Lecture 9: Boosting

Akshay Krishnamurthy
akshay@cs.umass.edu

October 3, 2017

1 Recap

Last week we discussed some algorithmic aspects of machine learning. We saw one very powerful family of learning algorithms, namely nonparametric methods that make very weak assumptions on that data-generating distribution, but consequently have poor generalization error/convergence rates. These methods tend to have low approximation errors, but extremely high estimation errors. Then we saw some principled ways to manage the approximation/estimation tradeoff. Today we'll see another powerful class of learning algorithm that tends to have small approximation errors in practice.

2 Weak Learning

Boosting refers to the technique of aggregating many predictors together to form one (hopefully better) predictor. There are many boosting algorithms for many different settings and the theory here is quite rich. Boosting arose to positively answer a question about weak versus strong learning from the computational learning theory literature:

Definition 1 (Weak PAC learning)**.** *A hypothesis class $\mathcal{H}$ is weakly-PAC learnable with edge γ if there exists a function $n_{\mathcal{H}}^{weak} : (0,1) \to \mathbb{N}$ and an algorithm such that for every $\delta \in (0,1)$ and for every realizable data distribution, if the algorithm is run on $n \geq n_{\mathcal{H}}^{weak}(\delta)$ samples from the distribution, then with probability at least $1-\delta$ it produces a predictor $\hat{h}$ with $R(\hat{h}) \leq 1/2 - \gamma$.*

At face value, this is much weaker than the previous definition of (realizable) PAC learning that we saw before. In particular, with the association $\epsilon = 1/2 - \gamma$ it could very well be the case that $n_{\mathcal{H}}(\epsilon, \delta) = n_{\mathcal{H}}^{\text{weak}}(\delta)^{\frac{1}{1/2-\epsilon}}$ which would not be a polynomial sample complexity bound for (strong) PAC learning. On the other hand, we saw that for classification, a hypothesis class is PAC learnable if and only if the VC dimension is finite. This applies equally for weak learning again using the association $\epsilon = 1/2 - \gamma$ in the sample complexity lower bound.

The main advantage of weak learning is *computational*. When we are learning a class $\mathcal{H}$ for which we might not know how to find the ERM efficiently, the idea is to instead use a different class $\mathcal{B}$ for which (a) any ERM in $\mathcal{B}$ satisfies the weak learning property for the original problem, and (b) we can find an ERM for $\mathcal{B}$ in polynomial time. Thinking about it this way, the algorithm just may not be able to find a hypothesis with error rate ϵ, but it can always find a hypothesis with error rate 0.45 which is better than random.

Example 1 (Decision stumps)**.** *Decision stumps are similar to the decision lists that we saw on the homework. The decision stump class is:*

$$\mathcal{B}_{DS} = \{x \to sign(\theta - x_i) \times b \mid \theta \in \mathbb{R}, i \in [d], b \in \{-1, +1\}\}.$$

These are functions that pick an axis, and split the data on that axis and assign positive label to one side and negative label to the oither side. These are essentially axis aligned thresholds.

It is easy to find the ERM for decision stumps in polynomial time. For each coordinate, sort the data according to that coordinate value, and compute the risk of the threshold between each pair of points. Then take the minimal.

A natural question is: suppose a class $\mathcal{H}$ can be γ-weak learned in a computational efficient manner. Then is there a way to strongly PAC-learn this class in a computationally efficient manner? Boosting provides a positive resolution to this question.

Theorem 2. *Let $\mathcal{H}$ be any concept class and let $\mathcal{B}$ be another hypothesis class. Then if $\mathcal{H}$ is efficently weakly PAC learnable using $\mathcal{B}$, $\mathcal{H}$ is also efficiently strongly PAC learnable using a hypothesis class that is derived from $\mathcal{B}$. This hypothesis class consists of ternery majority trees with leaves from $\mathcal{B}$.*

We will not prove this theorem (it is quite complicated). However at a high level the idea is to combine the weak learners, trained on different data distributions, in a clever way to boost the accuracy of the predictions. This idea has been turned into a much more practical algorithm called AdaBoost, which itself has many interesting theoretical properties.

3 Boosting

Let's assume for now that we have a weak learning algorithm with edge γ. We'd like to use this algorithm iteratively to compute a globally good predictor. When thinking about this, a natural idea is to first train a weak predictor on the empirical distribution over the samples, then see where the weak predictor is making mistakes and train a predictor focuses on just this part. Intuitively if we aggregate these predictors together, we should do well on the entire domain. This leads to the AdaBoost algorithm which formalizes this intuition.

