COS 511: Foundations of Machine Learning

Rob Schapire
Scribe: Kevin Ko

Lecture #9
March 7, 2006

Strong Learning	**Weak Learning**
$\exists$ algorithm A	$\exists$ algorithm A
$\forall c \in \mathcal{C}$	$\exists \gamma > 0$
$\forall D$	$\forall c \in \mathcal{C}$
$\forall \epsilon > 0$	$\forall D$
$\forall \delta > 0$	$\forall \epsilon \geq \frac{1}{2} - \gamma$
	$\forall \delta > 0$
A produces $h \in \mathcal{H}$:	A produces $h \in \mathcal{H}$:
$Pr[err(h) > \epsilon] \leq \delta$	$Pr[err(h) > \epsilon] \leq \delta$

Table 1: Strong & Weak Learnability

In the previous lecture, we defined the notion of weak learnability for a concept class $\mathcal{C}$. Under weakly learnable conditions, the constraint on ϵ is loosened so that A need only drive the error below $1/2$ (ie. better than guessing). This differs from the strongly learnable criteria of PAC, which requires that A allow the error to be made arbitrarily small. Table 1 compares the two.

Our motivating question asks whether weak learning is equivalent to strong learning.

1 A Scenario

It is educational to examine how conditions on our learning models affect the link between weak and strong. For example, suppose we ignore the general distribution requirement and let our learning algorithms accept a fixed $\mathcal{C}$ and D. Are the two models equivalent? In this case, the answer is no; consider:

$$\begin{aligned} \mathcal{C} &= \text{all functions over } \{0,1\}^n \cup \{Z\} \\ D(x) &= \begin{cases} 1/2, & x = Z \\ 1/2^{n+1}, & x \in \{0,1\}^n. \end{cases} \end{aligned}$$

Here, the distribution D is chosen so that point Z will occur $1/2$ the time and so will be easily learnable. The remaining time, a vector in $\{0,1\}^n$ will be presented uniformly at random, making it difficult to characterize all possible elements.

Optimistically, we can assume that a reasonable A will produce an $h \in \mathcal{H}$ that correctly classifies Z; this ensures that the overall error, with respect to D, will be no more than $1/2$. Extending this, a reasonable A should also do at least as well as random guessing when classifying some $x \in \{0,1\}^n$. However, a polynomial number of examples cannot possibly cover the input space, and so the vectors $\{0,1\}^n$ will be misclassified roughly $1/2$ the time.

Examining the error rate for such a "reasonable" h:

$$err_D(h) \approx \underbrace{\frac{1}{2} \cdot 0}_{\text{from } Z} + \underbrace{\frac{1}{2} \cdot \frac{1}{2}}_{\text{from } \{0,1\}^n} = \frac{1}{4}$$

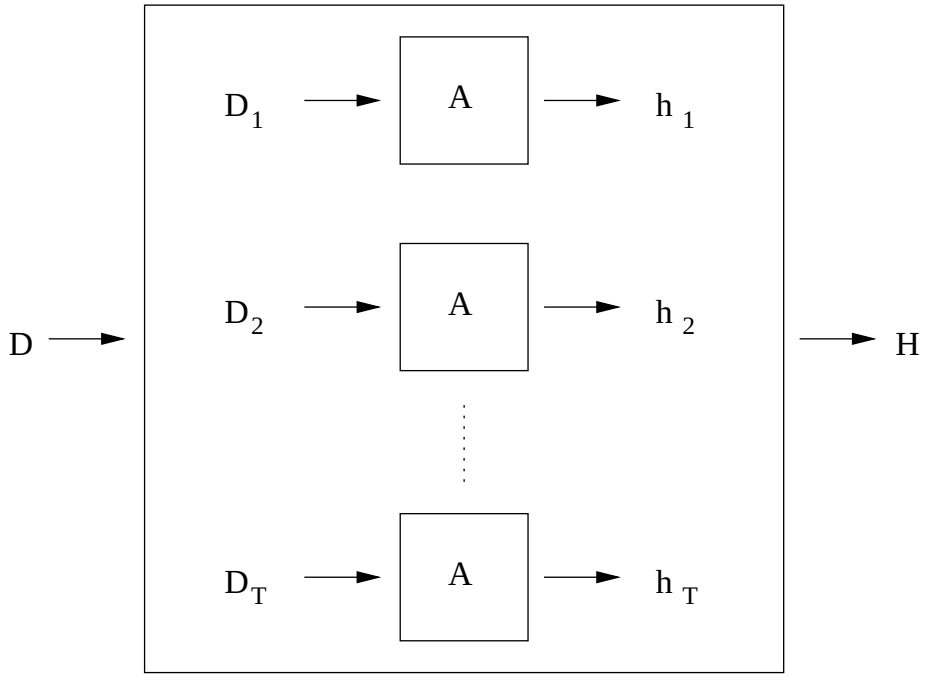

Figure 1: Diagram of generic boosting

Clearly, this instance is weakly learnable. We wish to consider the existence of a method that will convert a weak hypothesis like h into a strong one. The polynomial constraint on the number of examples prevents one from arbitrarily reducing the error over this choice of D, which suggests that such a method cannot exist.

