CIS 320: Algorithms	November 10, 2021

Lecture 19

Lecturer: Aaron Roth *Scribe: Aaron Roth*

Boosting

In this class we'll derive a powerful family of machine learning algorithms, taking advantage of the polynomial weights algorithm and the minimax theorem we proved last class. First some basic definitions:

Definition 1 *A labeled* datapoint *is a pair $(x, y) \in X \times Y$, where X is some space of* features *and Y is some space of* labels*: for example, a common case is $X = \mathbb{R}^d$, and $Y = \{0, 1\}$.*

A dataset $D \in (X \times Y)^n$ is a collection of n labeled datapoints.

Our goal will be to find some function $f : X \to Y$ for predicting labels from their features that has high accuracy:

Definition 2 *Given a predictor $f : X \to Y$, its prediction accuracy on a dataset D is:*

$$acc(f, D) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[f(x_i) = y_i]$$

The prediction accuracy as defined uniformly weights all of the points in the dataset. But we can also define weighted *prediction accuracy relative to any other weighting $w \in \Delta[n]$ of the n points:*

$$acc(f, D, w) = \sum_{i=1}^{n} w_i \mathbb{1}[f(x_i) = y_i]$$

Note that $acc(f, D)$ is simply the special case of $acc(f, D)$ in which $w_i = 1/n$ for all i.

Remark We're ignoring an important statistical aspect of machine learning here: our goal is typically not actually to predict the labels of the points in our dataset D, but to predict the labels of new points drawn from the same distribution as D that we have never seen before. Doing so requires making high accuracy predictions using "simple" hypotheses. We'll focus on the algorithmic aspect here, but the "boosting" approach we discuss here *also* has good statistical generalization properties.

Definition 3 *A hypothesis class H is a collection of predictors or* hypotheses *$h : X \to Y$. A weighted learning algorithm A with range H is a mapping from datasets and weight vectors to hypotheses in H. $A : (X \times Y)^n \times [0, 1]^n \to H$.*

If (e.g.) $Y = \{0, 1\}$ then it is generally uninteresting to find a hypothesis h that has $acc(h, D) \leq 1/2$, since we could obtain that just by randomly guessing or always predicting the most common label. But what about if we could come up with a hypothesis that has just *slightly* better accuracy: $acc(h, D) = 0.51$. Would that be interesting? This is often not hard: it might involve simply finding a single feature that has a small correlation with the label. We'll define a weak learning algorithm for a dataset D as one that can always come up with a hypothesis with weighted accuracy better than random guessing, for any weight vector:

Definition 4 *A weighted learning algorithm A is a weak learning algorithm for D if for every distribution $w \in \Delta[n]$, $A(D, w) = h$ such that:*

$$acc(h, D, w) \geq 0.51$$

Remark If the notion of a "weighted" learning algorithm seems odd, observe that one way to implement a weighted learning algorithm is to construct a new dataset D' by sampling from D under the probability distribution specified by w, and then running a regular old (unweighted) learning algorithm on D'. We'll use weighted learning algorithms just because it avoids the need to argue about subsampling.

A weak learning algorithm seems — well, weak. It only guarantees to beat random guessing by a tiny amount. But remarkably, we will show that if there is a computationally efficient weak learning algorithm for D, then there is also a computationally efficient strong learning algorithm for D — one that can find a perfect predictor.

Definition 5 *A is a strong learning algorithm for D if $A(D) = h$ such that $acc(h, D) = 1$.*

Theorem 6 *For any dataset D, if there exists an efficient (polynomial time) weak learning algorithm A for D, then there exists an efficient strong learning algorithm A' for D.*

Proof Our construction will involve applying the minimax theorem to analyze an appropriately defined zero sum game, and then computing an approximate equilibrium strategy in the zero sum game.

Let H be the hypothesis class used by the weak learning algorithm A. We define a zero sum game as follows:

1. The action space for the minimization player (the "Data Player") is the set of datapoints in the dataset: $A_1 = D$.
2. The action space for the maximization player (the "Learner") is $A_2 = H$.
3. The cost function is C is defined as $C((x_i, y_i), h) = \mathbb{1}[h(x_i) = y_i]$.

How well can the players do in this game? The assumption that there exists a weak learning algorithm for D immediately lets us compute the min max value for the game — i.e. how well the learner could do if she got to best respond to a fixed strategy of the data player:

$$\begin{aligned}
\min\max(C) &= \min_{w \in \Delta[n]} \max_{h \in H} \sum_{i=1}^{n} w_i C((x_i, y_i), h) \\
&= \min_{w \in \Delta[n]} \max_{h \in H} w_i \mathbb{1}[h(x_i) = y_i] \\
&= \min_{w \in \Delta[n]} \max_{h \in H} acc(h, D, w) \\
&\geq 0.51
\end{aligned}$$

where the final inequality follows from the assumption that we have a weak learner that uses hypotheses in H.

We can now apply the minimax theorem to conclude that the Learner can do just as well, even if she is forced to commit to her strategy first:

$$\min\max(C) = \max\min(C) = \max_{p \in \Delta H} \min_{i \in [n]} \sum_{h \in H} p_h \mathbb{1}[h(x_i) = y_i] \geq 0.51$$

In other words, there is some *fixed* distribution p^* over hypotheses $h \in H$ such that for *every* data point $(x_i, y_i) \in D$, at least 51% of the probability mass under p is on hypotheses that correctly label (x_i, y_i). How can we use this? Consider the following "majority vote" classification rule f_{p^*}:

$$f_{p^*}(x) = \mathbb{1}\left[\sum_{h:h(x)=1} p^*_h \geq 0.5\right]$$

f_{p^*} turns out to have perfect accuracy.

