CS 2416 - Machine Learning Theory

Lecture #8: 3 November 2005

Lecturer: Toniann Pitassi

Scribe Notes by: Toniann Pitassi

1 Relaxed Definitions of PAC Learning

Recall that we say a learning algorithm $\mathcal{A}$ efficiently PAC learn concept class $\mathcal{C}$ is that given any distribution $\mathcal{D}$, $\epsilon > 0$ and $\delta > 0$, with probability $1 - \delta$, algorithm $\mathcal{A}$ should produce a hypothesis $h \in \mathcal{H}$ with error at most ϵ in polynomial time w.r.t $n, 1/\epsilon, 1/\delta$.

We consider the following weaker definitions by fixing error bound ϵ or confidence interval parameter δ.

(D1) δ is fixed and smaller than 1/2.

(D2) ϵ is fixed and smaller than 1/2.

It is not very hard to show that a learner satisfying the weaker condition D1, implies PAC learning. The book gives this argument. (We may have time to present it in a later lecture.)

To see that a learner satisfying D2 implies PAC learning is a much deeper result, and algorithms which accomplish this are referred to as *boosting methods*. (Thus boosting refers to a meta-algorithm that takes a "weak" D2-type learner that can acheive some small advantage over random guessing, and "boost" this algorithm to obtain a new one that acheives a much lower error probability.)

There are two general boosting methods. One is boosting by filtering proposed by Schapire[**?**]. The other is boosting by sub-sampling proposed by Freund and Schapire[**?**]. Freund and Schapire recently won the 2003 Godel paper for the latter paper. In both of these methods the overall approach is as follows. We have an algorithm, call it A, that weak-learns with respect to any distribution, and we want to use A to develop another algorithm, call it B, to strong-learn with respect to a particular distribution D. (This approach will work for any D in the same way.)

In boosting by filtering, algorithm B begins by running A on examples chosen according to D. This returns a hypothesis, h_0 which has some error ϵ on D. Now B will rerun A on a new distribution, D_1, chosen so that the examples that were incorrectly classified the first time around now have higher weight, and the examples that were correctly classified the first time around, now have lower (relative) weight. However doing this is tricky since B can only ask for examples from an oracle that chooses the samples from the original distribution D. Thus, D_1 is simulated by filtering. That is, B sees a stream of examples from D, and takes some and rejects some, so that the actual examples fed to A look as though they are from D_1. At the end, B returns a hypothesis that is a weighted majority of the hypotheses returned at each stage. Boosting by filtering is tricky to implement for several reasons. First, one needs to estimate the actual errors ϵ_i made by each of the hypotheses h_i. And secondly, the process of creating a distribution by filtering that looks like D_i is tedious.

In contrast boosting by subsampling works by doing everything not with respect to the real distribution D, but instead with respect to a set of samples chosen according to D. Now algorithm B initially takes some number, m, of examples (chosen according to D). Call this set of samples S. We can think of the examples from S as being uniformly distributed when given to A. Now B runs A on these examples and gets a hypothesis h_0 which has some error ϵ on S. Now B will reweight the samples from S, giving higher weight to the examples that were misclassified the first time around, and lower (relatiave) weight to the examples that were correctly classified the second time around. Now B reruns A on this new distribution of S. Again this repeats for some number of stages, and at the end of the stages, B returns a hypothesis that is a weighted majority of the hypotheses returned at each stage. The above problems become much simpler in this context. First, it is easy to calculate the errors of the hypotheses with respect to the weighted sets S, because S is a fixed polynomial-sized set of samples that algorithm B has full access to. And similarly, it is easy to sample according to a weighted set S again because S is a fixed polynomial-sized set. However, we have not accomplished as much in boosting by subsampling, since in the end, all we've produced here is an algorithm B' that is guaranteed to have very low error with respect to the *training set*. (This is called the empirical error as opposed to the true error.) Thus one still has to show generalization (after boosting by subsampling); that is, we still need to show that low empirical error implies low acual error (with respect to D) with high probability. This last step can be proven similarly to arguments showing that good OCCAM algorithms imply PAC learning.

The book describes Schapire's original approach in detail, which is a boosting by filtering method. In our lectures, we will discuss the latter method (boosting by subsampling) since it is much more practical to implement (as argued above), and it gives theoretical results which are as strong as Schapire's original method (boosting by filtering).

2 Adaboost

Today we will prove a beautiful result showing that weak learning is equivalent to strong learning. As mentioned in the last lecture, this result is originally due to Schapire, and that proof can be found in the book. We will present Freund and Schapire's later proof and the corresponding Adaboost algorithm.

