**COS 511: Theoretical Machine Learning**

Lecturer: Rob Schapire
Scribe: Yannis Karakozis

Lecture 11
March 12, 2018

---

## Strong vs Weak PAC-Learning:

So far in this class, we have shown that if we can PAC-learn a concept class $C$, then we can achieve a particular level of accuracy ($\epsilon$) at a particular level of confidence ($\delta$) by only requiring a polynomial amount of data with respect to the levels of accuracy and confidence. However, in real life, we do not have an infinite amount of data to achieve 100% accuracy with 100% confidence. In particular, a lot of the time, our algorithms can reliably achieve only a low level of accuracy with high confidence. Therefore, a natural question to ask is, if we manage to develop a weak-learner, i.e. an algorithm that can learn $C$ (i.e. reliably perform better than unbiased coin flips), but with a low level of accuracy, how can we automatically and reliably generate a strong learner, an algorithm that can achieve high accuracy with high confidence?

In this lecture, we will examine a boosting algorithm that achieves this goal. To understand how this algorithm work, we have to distinguish between two types of PAC-learning:

**Definition:** A concept class $C$ is strongly PAC-learnable if there exists an algorithm A such that, $\forall c \in C$, $\forall D$, $\forall \epsilon > 0$, $\forall \delta > 0$, if A is given $m = poly(\frac{1}{\delta}, \frac{1}{\epsilon})$ examples, then it outputs $h$ such that:

$$P(err_D(h) \leq \epsilon) \geq 1 - \delta$$

**Definition:** A concept class $C$ is weakly PAC-learnable if $\exists \gamma > 0$, there exists an algorithm A such that, $\forall c \in C$, $\forall D$, $\forall \delta > 0$, if A is given $m = poly(\frac{1}{\delta})$ examples, then it outputs $h$ such that:

$$P(err_D(h) \leq 0.5 - \gamma) \geq 1 - \delta$$

The big difference between the two definitions are the terms $\gamma$ and $\epsilon$ that measure error. So the link we are trying to achieve is whether an algorithm that can achieve a fairly low level of accuracy reliably for all concepts $c$ and distributions $D$ can be "boosted" into an algorithm that can do the same for arbitrarily high levels of accuracy when given enough samples. If we also consider algorithmic efficiency, then our final goal becomes finding an efficient boosting algorithm that boosts a weak learner into a strong learner, so that if the weak learner is efficient, then the strong learner that results from applying boosting will also be efficient. This means that either a concept class is learnable "all the way", or it is not learnable at all, not even partially.

## Boosting - The Theorem:

**Theorem:** A concept class $C$ is weakly PAC-learnable if and only if it is strongly PAC-learnable.

This theorem implies that learning is an all or nothing phenomenon. In other words, if you can find an algorithm that achieves a low level of accuracy in learning $C$, then there exists an algorithm that can do the same with a high level of accuracy.

The goal is to now prove the Theorem while working under the distribution-free model. In other words, we will show that, assuming the existence of a weak learning algorithm, we can develop a strong learning algorithm, without making any assumptions on $D$, by taking advantage of the fact that the weak learner must be effective for every target distribution $D$. The reverse direction trivially follows.

## Boosting - Proof Setup:

Assume we are given random sample $S = \langle (x_1, y_1), ..., (x_m, y_m) \rangle$, where $\forall i$, $x_i \in X$, $y_i \in \{-1, +1\}$ and $(x_i, y_i) \sim D$. Assume we also have access to a weak-learner A (i.e. an algorithm that can weakly PAC-learn the target class $C$, according to the definition above).

A good way of thinking about A is as an algorithm, such that, $\forall D$, $\forall \delta > 0$, when given $m = poly(1/\delta)$ examples from $D$, it computes a hypothesis $h$ such that, given a fixed $\gamma$:

$$P(err_D(h) \leq 0.5 - \gamma) \geq 1 - \delta$$

What we need to do now is identify a boosting algorithm that, using A, finds a boosted hypothesis $H$ such that $err_D(H) \leq \epsilon$. The trick we will use is the fact that A generates a weak hypothesis $h$ for any distribution $D$, so we can create different distributions $D_t$ such that we generate multiple weak $h_t$ to ultimately combine them into a stronger hypothesis $H$.

## The AdaBoost Algorithm:

The boosting algorithm we will consider and prove the Theorem for is AdaBoost. Here are the steps of AdaBoost:

0. Take in input sample $S = \langle (x_1, y_1), ..., (x_m, y_m) \rangle$. Define $D_1(i) = \frac{1}{m}, \forall i$. Pick the number of iterations $T$. Then, for $t = 1, ..., T$ repeat the following steps:

1. Run A on $D_t$ to get weak hypothesis $h_t : X \to \{-1, 1\}$. We can achieve this because of the distribution-free assumption of weak PAC-learning.

