COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Bryan Conroy

Lecture #9
March 3, 2008

1 Boosting

Under the PAC learning model that we have been discussing, learning algorithms are required to be capable of attaining arbitrarily small error probabilities. To review, a concept class C is *learnable* (under the PAC model) if

$\exists$ algorithm A
$\forall\ c \in C$
$\forall\ D$
$\forall\ \epsilon > 0$
$\forall\ \delta > 0$

A takes $m = poly\left(\frac{1}{\epsilon}, \frac{1}{\delta}, \cdots\right)$ examples and computes a hypothesis h such that

$$\Pr\left[err_D\left(h\right) \leq \epsilon\right] \geq 1 - \delta$$

However, suppose a learning algorithm cannot attain error rates less than some fixed amount, say 40%. Is it possible, despite this limitation, to still drive the error rate arbitrarily close to zero? This question motivates the weak learning model, which will be described below. This model will consider circumstances in which the learning algorithms can only do slightly better than random guessing, that is, error probability less than 50%.

A concept class C is *weakly learnable* if

$\exists\ \gamma > 0$
$\exists$ algorithm A
$\forall\ c \in C$
$\forall\ D$
$\forall\ \delta > 0$

A takes $m = poly\left(\frac{1}{\delta}, \cdots\right)$ examples to compute a hypothesis h such that

$$\Pr\left[err_D\left(h\right) \leq \frac{1}{2} - \gamma\right] \geq 1 - \delta$$

In other words, the algorithm A can, with high probability, do slightly better than random guessing.

One question we can ask is: given a fixed distribution D, is it the case that weak learning equals strong learning? The answer is no for general D, and this can be illustrated through the following example.

Example: Let $X = \{0,1\}^n \cup \{Z\}$, $C = \{$ all functions on $X\}$, and D be such that it puts probability $\frac{1}{4}$ on the point Z, and is uniform everywhere else. Given a set of examples, the algorithm will quickly learn $c\left(Z\right)$, since it is very likely that Z will appear amongst the examples. However, since it only sees a polynomial (in n) number of examples from the space of size 2^n length n binary strings, the algorithm is not expected to do much better

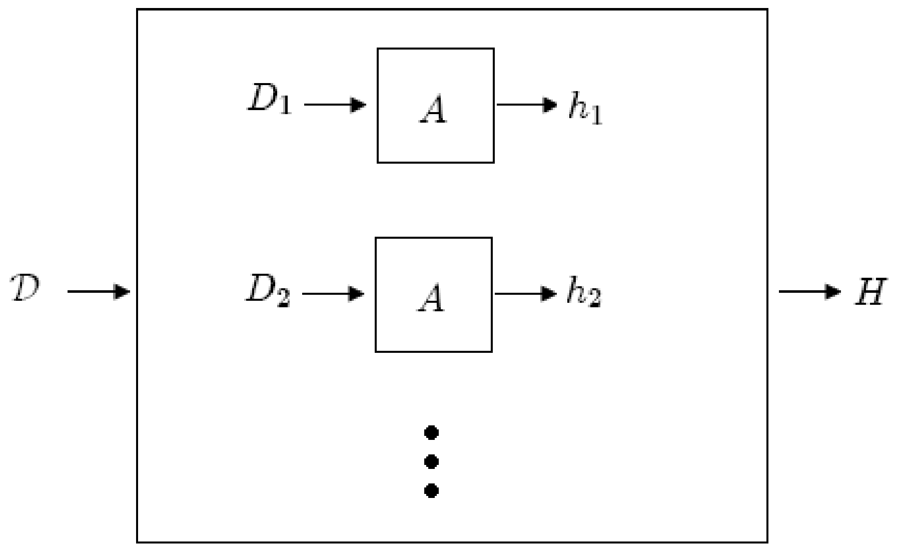

Figure 1: Generic boosting algorithm

than random guessing on this set. Assuming the algorithm learns $c(Z)$ correctly, the expected error rate of any learning algorithm is then

$$\begin{aligned} error &\approx \frac{1}{4}(0) + \frac{3}{4}\left(\frac{1}{2}\right) \\ &= \frac{3}{8}. \end{aligned}$$

In this case, given only a polynomial size example set, there is no way to drive the error much lower than $\frac{3}{8}$.

