CS269: Machine Learning Theory
Lecture 13: Boosting

November 8, 2010

Lecturer: Jennifer Wortman Vaughan
Scribe: Kun Huang, Maria Pavlovskaia, Matt Wang

1 Introduction to Boosting and the AdaBoost Algorithm

1.1 Background and Motivation

The idea behind boosting is to construct an accurate hypothesis from a set of hypotheses that are each guaranteed to perform slightly better than random guessing on particular distributions of data. This idea originally arose from attempts to prove the robustness of the PAC model. By *robustness*, we mean the notion that slight alterations to the model definitions should not result in dramatically different results, such as the ability to learn different classes of functions.

The question of whether weak learning implies strong learning was first posed in the late 80s by Kearns and Valiant. The first boosting algorithm was developed by Rob Schapire in order to answer this question. This paved the way for the immensely popular AdaBoost algorithm, which was developed by Freund and Schapire a couple of years later.

To formalize this discussion we will first review the basic definition of PAC learnability, which we now refer to as *strong* PAC learnability to distinguish it from the weak version. Note that this is the standard definition that we have discussed in class previously.

Definition 1. *Strong PAC Learnability: $\mathcal{C}$ is PAC learnable by $\mathcal{H}$ if there exists an algorithm A such that for any concept $c \in \mathcal{C}$, for any distribution D, for any $\epsilon, \delta \in (0, 1)$, given a polynomial number of examples i.i.d from D and labeled by c, A outputs an $h \in \mathcal{H}$ such that with probability $\geq 1 - \delta$, $err(h) \leq \epsilon$.*

Now we will define the notion of weak PAC learnability. This definition essentially says that we can weakly learn a concept class $\mathcal{C}$ if we can produce a hypothesis that does some amount better than random guessing.

Definition 2. *Weak PAC Learnability: $\mathcal{C}$ is weakly learnable by $\mathcal{H}$ if $\exists$ algorithm A, $\exists \gamma > 0$, such that $\forall c \in \mathcal{C}, \forall D, \forall \delta > 0$, given a polynomial number of examples, A outputs an $h \in \mathcal{H}$ such that with probability $\geq 1 - \delta$, $err(h) \leq \frac{1}{2} - \gamma$.*

It should be clear that strong learnability implies weak learnability. The question we want to answer is whether weak learnability implies strong learnability. We consider two versions of this question:

- Does weak learnability imply strong learnability for a fixed distribution D?
- Does weak learnability imply strong learnability in the general PAC sense?

The answer to the first case is no, which can be illustrated through the following example:

Suppose the input space is $\{0,1\}^n$ (all binary strings of length n). Define the distribution D so that $P_D(<0,0,\cdots,0>) = \frac{1}{2}$, and D is uniform everywhere else. $\mathcal{C}$ is all functions $\{0,1\}^n \rightarrow \{0,1\}$. Given a set of examples, the algorithm is likely to learn the label of the string $<0,0,\cdots,0>$ since it is very likely that it will appear amongst the examples. However, most other new points will pose new information that we know nothing about, since we are only given a polynomial number of points but there is an exponential (2^n) number of points in total.

The answer to the second case is yes, as we will start to see in the following sections. Let's first define this question more formally:

If $\mathcal{C}$ is weakly learnable by $\mathcal{H}$, does there exist $\mathcal{H}'$ such that $\mathcal{C}$ is (strongly) learnable by $\mathcal{H}'$?

That is, suppose we are given examples drawn i.i.d from D and labeled by $c \in \mathcal{C}$, and a weak learning algorithm A for $\mathcal{C}$. Can we produce a new function h such that with high probability, $err(h) \leq \epsilon$, for any fixed $\epsilon > 0$?