We are given n samples $\{(x_i, y_i)\}_{i=1}^n$ where $y_i \in \{-1, +1\}$ and $x_i \in \mathcal{X}$, some abstract instance space. We also have a weak learning algorithm Alg. The algorithm will operate on distributions over or re-weightings of the training set, specified by D_t where $D_t \in \mathbb{R}_+^n$ with $\sum_{i=1}^n D_t(i) = 1$. With this re-weighting, the empirical error of hypothesis h_t is

$$\epsilon_t = R_{D_t}(h_t) = \sum_{i=1}^n D_t(i)\mathbf{1}\{h_t(x_i) \neq y_i\}. \tag{1}$$

The algorithm operates iteratively. We start with $D_1(i) = 1/n$ for all i. For each round $t = 1, \ldots, T$

1. Let h_t be the weak learner trained on D_t. Let ϵ_t be as in Eq. (1).
2. Set $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
3. Update

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} \text{ if } h_t(x_i) = y_i \\ e^{-\alpha_t} \text{ if } h_t(x_i) \neq y_i \end{cases} = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

   where Z_t normalizes the distribution.

After T iterations we output the weighted majority hypothesis $H(x) = \text{sign}(\sum_{t=1}^T \alpha_t h_t(x))$.

Let's try to understand the algorithm intuitively first. If the weak learning assumption holds, then $\epsilon_t < 1/2$ and this means that $\alpha_t > 0$. With this in mind, the re-weighting scheme clearly downweights points that h_t labels correctly and upweights points that h_t gets wrong. Based on this, if the weak classifier at round $t+1$ is to also satisfy the weak learning assumption, then it must focus its attention on the examples that h_t got wrong, which will help us when we do the aggregation. In particular, it is not too hard to check that the error rate of h_t on the distribution D_{t+1} is exactly $1/2$, and this means that the weak learner (if it is to maintain its edge), must choose a different hypothesis on the next round.

There are many theoretical properties of boosting that are worthy of exploration. Indeed there is a whole textbook dedicated to it! Today we'll discuss just a few of them.

Theorem 3 (Training error)**.** *Let $\gamma_t = 1/2 - \epsilon_t$ be the edge at round t. Then*

$$\frac{1}{n}\sum_{i=1}^n \mathbf{1}\{H(x_i) \neq y_i\} \leq \prod_{t=1}^T \sqrt{1 - 4\gamma_T^2} \leq \exp\left(-2\sum_{t=1}^T \gamma_t^2\right).$$

Remark 4. *Under a slightly weaker notion than the weak learning assumption (we just need it on the empirical distribution), we are guaranteed that $\gamma_t \leq \gamma$ for all rounds t. Since if the training error is strictly less than $1/n$, it must be exactly zero, this means that after $\frac{\log(n)}{2\gamma^2}$ rounds, AdaBoost finds a predictor with zero training error.*

Remark 5. *Of course this says nothing about the generalization performance of AdaBoost, which is what we are ultimately interested in. We will cover this shortly.*

Proof. The intuition for the proof is that any example that you are predicting incorrectly after T rounds has an exponentially large weight in the distribution. But since it is a distribution, it must sum to one, so you cannot be wrong on many points.

Define $F(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$ and let us unravel the recurrence for the definitions of D_T

$$\begin{aligned} D_{T+1}(i) &= D_1(i) \times \frac{\exp(-\alpha_1 y_i h_1(x_i))}{Z_1} \times \frac{\exp(-\alpha_T y_i h_T(x_i))}{Z_T} \\ &= \frac{D_1(i)\exp(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i))}{\prod_{t=1}^{T} Z_t} = \frac{D_1(i)\exp(-y_i F(x_i))}{\prod_{t=1}^{T} Z_i}. \end{aligned}$$

The next step is to observe that $\exp(-yF(x))$ is an upper bound on the 0/1 loss. This follows since $H(x) = \text{sign}(F(x))$ and if $H(x) \neq y$ then $yF(x) \leq 0$ so that $\exp(-yF(x)) \geq 1 = \mathbf{1}\{H(x) \neq y\}$. Thus we can express the empirical error as

$$\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{H(x_i) \neq y_i\} \leq \sum_{i=1}^{n} D_1(i)\exp(-y_i F(x_i)) = \sum_{i=1}^{n} D_{T+1}(i) \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} Z_t$$