However, in the distribution free setting (PAC model), we want to prove the following theorem:

Theorem: Weak learning = strong learning under the PAC model

The proof is not immediate, but will instead follow from the construction of a boosting method, which is capable of developing strong learning algorithms from weak learning algorithms.

1.1 Boosting Algorithm

Here, the details of the boosting method are outlined.

Given:

1. m examples from some fixed, but unknown, distribution $\mathcal{D}$.

   Assume $S = \langle (x_1, y_1), \cdots, (x_m, y_m) \rangle$, with $y_i \in \{-1, +1\}$ $\forall i \in \{1, ..., m\}$.

   (The switch to $\{-1, +1\}$ for the label set makes the math more convenient).

2. Access to a weak learning algorithm A, producing hypotheses $h \in \mathcal{H}$.

Goal: Produce a new hypothesis H with error $\leq \epsilon$, for any fixed $\epsilon > 0$. Here, the output hypothesis H does not necessarily come from the same hypothesis space $\mathcal{H}$.

Intuitively, since we only have access to the weak learning algorithm, we might run A a number of times to produce a sequence of weak hypotheses. However, it is not sufficient to simply run A numerous times without modifying its input. For example, if A is deterministic, then running A multiple times on the same example set S will produce the same hypothesis over and over again. Instead, the input to A must give new and informative information on each round. Boosting accomplishes this by producing a new distribution from the original $\mathcal{D}$ for each round. A graphical representation of the boosting algorithm is shown in Figure 1.

Outline of a generic boosting algorithm:

```
for $t = 1, \cdots, T$ ($T = \#$ rounds)
    construct $D_t$, where $D_t$ is a distribution over the indices $\{1, \cdots, m\}$
    run $A$ on $D_t$, producing $h_t : X \to \{-1, +1\}$
    $\epsilon_t = err_{D_t}(h_t) = \frac{1}{2} - \gamma_t$, where $\gamma_t \geq \gamma \ \forall t$ by the weak learning assumption
end
output $H$ (a combination of the weak hypotheses $h_1, \cdots, h_T$)
```

Although the algorithm is outlined above, there are still two questions that need to be addressed. First, how do we design the distributions $D_1, D_2, \cdots, D_T$ from the true distribution $\mathcal{D}$ such that A is forced to provide new and useful information on each round? Also, once the rounds have completed, how do we combine the sequence of weak hypotheses into the strong hypothesis H? To answer these questions, we will look at a popular boosting algorithm called AdaBoost.

1.2 AdaBoost

AdaBoost uses a recurrent construction for the distributions D_t. The idea is to increase the weights on "hard" examples (examples which have been previously misclassified on earlier rounds), while at the same time decreasing the weights on "easy" examples (examples which have been classified correctly).

$$\begin{aligned} D_1(i) &= \frac{1}{m} \\ D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases} \\ &= \frac{D_t(i)}{Z_t} \exp(-y_i \alpha_t h_t(x_i)) \end{aligned}$$

where $\alpha_t > 0$ and Z_t is a normalizing weight to ensure that D_{t+1} is a probability distribution. Also, note that D_t is a probability distribution over the indices $\{1, \cdots, m\}$ so that $D_t(i)$ represents the probability of the pair (x_i, y_i). For the time being, α_t is any fixed positive constant. Later, an optimal value for α_t will be chosen to minimize a bound on the training error $\widehat{err}(H)$.

The output hypothesis H is constructed from the weak hypotheses by:

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

which is a weighted voting rule.

One of the reasons AdaBoost has become so popular is that it is more practical for applications because it does not depend on the weak learning parameter γ. In this sense, the algorithm can *adapt* to the weak learning algorithm (hence, the name AdaBoost). Additionally, as the next theorem will show, the training error of a hypothesis H generated by AdaBoost decays exponentially fast in the number of rounds T.

Theorem: Let H be the output hypothesis of AdaBoost. Then:

$$\begin{aligned} \widehat{err}(H) &\leq \prod_{t=1}^{T}\left(2\sqrt{\epsilon_t(1-\epsilon_t)}\right) \\ &= \prod_{t=1}^{T}\sqrt{1-4\gamma_t^2} \\ &\leq \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right) \end{aligned}$$

where the last line is obtained from the inequality $1 + x \leq e^x$. Thus, if $\gamma_t \geq \gamma\ \forall t$ by the weak learning assumption, then

$$\widehat{err}(H) \leq \exp\left(-2\gamma^2 T\right)$$

so that the error dies exponentially fast in T.