1.2 AdaBoost

The algorithm:

The input to the AdaBoost algorithm is a weak learning algorithm A and a set of points $(x_1, y_1), ..., (x_m, y_m)$ where $y_i \in \{-1, 1\}$. Note that this set of points is fixed. Therefore in order to draw new information from A, we modify the distribution of these points and run A on a polynomial sized sample chosen over this distribution. We do this for T rounds, with the idea that each round finds a new weak function to classify the points in the current sample that previous rounds were unable to classify well. Note that α_t below is a parameter of the algorithm, and we will define this parameter optimally in our analysis. Formally the algorithm does the following:

- Initialize $D_1(i) = \frac{1}{m}\ \forall i$ (uniform distribution)
- For $t = 1, ..., T$
  - Draw a polynomial sized sample from D_t
  - Run A on this sample to produce h_t
  - Update the distribution:
$$D_{t+1}(i) = \frac{D_t(i)\text{exp}\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t}$$

This algorithm produces as output a function h where

$$h(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

It can be noted that, when the algorithm gets the label correctly, $y_i h_t(x_i) = 1$ and $y_i h_t(x_i) = -1$, when labeled wrong. Z_t is the normalization factor and is equal to:

$$\sum_{i=1}^{m} D_t(i)\text{exp}\left(-\alpha_t y_i h_t(x_i)\right)$$

AdaBoost is popular for a number of reasons:

- It has no tricky parameters to set.
- It does not require prior knowledge of γ (which is discussed below).
- It is computationally simple.
- Its training error goes to zero very quickly.
- It generalizes very well and avoids over-fitting.
- It satisfies nice theoretical guarantees.

2 Analysis of AdaBoost

2.1 Goal

The focus of today's lecture is the traditional analysis of training error. In particular, we will show that the training error of the algorithm goes to zero exponentially in the number of time steps.

2.2 Some Definitions

Definition 3. *Define ϵ_t to be the error of h_t on the distribution D_t that was used to learn h_t*

$$\epsilon_t := \Pr_{X \sim D_t}(h_t(x) \neq c(x)).$$

Definition 4. *Let $\gamma_t := 1/2 - \epsilon_t$. This is a measure of how much better than random guessing we're doing using this weak classifier at time t.*

2.3 The Main Result

Theorem 1. *Let h be the function output by AdaBoost after T rounds. Then*

$$\widehat{err}(h) \leq e^{-2\sum_{t=1}^{T} \gamma_t^2}.$$

In particular, if $\gamma_t \geq \gamma \ \forall t$,

$$\widehat{err}(h) \leq e^{-2\gamma^2 T}.$$

That is, the training error decreases exponentially as the number of rounds T grows. Since the training error goes to zero really quickly, the algorithm attains 0 training error in very few time-steps. (Note that the training error is always a multiple of $1/m$, so if the bound gets lower than this, we are guaranteed to have $\widehat{err}(h) = 0$.)

Notice that while the error bound depends on γ, the algorithm has no γ dependence. Hence we get the error bound guarantee without having to know γ to run the algorithm.

2.4 The Proof

The proof will proceed in three steps:

1. Unravel the recurrence on D_t to show that
$$D_{T+1}(i) = \frac{e^{-y_i \sum_{t=1}^T \alpha_t h_t(x_i)}}{m \prod_{t=1}^T Z_t}$$
2. Use Step 1 to show that
$$\widehat{\text{err}}(h) \leq \prod_{t=1}^T Z_t$$
3. Show that $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$ for all t, and plug this into Step 2 to achieve the bound.

Step 1

We unravel the recurrence for D_t to get

$$\begin{aligned}
D_{T+1}(i) &= \frac{D_T(i) e^{-\alpha_T y_i h_T(x_i)}}{Z_T} \\
&= \frac{D_{T-1}(i) e^{-\alpha_{T-1} y_i h_{T-1}(x_i)} e^{-\alpha_T y_i h_T(x_i)}}{Z_T Z_{T-1}} \\
&\quad \text{(continue recurrence substitutions until we reach } D_1(i)\text{)} \\
&= D_1(i) \frac{\prod_{t=1}^T e^{-\alpha_t y_i h_t(x_i)}}{\prod_{t=1}^T Z_t} \\
&= \frac{\prod_{t=1}^T e^{-\alpha_t y_i h_t(x_i)}}{m \prod_{t=1}^T Z_t} \\
&= \frac{e^{\sum_{t=1}^T -\alpha_t y_i h_t(x_i)}}{m \prod_{t=1}^T Z_t} \\
&= \frac{e^{-y_i \sum_{t=1}^T \alpha_t h_t(x_i)}}{m \prod_{t=1}^T Z_t}.
\end{aligned}$$