The last equality follows since D_{T+1} is a distribution. The last step in the proof is to show that $Z_t = \sqrt{1-4\gamma_t^2}$, which requires analyzing the update scheme.

$$\begin{aligned} Z_t &= \sum_{i=1}^{n} D_t(i)\exp(-\alpha_t y_i h_t(x_i)) = \sum_{i:y_i = h_t(x_i)} D_t(i)\exp(-\alpha_t) + \sum_{i:y_i \neq h_t(x_i)} D_t(i)\exp(\alpha_t) \\ &= \exp(-\alpha_t)(1-\epsilon_t) + \exp(\alpha_t)\epsilon_t \\ &= \exp(-\alpha_t)(1/2+\gamma_t) + \exp(\alpha_t)(1/2+\gamma_t) = \sqrt{1-4\gamma_t^2}. \end{aligned}$$

In the last step we use the definition of $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t}) = \frac{1}{2}\log(\frac{1/2+\gamma_t}{1/2-\gamma_t})$. Finally, we use the approximation $1 + x \leq e^x$. □

There are two natural questions you might be asking at this point. First, how do we ensure that the weak-learning assumption is satisifed at every iteration so that we can drive the training error to zero? Second, what about generalization? On our way to thinking about generalization, let us first take a detour to describe another perspective about boosting.

4 Boosting and Loss Minimization

Another deep connection for boosting is that it can be seen as an algorithm for computing the empirical risk minimizer with a particular hypothesis space and loss function. Revisiting the proof of the training error bound, we used the inequality

$$\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{y_i \neq H(x_i)\} \leq \frac{1}{n}\sum_{i=1}^{n} \exp(-y_i F(x_i)).$$

In fact, every other step in the proof was an equality! This term on the right hand side is the empirical exponential loss of the function F. Rather than derive AdaBoost as an algorithm, we can think of the ERM problem, where

the hypothesis class consists of distributions over base learners i.e. $\mathcal{F} = \{x \mapsto \sum_{j=1}^{N} \alpha_j h_j(x) \mid N \in \mathbb{N}, \alpha \in \mathbb{R}_+^N, h_1, \ldots, h_N \in \mathcal{H}\}$, and the loss function is the exponential loss.

$$\text{minimize}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \exp(-y_i f(x_i)).$$

We have seen many problems of this form before, although not using the exponential loss and not with this strange function class. But AdaBoost can be viewed as a *coordinate descent* algorithm for solving this optimization problem. Specifically, consider the following procedure, which initializes $F_0 = 0$ and repeats for T iterations:

1. Choose $h_t \in \mathcal{H}, \alpha_t \in \mathbb{R}$ to minimize

$$\frac{1}{n} \sum_{i=1}^{n} \exp(-y_i(F_{t-1}(x_i) + \alpha_t h_t(x_i))).$$

2. Update $F_t \leftarrow F_{t-1} + \alpha_t h_t$.

If you think about $|\mathcal{H}| = N$, this procedure is just coordinate descent on the α parameters, which are in $\mathbb{R}_+^N$. At each iteration you find the "coordinate" (hypothesis) that helps minimize the loss function as much as possible, and then you take a step in the α space by adding it in to the current predictor. In fact this is exactly what AdaBoost is also doing.

Theorem 6. *The coordinate descent algorithm is equivalent to AdaBoost if the weak learner always minimizes the training error on the weighted distribution.*

Proof. We already saw in AdaBoost's training error bound that

$$\frac{1}{n} \sum_{i=1}^{n} \exp(-y_i F_{t-1}(x_i)) = \sum_{i=1}^{n} D_t(i) \prod_{\tau=1}^{t-1} Z_\tau$$

And this means that

$$\frac{1}{n} \sum_{i=1}^{n} \exp(-y_i(F_t(x_i)) = \sum_{i=1}^{n} D_t(i) \left(\prod_{\tau=1}^{t-1} Z_\tau \right) \exp(-y_i \alpha_t h_t(x_i)) \propto \sum_{i=1}^{n} D_t(i) \exp(-y_i \alpha_t h_t(x_i)) = Z_t$$