2. Compute the generalization error on $S$: $\epsilon_t = err_{D_t}(h_t) = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$. We also write $err_{D_t}(h_t)$ as $0.5 - \gamma_t$. We call the term $\gamma_t$ the "edge", as it measures how far away the error is from the average error of an unbiased coin-hypothesis.

3. Compute $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t}) > 0$.

4. Update probability distribution over $S$:

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

where $Z_t$ is the normalization factor $Z_t = \sum_{i=1}^{m} D_t(i) \cdot e^{-\alpha_t h_t(x_i) y_i}$. Dividing by $Z_t$ gives us back a probability distribution over $S$.

Once the algorithm terminates, compute the combined hypothesis $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$. Note that $H$ is a weighted majority vote over the weak hypotheses, since:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) = sign\left(\sum_{t=1}^{T} \frac{\alpha_t}{||\alpha||_1} h_t(x)\right)$$

where $||\alpha||_1 = \sum_{t=1}^{T} \alpha_t$. Thus, it is a linear threshold function in $T$ dimensions.

At every iteration $t$, $D_t$ is a distribution over $S$. $D_t(i)$ is equal to the probability measure $P((x, y) = (x_i, y_i))$ assigned to each example $(x_i, y_i)$ of $S$.

The reasoning behind the choice of $\alpha_t$ becomes evident in the proof to follow. The intuition behind step 4 is that the booster wants to penalize the weak learner more if it misclassifies examples that it has misclassified in the past. To achieve that, on each round, it increases the weight of examples that are misclassified on that round, and decreases the weight of examples that are correctly classified. This makes the classifier better at learning how to classify correctly "hard" examples, which account for most errors.

## AdaBoost - Training Error Bound:

To complete the proof of the initial Theorem, we need to show that for the hypothesis $H$ produced by AdaBoost, $err_D(H) \leq \epsilon$. To do that, we will first prove the following bound on the training error:

**Theorem 2:** For AdaBoost, $\hat{err}(H) \leq \prod_{t=1}^{T}(2\sqrt{\epsilon_t(1-\epsilon_t)}) = \prod_{t=1}^{T}(\sqrt{1-4\gamma_t^2}) \leq e^{-2\sum_{t=1}^{T}\gamma_t^2}$

Before we prove this Theorem, let's consider its implications. Note that, if $\forall t : \gamma_t \geq \gamma$, then $e^{-2\sum_{t=1}^{T}\gamma_t^2} \leq e^{-2\gamma^2 T}$. This implies that the training error is going down exponentially fast in the number of rounds $T$. This is consistent with the behavior of the training error we saw in the examples of the slides.

Proof:

Step 1: Consider the final distribution $D_{T+1}$ defined on $S$:

$$\forall i, D_{T+1}(i) = D_1(i) \cdot \frac{e^{-a_1 h_1(x_i) y_i}}{Z_1} \cdot ... \cdot \frac{e^{-\alpha_t h_T(x_i) y_i}}{Z_T} = \frac{e^{-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)}}{m \prod_{t=1}^{T} Z_t} = \frac{e^{-y_i F(x_i)}}{m \prod_{t=1}^{T} Z_t}$$

where $F(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$. This equation results by unwrapping the recurrence that

defines $D_{t+1}$ in terms of $D_t$. Thus, we have just shown that:

$$\forall i : D_{T+1}(i) = \frac{e^{-y_i F(x_i)}}{m \prod_{t=1}^{T} Z_t}$$

.

Step 2: Consider now $\hat{err}(H)$, the training error of $H$ on $S$:

$$\hat{err}(H) = \frac{1}{m} \sum_{i=1}^{m} 1\{H(x_i) \neq y_i\} \leq \frac{1}{m} \sum_{i=1}^{m} e^{-y_i F(x_i)}$$

The inequality follows from the fact $1\{H(x_i) \neq y_i\} = 1\{y_i F(x_i) \leq 0\} \leq e^{-y_i F(x_i)}$. In particular:

$$y_i F(x_i) > 0 \implies \forall a \in \mathbb{R} : 1\{y_i F(x_i) \leq 0\} = 0 < e^a \implies 1\{y_i F(x_i) \leq 0\} < e^{-y_i F(x_i)}$$

$$y_i F(x_i) \leq 0 \implies \forall a \leq 0 : 1\{y_i F(x_i) \leq 0\} = 1 \leq e^{-a} \implies 1\{y_i F(x_i) \leq 0\} \leq e^{-y_i F(x_i)}$$