2 Boosting

At the highest level, boosting can be viewed as deriving several hypotheses from an initial set of samples, $S = \langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$, and combining them into one much more powerful hypothesis. Here, we will focus on boosting approaches that take an initial S and sample from it according to some varying distribution.

Figure 1 illustrates this notion of boosting. D represents the initial distribution of examples, and D_t are the distributions chosen by the method. While it is entirely possibly for some of the D_t's to be identical, this may not be the best approach since it could lead to duplicate hypotheses.

Pseudo-code for this kind of boosting approach is in Figure 2. In general, boosting algorithms are distinguished by the choice of D_t and the merging of the hypotheses h_t.

3 AdaBoost

Previously, we labeled examples from the set $\{0, 1\}$. This is not always convenient from a mathematical perspective, so in the following we use labels from $\{-1, 1\}$. Moreover, let $D_t(i)$ denote distribution D_t's weighting for sample (x_i, y_i).

for $t = 1, 2, \ldots, T$
    construct distribution D_t
    run (weak-learning) algorithm A on D_t
        get hypothesis h_t where
        $\epsilon_t = err_{D_t}(h_t) \leq \frac{1}{2} - \gamma$
output H

Figure 2: Pseudo-code for generic boosting

Now, we introduce a simple, yet practically effective, boosting algorithm called AdaBoost. AdaBoost adheres to the intuition that a learning algorithm should focus on difficult examples. Thus, it begins initially with all samples weighted equally. It then iteratively revises the distribution to focus on hard (mis-classified) cases and de-emphasize easy (correctly classified) ones.

More concretely:

Initially, $D_1(i) = \frac{1}{m}$, $\forall i$. Then, update the distribution according to the following template:

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

where $\alpha_t > 0$ and Z_t is a normalizing factor to make D_{t+1} a distribution.

Notice that $e^{\alpha_t} > 1$ ($e^{-\alpha_t} < 1$); as desired, this choice (de)emphasizes the i-th example if it was (not) mis-classified. Performing this iterative computation T times will construct T distributions and hypotheses. AdaBoost merges the result using a weighted majority vote:

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

where $sign(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0. \end{cases}$

Realize that this recursive definition can lead to a cycle and that it is an open problem whether cycles will always occur. However, cycles are not necessarily harmful in this context. To help determine α_t or Z_t it will be instructive to prove the following result.

3.1 The Training Error of AdaBoost

Theorem

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

First, we observe a few consequences of this theorem. Since ϵ_t is the error rate of the weak hypothesis h_t on D_t, $\epsilon_t = 1/2 - \gamma_t \leq 1/2 - \gamma \Rightarrow \gamma_t \geq \gamma$. Substituting this into the

right-hand expression:

$$\hat{err}(H) \leq \prod_{t=1}^{T} \sqrt{4(1/2-\gamma_t)(1/2+\gamma_t)} = \prod_{t=1}^{T} \sqrt{1-4\gamma_t^2}.$$

Applying our favorite exponential bound:

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

Hence, this theorem implies that the training error rate falls exponentially with T.

As an interesting note, the right-hand quantity in the theorem can be rewritten in terms of cross-entropy:

$$\exp\left(-RE\left(\frac{1}{2}||\epsilon_t\right)\right) = (2\epsilon_t)^{1/2} \cdot (2(1-\epsilon_t))^{1/2} = 2\sqrt{\epsilon_t(1-\epsilon_t)}.$$

Below, $\prod_t$ will act as shorthand for $\prod_{t=1}^{T}$ and $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$, where $f(x)$ represents the sum within AdaBoost's majority vote: $H(x) = sign(f(x))$. The proof proceeds in three steps.

Step 1.

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

Our choice of labels implies that $h_t(x_i)y_i$ is 1 when $h_t(x_i)$ and y_i agree and -1 otherwise. As a result, the quantity $\exp(-\alpha_t h_t(x_i)y_i)$ exactly corresponds to our conditional assignment in the recursive definition of $D_{t+1}(i)$.

