, \ldots, n$
For $t = 1, \ldots, T$:
    • Call $\mathcal{A}$ on distribution $D_t$, obtaining hypothesis $h_t$
    • Set $\alpha_t = \frac{1}{2} \log \frac{1-\varepsilon_t}{\varepsilon_t}$
    • For $j = 1, \ldots, n$:
          $D_{t+1}(j) = \frac{D_t(j) \cdot \exp(-\alpha_t y_j h_t(x_j))}{Z_t} = \frac{D_t(j)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } h_t = y_j \\ e^{\alpha_t} & \text{if } h_t \neq y_j, \end{cases}$
      where $Z_t$ is the normalization factor that renders $D_{t+1}$ a probability distribution
Output hypothesis:    $\hat{f} \colon x \mapsto \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$.
```

For now, we will just think of α_t as a learning rate; larger values of α_t lead to more aggressive updates to the distribution. In the analysis, we will see precisely why AdaBoost chooses the above setting for α_t.

One point begs for further clarification: how can a weak learner be run on a reweighted version of the training sample? For many learning algorithms, it is natural to incorporate weights on the training examples and to approximately minimize (up to error $\frac{1}{2} - \gamma$) the reweighted empirical risk; in fact, this often can be done with complete certainty (not merely with high probability). Still, for other weak learners that cannot incorporate weights, we can instead employ a technique called *boosting-by-resampling*.

The idea behind boosting-by-resampling, taking the case of $\delta_0 = 1/2$ purely for simplicity, is to sample n_0 examples from D_t, feed these examples to $\mathcal{A}$, and check whether or not the error guarantee holds for the resulting hypothesis (note that it is indeed possible to verify the error of the hypothesis!); it is guaranteed to hold with probability at least $1/2$ by the weak learning assumption. If it does not hold, simply try again with another sample of size n_0. Continuing in this fashion, the probability of k consecutive failures is at most 2^{-k}, and by choosing k large enough, we can wrap the probability of the unlucky event of k successive failures into the overall failure probability of our strong learning algorithm; it will suffice to set k to be logarithmic in $1/\delta$.

4 Weak learners

Since we will be proving that weak learnability implies strong learnability, it is worth grounding the idea of a weak learner via an example. Let's take $\mathcal{X}$ to be a feature space for animals, with each feature being a categorical variable. For instance, the first feature might be `biped` or `quadruped`, the second might indicate the color (from some finite set), the third might be `small` or `large`, and so on. A *decision tree* is a classifier for which each internal node is a decision node that inspects a single feature, and depending on the value, an example is sent to one of finitely many child nodes. This proceeds recursively until reaching a leaf node associated with a label, and this label is then predicted. *(I drew an example in class)* The C4.5 algorithm is a highly successful method for learning decision trees that has enjoyed widespread use. However, in terms of the theory, there is still a large gap in our understanding of how to learn decision trees well.

A *decision stump* is a special case of a decision tree of height 1; a decision stump thus has only one decision node. Given training data, it is trivial to efficiently identify a decision stump that minimizes the training error: just try them all and pick the best one. Moreover, it is conceivable that by inspecting one feature one may be able to obtain an edge γ over the baseline accuracy of $\frac{1}{2}$. Because of our ability to learn them efficiently and the plausibility that they can serve as weak learners, decision stumps will form our running example of a weak learner.

Even though decision stumps are incredibly simple models, it turns out that boosted decision stumps often outperform more sophisticated models like decision trees learned by C4.5.

(I showed some figures demonstrating this in class)

5 Empirical risk / consistency analysis for AdaBoost

Recall that our goal is to prove that weak learnability implies strong learnability. A first natural step is to ensure that we have obtained low empirical risk on the sample. This guarantee is captured by the following theorem:

Theorem 1. *If AdaBoost is run for T rounds on a training sample of size n, then*

$$\frac{1}{n}\sum_{j=1}^{n} \mathbf{1}\left[\hat{f}(X_j) \neq Y_j\right] \leq \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right).$$

Recall that

$$\varepsilon_t = \Pr_{j \sim D_t}\left(h_t(X_j) \neq Y_j\right) = \frac{1}{2} - \gamma_t.$$

A proof of Theorem 1 *(which we did in class)* can be found in Schapire's lecture notes[2].

Now, suppose that each $\gamma_t \geq \gamma$; this can be arranged with high probability as explained below. Then consistency must hold as soon as $e^{-2T\gamma^2} < \frac{1}{n}$, i.e. as soon as $T \geq \frac{\log n}{2\gamma^2}$.

[2]`http://www.cs.princeton.edu/courses/archive/spr08/cos511/scribe_notes/0303.pdf`.

Now, let's see how to ensure, with high probability over the internal randomization of the algorithm, that each $\gamma_t \geq \gamma$. Assume that $\mathcal{A}$ is a weak learner for $\mathcal{C}$ with edge $\gamma > 0$ and success probability $\delta_0 > 0$, and consider running AdaBoost with boosting-by-resampling. That is, in obtaining the weak hypothesis h_t for each $t \in [T]$, we repeat the following procedure enough times until either Step 3 succeeds *or* until we have repeated the procedure enough times where the probability of every Step 3 failing is as small as desired:

Step 1. Resample from D_t: obtain a sample of size n_0 from D_t.

Step 2. Call the weak learner on this sample, obtaining candidate hypothesis h_t.

Step 3. Verify that $\gamma_t \geq \gamma$.

Taking the union bound over $[T]$, we can also ensure that with high probability every $\gamma_t \geq \gamma$. Thus, by allowing for $\Omega\left(\log \frac{T}{\delta}\right)$ rounds of resampling for each t, we can ensure with high probability at least $1-\delta$ that each $\gamma_t \geq \gamma$, which, by Theorem 1 with $T \geq \frac{\log n}{2\gamma^2}$ implies (with the same probability) that $\hat{f}$ is consistent with c on the training sample.