Step 2

Now we try to bound the empirical error $\widehat{\text{err}}(h)$. To do this, we first consider the exponential term

$$e^{-y_i \sum_{t=1}^T \alpha_t h_t(x_i)}.$$

If the final classifier makes a mistake, $\text{sign}(\sum_{t=1}^T \alpha_t h_t(x_i)) \neq \text{sign}(y_i)$. Thus $\text{sign}(y_i \sum_{t=1}^T \alpha_t h_t(x_i)) = -1$ and $-y_i \sum_{t=1}^T \alpha_t h_t(x_i) \geq 0$. Hence

$$e^{-y_i \sum_{t=1}^T \alpha_t h_t(x_i)} \geq 1 \quad \text{if} \quad h(x_i) \neq y_i.$$

Also, since $e^x \geq 0 \, \forall x$,

$$e^{-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)} \geq 0 \quad \text{if} \quad h(x_i) = y_i.$$

Hence

$$\begin{aligned}
\widehat{\text{err}}(h) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}_{h(x_i) \neq y_i} &\leq \frac{1}{m} \sum_{i=1}^{m} e^{-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)} \\
&\quad \text{(substituting from Step 1)} \\
&= \frac{1}{m} \sum_{i=1}^{m} m \prod_{t=1}^{T} Z_t \, D_{T+1}(i) \\
&= \sum_{i=1}^{m} D_{T+1}(i) \prod_{t=1}^{T} Z_t \\
&\quad \text{(since the sum of the weights is 1)} \\
&= \prod_{t=1}^{T} Z_t.
\end{aligned}$$

This gives $\widehat{\text{err}}(h) \leq \prod_{t=1}^{T} Z_t$.

Step 3

Finally, we bound the values of Z_t.

$$\begin{aligned}
Z_t &= \sum_{i=1}^{m} D_t(i) e^{-\alpha_t y_i h_t(x_i)} \\
&= \sum_{i:\ y_i = h_t(x_i)} D_t(i) e^{-\alpha_t} + \sum_{i:\ y_i \neq h_t(x_i)} D_t(i) e^{\alpha_t}
\end{aligned}$$

Here we have broken up the sum into 2 parts: the first is the sum over all points labeled correctly and the second is the sum over all points labeled incorrectly. Now since $\sum_{i:\ y_i \neq h_t(x_i)} D_t(i) = \epsilon_t$ and $\sum_{i:\ y_i = h_t(x_i)} D_t(i) = 1 - \epsilon_t$,

$$Z_t = (1 - \epsilon_t) e^{-\alpha_t} + \epsilon_t e^{\alpha_t}.$$

Now, we want to minimize $\prod_{t=1}^{T} Z_t$. To do this, we minimize each Z_t with respect to α_t.
Setting $\partial Z_t / \partial \alpha_t = 0$, we obtain

$$0 = \frac{\partial Z_t}{\partial \alpha_t} = -(1 - \epsilon_t) e^{-\alpha_t} + \epsilon_t e^{\alpha_t}.$$

Solving for α_t gives $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$. Note: it's ok to let α_t depend on ϵ_t because ϵ_t can be calculated from h_t alone.
Plugging in α_t,

$$Z_t = (1 - \epsilon_t) e^{-\alpha_t} + \epsilon_t e^{\alpha_t} = 2\sqrt{(1 - \epsilon_t)\epsilon_t}.$$

Now combining Steps 2 and 3,

$$
\begin{aligned}
\widehat{\mathrm{err}}(h) &\leq \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} 2\sqrt{(1-\epsilon_t)\epsilon_t} \\
&\quad (\text{using } \ \gamma_t = \frac{1}{2} - \epsilon_t) \\
&= \prod_{t=1}^{T} 2\sqrt{(\frac{1}{2} - \gamma_t)(\frac{1}{2} + \gamma_t)} \\
&= \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \\
&\quad (\text{using } \ 1 + x \leq e^x) \\
&\leq \prod_{t=1}^{T} \sqrt{e^{-4\gamma_t^2}} = \prod_{t=1}^{T} e^{-2\gamma_t^2} = e^{-2\sum_{t=1}^{T}\gamma_t^2} \leq e^{-2T\gamma^2}.
\end{aligned}
$$

This completes the proof. □

We will discuss the generalization error of AdaBoost in the next lecture.