Proof: The proof follows in three steps:

1. First, obtain the following expression for $D_{T+1}(i)$:

   $$D_{T+1}(i) = \frac{\exp(-y_i f(x_i))}{m\prod_{t=1}^{T} Z_t}$$

   where $f(x) = \sum_{t=1}^{T}\alpha_t h_t(x)$.

   Proof: Since $D_{t+1}(i)$ is given by:

   $$D_{t+1}(i) = \frac{D_t(i)}{Z_t}\exp(-y_i\alpha_t h_t(x_i))$$

   we can unravel the recurrence to obtain:

   $$\begin{aligned} D_{T+1}(i) &= \frac{1}{m}\times\frac{e^{-y_i\alpha_1 h_1(x_i)}}{Z_1}\times\cdots\times\frac{e^{-y_i\alpha_T h_T(x_i)}}{Z_T} \\ &= \frac{1}{m}\prod_{t=1}^{T}\frac{e^{-y_i\alpha_t h_t(x_i)}}{Z_t} \\ &= \frac{\exp\left(-y_i\sum_{t=1}^{T}\alpha_t h_t(x_i)\right)}{m\prod_{t=1}^{T}Z_t} \\ &= \frac{\exp(-y_i f(x_i))}{m\prod_{t=1}^{T}Z_t} \end{aligned}$$

2. Second, we bound the training error of H by the product of the normalizing weights Z_t:

$$\widehat{err}(H) \leq \prod_{t=1}^{T} Z_t.$$

Proof:

$$\begin{aligned}
\widehat{err}(H) &= \frac{1}{m}\sum_{i=1}^{m} 1\left(y_i \neq H\left(x_i\right)\right) \\
&= \frac{1}{m}\sum_{i=1}^{m} 1\left(y_i f\left(x_i\right) \leq 0\right) \\
&\leq \frac{1}{m}\sum_{i=1}^{m} e^{-y_i f(x_i)} \\
&= \frac{1}{m}\sum_{i=1}^{m} m\left(\prod_{t=1}^{T} Z_t\right) D_{T+1}(i) \\
&= \left(\prod_{t=1}^{T} Z_t\right)\sum_{i=1}^{m} D_{T+1}(i) \\
&= \prod_{t=1}^{T} Z_t
\end{aligned}$$

where $1(\cdot)$ is the indicator function. The third line comes from the observation that when $1\left(y_i f\left(x_i\right) \leq 0\right) = 1$, then $y_i f\left(x_i\right) \leq 0$ and so $\exp\left(-y_i f\left(x_i\right)\right) \geq 1 = 1\left(y_i f\left(x_i\right) \leq 0\right)$. (Also, when $1\left(y_i f\left(x_i\right) \leq 0\right) = 0$, then $\exp\left(-y_i f\left(x_i\right)\right) \geq 0$ trivially). The fourth line follows from step 1. The last line is obtained from the fact that D_{T+1} is a probability distribution over the examples.

3. Now that the training error has been bounded in step 2 by the product of the normalizing weights Z_t, the last step is to express Z_t in terms of ϵ_t:

$$Z_t = 2\sqrt{\epsilon_t\left(1-\epsilon_t\right)}.$$

Proof:

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

where the last step comes from the fact that

$$\sum_{i:y_i \neq h_t(x_i)} D_t(i) = \epsilon_t$$

since the sum is taken over examples misclassified, which gives the error for the t^{th} round.

Since the expression above for Z_t is valid for all α_t, minimizing Z_t with respect to α_t for each t will produce the minimum training error $\widehat{err}\left(H\right)$. Taking the derivative of Z_t with respect to α_t and setting equal to zero, we obtain:

$$0 = -\left(1 - \epsilon_t\right) e^{-\alpha_t} + \epsilon_t e^{\alpha_t}$$

Solving, we find:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{\left(1 - \epsilon_t\right)}{\epsilon_t} \right).$$

Using this value of α_t in the expression for Z_t, and then plugging that into the bound on the training error for H, we end up with

$$\widehat{err}\left(H\right) \leq \prod_{t=1}^{T} \left(2\sqrt{\epsilon_t \left(1 - \epsilon_t\right)} \right)$$

which proves the theorem.