Recall the following definitions of weak and strong learning.

Weak Learn: $\exists \gamma > 0$, $\exists$ algorithm A, $\forall c \in C$, $\forall D$, $\forall \delta > 0$, A computes h, such that

$$Pr_D[err(h) > \frac{1}{2} - \gamma] =\leq \delta$$

The running time is polynomial in $(\frac{1}{\delta}, n)$.

Strong Learn: $\exists$ Algorithm A, $\forall c \in C$, $\forall D$, $\forall \epsilon > 0$, $\forall \delta > 0$, A computes h, such that

$$Pr_D err(h) > \epsilon] \leq \delta$$

The running time is polynomial in $(\frac{1}{\delta}, \frac{1}{\epsilon}, n)$.

Note: We cannot go from a weak learning algorithm to a strong learning algorithm if the distribution is fixed. For example, let the concept class C be all functions on $\{0,1\}^n \bigcup \{Z_0\}$.

$Pr[Z_0] = \frac{1}{4}$. The remaining 2^n points have uniform distribution. Assuming Z_0 got its label correct, the total error rate is approximately $(3/4)(1/2) = 3/8$. On the other hand, for this fixed distribution, since the concept class is so large, there is no way to get error significantly below $(3/8)$ with a polynomial-sized sample. (This can be proven rigorously.)

The general idea of boosting by subsampling is as follows. This is another application of the multiplicative weights update method.

- Pick a polynomial sized sample of inputs from distribution D_1, say, x_1, ..., x_m.
- Run weak learning algorithm on x_1, ..., x_m distributed according to D_1, to get h_1.
- Reweight $x_1, \ldots, x_n$ by putting more weight on the elements that we've made mistakes on. This new distribution Is called D_2. Run weak learning algorithm on x_1, ..., x_m distributed according to D_2, to get h_2
- ...
- Reweight $x_1, \ldots, x_n$ to get distribution D_3. Run weak learning algorithm on x_1, ..., x_m distributed according to D_q, to get h_q
- The algorithm then outputs a weighted majority of h_1, h_2, ..., h_q, say, h'.

The Adaboost algorithm is as follows.

Choose a set S of labelled examples from the underlying distribution $\mathcal{D}$. Let S consist of: (x_1, y_1), ..., (x_m, y_m), where $y_i \in \{0, 1\}$
Initialize $w_i(i) = 1$. (Initially all items have the same weight.)
The associated probability distribution of D_1 is $Pr[x_i] = \frac{w_i(i)}{\sum_i w_i(i)}$.
For $t = 1$ to T

- Run the weak learning algorithm on S using distribution D_t, to get hypothesis h_t.
- Let $\epsilon_t = Pr_{D_t}[h_t(x_i) \neq y_i]$
- $w_{t+1}(i) = \begin{cases} w_t(i) \cdot \frac{\epsilon_t}{1-\epsilon_t} & \text{if } h_t(x_i) = y_i \\ w_t(i) \cdot 1 & \text{if } h_t(x_i) \neq y_i \end{cases}$

Output $H(x) = 1$ iff $\sum_{i=1}^{T} \alpha_t h_t(x) > \frac{1}{2}$, where $\alpha_t = \frac{\ln(\frac{1-\epsilon_t}{\epsilon_t})}{\sum_{i=1}^{T} \ln(\frac{1-\epsilon_i}{\epsilon_i})}$

3 Bounding Empirical Error

We want to prove that the output of Adaboost, $H(x)$ is a strong learner. We will prove this in two stages. First, we will show that the error of H with respect to the training set S is very small. (This is called the empirical error.) Then we will show that this implies that the overall error is also very small (with respect to distribution D) with high probability. The following theorem gives us the bound that we want on the empirical error.

Theorem 1 $\widehat{error}(H) \leq \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)}$

Proof:

We will first prove the simpler case where $\epsilon_1 = \epsilon_2 = \ldots = \epsilon_t$.

Claim 1: All weights go down.
Let W_i denote the total sum of the weights after i iterations.

$$\begin{aligned} W_1 &= \sum_i w_1(i) \;=\; m \\ W_2 &= \epsilon W_1 + (1-\epsilon)(\frac{\epsilon}{1-\epsilon})w_1 \\ &= 2\epsilon W_1 \\ &\vdots \\ W_T &= (2\epsilon)^T W_1 = (2\epsilon)^T m \end{aligned}$$

Claim 2: Any bad example (where H gives the wrong answer) has high weight in the end.