Thus, by step 1, it follows that:

$$\hat{err}(H) \leq \frac{1}{m} \sum_{i=1}^{m} \left( D_{T+1}(i) \cdot m \prod_{t=1}^{T} Z_t \right) = \left( \prod_{t=1}^{T} Z_t \right) \cdot \left( \sum_{i=1}^{m} D_{T+1}(i) \right) = \prod_{t=1}^{T} Z_t$$

The sum goes away on the third equality because we sum over the probability measures of all possible outcomes as defined by distribution $D_{T+1}(i)$ on $S$.

Step 3: All we need to do now is compute $Z_t$:

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

Now our goal is to minimize the right hand side of the equality to get the tightest upper bound possible for $\hat{err}(H)$. This minimization yields $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$, which is exactly the $\alpha_t$ term used in the AdaBoost algorithm. This yields:

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

The results of steps 2 and 3 complete the proof. ■

## AdaBoost - Generalization Bound:

Now that we have bounded the training error of AdaBoost, we can use what we have already proven in class these past weeks to prove a bound for the generalization error to complete the proof of the initial Theorem. It is payoff time!

Since all of the generalization bounds we have proven require some measure of the complexity of the hypothesis space, we need to measure the complexity of $H(x)$. To do so, we will rewrite $H(x)$ as follows:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) = g(h_1(x), ..., h_T(x))$$

where $g(z_1, ..., z_T) = sign(\sum_t \alpha_t z_t)$. Note that $g$ is a linear threshold function in $T$ dimensions. Let $\mathcal{G}$ be the class of all linear threshold functions in $T$ dimensions and let $\mathcal{H}$ be the hypothesis space of all $h_t$, $t = 1, ..., T$, since all hypotheses generated from algorithm A come from the same hypothesis space. Let $\mathcal{F}$ be the hypothesis space of $H$. Then, from the theorem proven in HW2, Problem 2, it follows that:

$$\Pi_{\mathcal{F}}(m) \leq \Pi_{\mathcal{G}}(m) \cdot (\Pi_{\mathcal{H}}(m))^T$$

Note that $VCdim(G) = T$ and let $VCdim(F) = d$. Thus, using the VC-dimension bound for the growth function we proved in a previous lecture, we get:

$$\Pi_{\mathcal{F}}(m) \leq \left(\frac{em}{T}\right)^T \cdot \left(\frac{em}{d}\right)^{dT}$$

This in turn yields:

$$err(H) \leq e\hat{r}r(H) + O\left(\sqrt{\frac{\ln(\Pi_K(m)) + \ln(1/\delta)}{m}}\right) \leq e\hat{r}r(H) + \tilde{O}\left(\sqrt{\frac{Td + \ln(1/\delta)}{m}}\right)$$

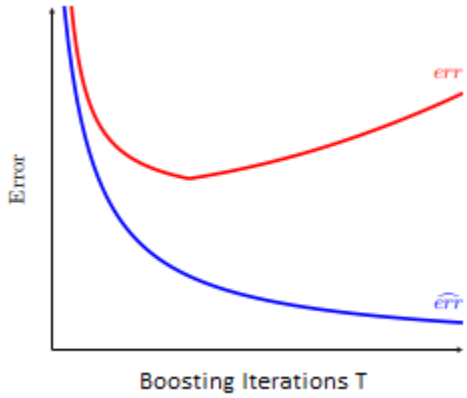
where $\tilde{O}$ stands for a soft O-bound that ignores log terms multiplied by polynomial terms.

An important thing to note is that $Td$ is a measure of the complexity of the final hypothesis $H$. This intuitively makes sense since we are combining $T$ weak hypotheses, each of which has complexity $d$, as measured by their VC-dimension.

This completes the proof of the initial Theorem. ■

## Conclusions:

The generalization bound we just proved predicts that, for not very large $T$, the generalization error decreases. However, once $T$ becomes sufficiently big, the O-term grows faster than the training error decreases, indicating that the generalization error increases. This is consistent with Ocamm's razor that states that simpler hypotheses attain lower generalization error. This behavior is exhibited graphically in the figure below.




What is very interesting though is that this is not always the case in practice. In fact, in many application of Boosting, we see the generalization error decreasing even for very large $T$. Examples of this odd behavior are included in the slides accompanying these notes.

So is Ocamm's razor flawed? Or are all these applications of Boosting misguided? Answers about these questions and the paradox seemingly contradicting Ocamm's razor are given in the next lecture.