COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Ji-Sung Kim

Lecture #11
March 11, 2019

Review. Last time we started to look at a seemingly much weaker notion of learning in which the learning algorithm only needs to do slightly better than guessing. We described the notion of a "weak learning" algorithm in comparison to our previously established notion of a "strong learning" algorithm. For comparison to the properties of a strong learning algorithm, we describe the properties of a weak learning algorithm by listing the properties of a strong learning algorithm, crossing out any non-necessary properties, and including any new properties of a weak learning algorithm in red. We say that a concept class $\mathcal{C}$ is ~~strongly~~ weakly learnable if:

- $\exists \gamma > 0$
- $\exists$ algorithm A
- $\forall c \in \mathcal{C}$, $\forall$distribution$D$
- ~~$\forall \epsilon > 0$~~
- $\forall \delta > 0$

for which the following holds: A takes $m = \text{poly}(1, /\delta)$ examples and computes h and ~~$\Pr[\text{err}_D(h) \le \epsilon] \ge 1 - \delta$~~ $\Pr[\text{err}_D(h) \le \frac{1}{2} - \gamma] \ge (1 - \delta)$. γ is sometimes called the edge.

One major question is: **are these two equivalent**? Yes! This has a very important implication for learning. We previously saw that learning appears to be an all-or-nothing phenomenon (recall the discussion of the "dichotomy" corollary of Sauer's lemma which states that either the VC-dimension of a hypothesis space $\mathcal{H}$ is finite and the bound on $\Pi_{\mathcal{H}}(m)$ is polynomial (good) with respect to m, or the VC-dimension is infinite and the bound is exponential (bad)). The answer to the previous question reinforces this idea in that we can only learn something all the way or not at all — being able to consistently achieve slightly-better-than random guessing (achieve weak learning) implies that we can also achieve strong learning.

Theorem. A concept class $\mathcal{C}$ is strongly learnable if and only if it is weakly learnable. This same principle holds with efficiency: $\mathcal{C}$ is efficiently (poly-time) strongly learnable if and only if it is efficiently weakly learnable.

1 The boosting problem

To show that strong and weak learning are equivalent, we need to find a way of converting any weak algorithm to a strong learning algorithm. We will describe a method for doing so here.

Given: m examples from a target distribution $\mathcal{D}$, $S = \langle (x_1, y_1), ..., (x_m, y_m) \rangle$, $x_i \in \mathcal{X}$, $y \in \{-1, +1\}$, and access to weak learning algorithm A (which, given "enough" random examples from any distribution D, outputs $h \in \mathcal{H}$ such that $\Pr[\text{err}_D(h) \le \frac{1}{2} - \gamma] \ge 1 - \delta$).

Problem: Find a hypothesis H which has, with high probability, arbitrarily small generalization error, $\mathrm{err}_{\mathcal{D}}(H)$.

In order to do so, we will want to combine the hypotheses $h_1, ..., h_T$ produced by weak learning algorithm A to create a final or combined hypothesis H.

The high-level idea is that we need to design an algorithm which calls a general weak learning algorithm A as a subroutine. This desired algorithm should somehow combine the collective work of multiple executions of A. We will take advantage of the fact that the weak learning algorithm A works for all distributions. We will run it on different distributions across all executions of A. This idea leads us to the AdaBoost algorithm.

1.1 AdaBoost

Algorithm 1 AdaBoost

```
for $t = 1, ..., T$ do
    (1) Run $A$ on $D_t$ to get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$
    (2) $\epsilon_t = \mathrm{err}_{D_t}(h_t) = \sum_{i:h_t(x_i)\neq y_i} D_t(i) = \frac{1}{2} - \gamma_t$
    (3) $D_{t+1}(i) = \frac{1}{Z_t} D_t(i) e^{-\alpha_t y_i h_t(x_i)}$ where $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$
end for
(4) Output $H$ where $H(x) = \mathrm{sign}\left(\sum_{i=1}^{T} \alpha_t h_t(x)\right)$.
```

Although in principle A should work for any distribution, AdaBoost only uses A for distributions defined over the training examples. Therefore, to simplify notation, we treat D_t as a distribution over the indices of the training examples $\{1, \ldots, m\}$. $D_t(i)$ is the weight under distribution D_t on the i-th example. We can think of $D_t(i)$ as the importance given to each example. How do we construct these distributions? Well, for the first distribution D_1, we will put equal importance to all examples.