Thus the optimization in the coordinate descent procedure is equivalent to minimizing the normalization factor in AdaBoost. We now show that the choices h_t, α_t from AdaBoost in fact minimize these. First for fixed h with error ϵ, the choice $\alpha = \frac{1}{2} \log(\frac{1-\epsilon}{\epsilon})$ minimizes the normalization factor, since it amounts to

$$Z_t = e^{-\alpha}(1 - \epsilon) + e^{\alpha}\epsilon.$$

By taking derivative, we can verify that the above choice of α minimizes the expression, and moreover the minimum is $2\sqrt{\epsilon(1-\epsilon)}$, which is monotonically decreasing in ϵ. Thus we should choose the h_t with minimum error on D_t. □

This connects AdaBoost with convex optimization and loss minimization, as AdaBoost can be seen as computing an approximate ERM. This can also help us obtain generalization bounds by thinking of AdaBoost as an ERM algorithm over the function space $\mathcal{F}$.

5 Boosting and Generalization

Thinking about AdaBoost as optimizing the exponential loss over $\mathcal{F}$ gives us an immediate VC-based bound.

Theorem 7 (VC-based generalization bound)**.** *Suppose AdaBoost is run for T iterations on n random examples using base classifiers from $\mathcal{H}$ with VCdim($\mathcal{H}$) = d. Assume $n \geq \max\{d, T\}$. With probability at least $1 - \delta$*

$$R(H) \leq \hat{R}(H) + \sqrt{\frac{32\left(T \log(en/T) + dT \log(en/d) + \log(8/\delta)\right)}{n}}.$$

Moreover, if H has $\hat{R}(H) = 0$ then

$$R(H) \leq \frac{2\left(T\log(2en/T) + d\log(2en/d) + 2\log(2/\delta)\right)}{n}.$$

The terms here should be somewhat familiar from the VC-theorem proof. Basically what we're going to show is that if you run AdaBoost for T iterations, then you are searching for a hypothesis of the form $\text{sign}(\sum_{t=1}^{T} \alpha_t h_t(\cdot))$. The VC-dimension of this linear combination class scales linearly with number of coefficients T. This more or less proves the theorem.

But if you remember the Rademacher complexity problem on homework 2, this should be unsatisfying since we are taking a convex combination of base learners but somehow paying for it, while there we did not pay for the number of hidden units. We'll see next time how to get a much better generalization bound that is independent of the number of iterations.

Proof. Let S be a training sample of size n and let $\mathcal{H}_S$ be the restriction of $\mathcal{H}$ to S as we have seen before. We know by the Sauer-Shelah lemma that $|\mathcal{H}_C| \leq (en/d)^d$ since $\text{VCdim}(\mathcal{H}) = d$ (and using the fact that $n \geq d$ for the approximation to kick in).

Let us choose a fixed sequence of T of these restrictions, which we call $h_1, \ldots, h_T$. From these, create a modified sample S' where $x_i' = (h_1(x_i), \ldots, h_T(x_i))$ is a T-dimensional binary vector. Once we commit to these T hypotheses, the aggregation step is equivalent to applying a linear separator to this augmented dataset. Since we know that linear separators in T dimensions have VC-dimension T, we get that number of labelings of the augmented dataset is at most $(en/T)^T$. Thus the total number of labelings is at most

$$\tau(n) \leq \left(\frac{en}{T}\right)^T \left(\frac{en}{d}\right)^{Td}$$

Plugging this in for the growth function in the usual VC theorem proof gives the result. □

Proposition 8. *In T dimensions, the set of linear thresholds, $h(x) = \text{sign}(\langle w, x\rangle)$, has VC-dimension T.*

Note that these are axis-aligned thresholds, which is why we are not getting $T + 1$ as we predicted earlier.

Proof. That there exists T points that can be shattered is obvious if we use the standard basis vectors as the points. The other part is less obvious. If we have $T + 1$ points, then they must be linearly dependent, so in particular we can write $x_{T+1} = \sum_{i=1}^{T} \beta_i x_i$. If this set can be shattered, then we can find a predictor w that agrees with β_i for each x_i and predicts $h(x_{T+1}) = -1$. This appears to be a contradiction

$$0 = \langle w, \sum_{t=1}^{T} \beta_t x_t - x_{T+1}\rangle = \sum_{t=1}^{T} \beta_t \langle w, x_t\rangle - \langle w, x_{T+1}\rangle > 0.$$ □