Lemma 7 *For the distribution* $p^* = \max_{p \in \Delta H} \min_{i \in [n]} \sum_{h \in H} p_h \mathbb{1}[h(x_i) = y_i]$*, the hypothesis* f_{p^*} *satisfies* $acc(f_{p^*}, D) = 1$

Proof We need to show that for *every* $(x_i, y_i) \in D$, $f_{p^*}(x_i) = y_i$. From our minimax calculation, we know that for every i, $\sum_{h \in H} p^*_h \mathbb{1}[h(x_i) = y_i] \geq 0.51$. Hence if $y_i = 1$, we have that

$$\sum_{h \in H} p^*_h \mathbb{1}[h(x_i) = 1] = \sum_{h : h(x) = 1} p^*_h \geq 0.51$$

and hence by definition $f_{p^*}(x_i) = 1$. Similarly, if $y_i = 0$, we know that $\sum_{h:h(x)=1} p^*_h < 0.49$ and hence by definition $f_{p^*}(x_i) = 0$, which completes the proof. ■

So we know there exists a hypothesis ($f_{p^*}(x_i)$) with zero training error — to complete the proof of our theorem, we only need to give an efficient algorithm for finding it. Note that in our proof of Lemma 7, the only property we used about p^* is that it was a distribution p which satisfied $\min_{i \in [n]} \sum_{h \in H} p_h \mathbb{1}[h(x_i) = y_i] > 0.5$. p^* satisfied this with some slack (0.01). But this means we would have the same result if we could compute $\hat{p}$, an ϵ-approximate max min strategy for our zero-sum game C, for $\epsilon < 0.01$. Fortunately, we derived an algorithm for computing approximate equilibria in zero sum games last class! Lets recall it here in our context. Remember that we need one player (here we will choose the minimization/data player) to maintain a distribution w^t over their actions using the polynomial weights update algorithm. The maximization player (the learner in our case) needs to be able to compute a cost maximizing action — an action that obtains value at least v in the game against the minimization players current strategy w^t, where v is the max min value of the game that we are aiming to achieve. For us, $v = 0.51$, and the corresponding computational task is exactly the weak learning problem. We assume we have access to a weak learner A, and so we can use it in our algorithm as a subroutine.

Algorithm 1 Boost(D, A)

Let $T \leftarrow \frac{4 \log n}{\epsilon^2}$ for $\epsilon < 0.01$.
Initialize a copy of polynomial weights to run over $w^t \in \Delta^n$.
for $t = 1$ to T **do**
  Let $h^t = A(D, w^t)$
  Let $\ell^t \in [0,1]^m$ be such that $\ell^t_i = \mathbb{1}[h^t(x_i) = y_i]$.
  Pass ℓ^t to the PW algorithm.
end for
Let $\hat{p} = \frac{1}{T} \sum_{t=1}^{T} e_{h^t}$. (Note that this is concisely representable even though H is large, because $\hat{p}$ has support over only the T models h^t.)
Return $f_{\hat{p}}(x)$.

It remains to examine the running time of our Boosting algorithm. Since ϵ is a constant, on a dataset of size n, it runs for only $O(\log n)$ many iterations. At each iteration it makes a single call to our weak learning algorithm A, which we assume is polynomial time. It then has to update the polynomial weights distribution over the n datapoints, which takes time $O(n)$. Thus the total running time is $O(\log n(n + R(A))$, where $R(A)$ is the running time of our weak learning algorithm. Thus if our weak learning algorithm runs in polynomial time, so does our strong learning algorithm. ■

A couple of remarks are in order:

1. First, as noted earlier, we have focused here on the problem of finding a hypothesis that correctly labels the training set, and have ignored the statistical issue of "generalization" — how well the learned hypothesis fits new data. This is a field of study in its own right, but by and large, for "simple" hypothesis classes H, fitting the training data implies coming close to fitting new data

drawn from the same distribution. Since H only needs to contain a weak learner, H can often taken to be something very simple, like depth 2 decision trees.

2. The hypothesis $f_{\hat{p}}$ that we output is not in H itself — but it is not too much more complex, in a way that can be formulated: namely, it is a threshold function that operates on a convex combination of at most $O(\log n)$ models from H. Hence, if H is a "simple" class, then $f_{\hat{p}}$ is also "simple" in a way that can be formalized to bound generalization error.

3. The assumption that there exists a weak learning algorithm is (as we have just shown!) much stronger than it first appears — in particular, it implies that the data can be perfectly classified. But recall that our analysis of our equilibrium computation algorithm last class made no assumptions on the action set of the maximization/best response player. Hence the analysis goes through even if for the $O(\log n)$ iterations of the algorithm, our learning algorithm A happens to be able to find models h^t that perform better than random guessing — even if it is not guaranteed to be able to do so for all possible distributions. Hence Boosting is a sensible – and popular — approach to learning in practice, even when the strong assumption of the existence of a weak learning algorithm does not technically hold.