Fix a bad example x. Since each $\epsilon_i = \epsilon$, $\alpha_T = 1/T$, so if x is a bad example, at least half of all h_i's $i \leq T$ get the incorrect answer on x. Thus the weight of x gets multiplied by $\frac{\epsilon}{1-\epsilon}$ at least $T/2$ times. Thus the final weight of x is at least $\frac{\epsilon}{1-\epsilon}^{T/2}$ after T steps.

Now if the final hypothesis H (after T steps) has error E, then there are at least Em bad examples. Thus the total weight at the end is at least $\frac{\epsilon}{1-\epsilon}^{T/2}EM$.

Putting these two claims together, we have:

$$\begin{aligned} (2\epsilon)^T m &\geq (\frac{\epsilon}{1-\epsilon})^{\frac{T}{2}} Em \\ E &\leq [2\sqrt{\epsilon(1-\epsilon)}]^T \end{aligned}$$

We will now prove the more general theorem stated above. We will slightly reformulate the Adaboost algorithm. As Avner Magen pointed out, this is really the same algorithm as the one above since the relative differences between the new weight of i when it is labelled correctly versus incorrectly, is the same.

Given: (x_1, y_1), ..., (x_m, y_m), where $y_i \in \{-1, 1\}$
Initialize $D_1(i) = 1/m$
For $t = 1$ to T

- Train weak learner using distribution D_t, get hypothesis h_t
- Let $\epsilon_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$
- $D_{t+1}(i) = \begin{cases} \frac{D_t(i)}{Z_t} \cdot e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ \frac{D_t(i)}{Z_t} \cdot e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$

Output $H(x) = sign(\sum_{t=1}^T h_t(x)\alpha_t)$, where $sign = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$

We will now prove the above theorem. Again the proof proceeds in two steps.

1. $\widehat{error}(H) \leq$ Product of all of the normalizing factors.

We will need the following claim, which follows from the definitions.

Claim:

$$\begin{aligned}
D_{T+1}(i) &= \frac{D_T(i) \cdot exp(-\alpha_T y_i h_T(x_i))}{Z_T} \\
&= \frac{D_{T-1}(i) \cdot exp(-\alpha_{T-1} y_i h_{T-1}(x_i)) \cdot exp(-\alpha_T y_i h_T(x_i))}{Z_{T-1} Z_T} \\
&= \frac{1}{m} \frac{exp(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i))}{\prod_{t=1}^{T} Z_t}
\end{aligned}$$

We will now show that the error is at most the product of the normalizing factors.

$$\begin{aligned}
\widehat{error}(H) &= \frac{1}{m} \sum_{i=1}^{m} [[y_i \neq H(x_i)]] \\
&= \frac{1}{m} \sum_{i=1}^{m} [[y_i f(x_i) \leq 0]] \\
&\leq \frac{1}{m} \sum_{i=1}^{m} e^{-y_i f(x_i)} \\
&= \frac{1}{m} \sum_{i=1}^{m} D_{T+1}(i) \cdot m \cdot \prod_{t=1}^{T} Z_t \quad \text{(from above claim)} \\
&= \frac{1}{m} \prod_{t=1}^{T} Z_t \underbrace{\sum_{i=1}^{m} D_{T+1}(i)}_{Distribution=1} \cdot m \\
&= \prod_{t=1}^{T} Z_t
\end{aligned}$$

2. The final weight is small. $Z_t \leq 2\sqrt{\epsilon_t(1-\epsilon_t)}$

$$\begin{aligned}
Z_t &= \sum_{i=1}^{m} D_t(i) e^{-\alpha_t y_i h_t(x_i)} \\
&= \sum_{i:h_t(x_i) \neq y_i} D_t(i) \cdot e^{\alpha_t} + \sum_{i:h_t(x_i) = y_i} D_t(i) \cdot e^{-\alpha_t} \\
&= \epsilon_t e^{\alpha_t} + (1-\epsilon_t) e^{\alpha_t} \\
&= 2\sqrt{\epsilon_t(1-\epsilon_t)}
\end{aligned}$$

The last line follows from the previous by setting $\alpha = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$ so that Z_T is minimized.

From 1 and 2, we have $\widehat{error}(H) \leq \prod_{i=1}^{T} Z_t \leq \prod_{i=1}^{T} 2\sqrt{\epsilon_t - (1-\epsilon_t)}$