The rest immediately follows when unrolling the construction of $D_{t+1}(i)$:

$$\begin{aligned}
D_{T+1}(i) &= \frac{1}{m}\frac{\exp(-\alpha_1 h_1(x_i)y_i)}{Z_1} \cdots \frac{\exp(-\alpha_T h_T(x_i)y_i)}{Z_T} \\
&= \frac{\exp(-y_i \sum_t \alpha_t h_t(x_i))}{m \prod_t Z_t} \\
&= \frac{\exp(-y_i f(x_i))}{m \prod_t Z_t}.
\end{aligned}$$

Step 2.

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

Proof:

Let $[\![\pi]\!] = \begin{cases} 1, & \text{if } \pi \text{ holds} \\ 0, & \text{else.} \end{cases}$

Observe that the following inequality holds:

$$[\![x \leq 0]\!] \leq \exp(-x)$$

(at $x \leq 0$, $[\![x \leq 0]\!] = 1 = \exp(0) \leq \exp(-x)$; when $x > 0$, $[\![x \leq 0]\!] = 0 < \exp(-x)$).

Then,

$$\begin{aligned} \hat{err}(H) &= \frac{1}{m}\sum_{i=1}^{m}[\![H(x_i) \neq y_i]\!] \\ &= \frac{1}{m}\sum_{i=1}^{m}[\![y_i f(x_i) \leq 0]\!] \\ &\leq \frac{1}{m}\sum_{i=1}^{m}\exp(-y_i f(x_i)). \end{aligned}$$

Applying the result from Step 1 and the fact that D_{T+1} is a distribution,

$$\begin{aligned} \hat{err}(H) &\leq \sum_{i=1}^{m} D_{T+1}(i)\prod_{t=1}^{T} Z_t \\ &= \prod_{t=1}^{T} Z_t. \end{aligned}$$

Step 3.

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

Since Z_t is a normalization constant, we have

$$Z_t = \sum_{i=1}^{m} D_t(i)\exp(-\alpha_t h_t(x_i)y_i)).$$

Splitting the sum:

$$\begin{aligned} Z_t &= \sum_{i:h_t(x_i)=y_i}[D_t(i)\exp(-\alpha_t h_t(x_i)y_i))] + \sum_{i:h_t(x_i)\neq y_i}[D_t(i)\exp(-\alpha_t h_t(x_i)y_i))] \\ &= \underbrace{\sum_{i:h_t(x_i)=y_i} D_t(i)}_{1-\epsilon_t}\exp(-\alpha_t) + \underbrace{\sum_{i:h_t(x_i)\neq y_i} D_t(i)}_{(\text{error})\ \epsilon_t}\exp(\alpha_t) \\ &= (1-\epsilon_t)\exp(-\alpha_t) + \epsilon_t\exp(\alpha_t). \end{aligned}$$

Recall from Step 2 that $\hat{err}(H) \leq \prod_t Z_t$. Since $\alpha_t > 0$ is a free-parameter we can minimize the training error by minimizing each Z_t:

$$\begin{aligned} & \tfrac{d}{d\alpha_t}[(1-\epsilon_t)\exp(-\alpha_t) + \epsilon_t \exp(\alpha_t)] = 0 \\ \Rightarrow \quad & (1-\epsilon_t)\exp(-\alpha_t^*) = \epsilon_t \exp(\alpha_t^*) \\ \Rightarrow \quad & \alpha_t^* = \tfrac{1}{2}\ln\tfrac{1-\epsilon_t}{\epsilon_t}. \end{aligned}$$

Substituting,

$$\begin{aligned} Z_t &= ((1-\epsilon)\epsilon_t)^{1/2} + (\epsilon_t(1-\epsilon_t))^{1/2} \\ &= 2\sqrt{\epsilon_t(1-\epsilon_t)}. \end{aligned}$$

4 Summary

The theorem proved in lecture today provides a bound on the empirical error $\hat{err}(H)$. We will proceed next time by viewing AdaBoost in terms of its underlying learning problem. In particular, because $H(x) = sign(\sum_t \alpha_t h_t(x_i))$, $h_t \in \mathcal{H}$, we will examine H as a member of class $\mathcal{G}$, where $\mathcal{G} = \{\text{linear threshold functions over } h_t \in \mathcal{H}\}$.

Ultimately, we want to bound the true error (using dichotomies of $\mathcal{G}$):

$$err(H) \leq \hat{err}(H) + O\left(\sqrt{\frac{\ln(\Pi_{\mathcal{G}}(2m)) + \ln(1/\delta)}{m}}\right).$$

Exercise: How does one count the dichotomies of $\mathcal{G}$?