On each round, after receiving h_t from A, we need to update the next distribution. Each time, we want to increase the weight on incorrectly classified examples (multiply old weight by some factor > 1), and decrease the weight on correctly classified examples (multiply old weight by some factor < 1). For this multiplicative factor, we choose $e^{-\alpha_t y_i h_t(x_i)}$. Z_t is the normalization factor (we will manipulate this a lot later on). We implicitly use the following simplification:

$$e^{-\alpha_t y_i h_t(x_i)} = \begin{cases} e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \\ e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \end{cases}$$

The choice of $e^{-\alpha_t y_i h_t(x_i)}$ as the multiplicative factor seems arbitrary but we will see in later computations that this choice leads to a bound on the training error.

The final hypothesis H is a weighted majority vote of the weak hypotheses h_t. The weights are given by α_t. In our notation, we use the sign$(\cdot)$ function which simply returns -1 if the input is negative, $+1$ if the input is positive, and 0 if it is zero.

2 Theorem on AdaBoost's training error

For AdaBoost, the training error is bounded in the following fashion:

$$\begin{aligned}
\widehat{\mathrm{err}}(H) &\leq \prod_{t=1}^{T} \sqrt{2\epsilon_t(1-\epsilon_t)} \\
&= \prod_{t} \sqrt{1-4\gamma_t^2} \qquad \text{(note: } 1+x \leq e^x\text{)} \\
&\leq \exp\left(-2\sum_{t=1}^{T} \gamma_t^2\right)
\end{aligned}$$

Small corollary. If $\forall t\ \gamma_t \geq \gamma$ (this is the weak learning assumption), then $\widehat{\mathrm{err}}(H) = e^{-2\gamma^2 T}$. This tells us that as long as each weak hypothesis is slightly better than random, the training error goes down exponentially fast.

We will now prove this in three steps!

2.1 Proof, Step 1

We would like to show that: $D_{T+1}(i) = \frac{e^{-y_i F(x_i)}}{m\prod_{t=1}^{T} Z_t}$ where $F(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$

Proof. We first reason about the update on the weight on each example $D_t(i)$. We use the fact that D_{t+1} can be directly computed by rolling out all prior recursive multiplicative updates to a single flat product.

$$\begin{aligned}
D_{t+1}(i) &= D_t(i) \cdot \frac{e^{-y_i \alpha_t h_t(x_i)}}{Z_t} \\
D_{T+1}(i) &= D_1(i)\frac{e^{-y_i \alpha_1 h_1(x_i)}}{Z_1} \cdot \ldots \cdot \frac{e^{-y_i \alpha_T h_T(x_i)}}{Z_T} \\
&= \frac{1}{m}\frac{\exp\left(-y_i \sum_t \alpha_t h_t(x_i)\right)}{\prod_t Z_t} \\
&= \frac{\exp\left(-y_i F(x_i)\right)}{m\prod_t Z_t}
\end{aligned}$$

2.2 Proof, Step 2

We would like to show that the training error $\widehat{err}(H) \leq \prod_{t=1}^{T} Z_t$.

Proof. We can rearrange the result from step 1 to get an expression for $e^{-y_i F(x_i)}$:

$$\begin{aligned}
D_{T+1}(i) &= \frac{e^{-y_i F(x_i)}}{m\prod_{t=1}^{T} Z_t} \\
e^{-y_i F(x_i)} &= D_{T+1}(i) \cdot m \prod_{t=1}^{T} Z_t
\end{aligned}$$

We observe the following $1\{H(x_i) \neq y_i\} = 1\{\text{sign}(F(x_i)) \neq y_i\} = 1\{y_i f(x_i) \leq 0\} \leq e^{-y_i F(x_i}$. The last step follows from the fact that:

- if $y_i F(x_i) \leq 0$, then $1\{y_i F(x_i) \leq 0\} = 1 \leq e^{(\text{something non-negative})}$
- else (if $y_i F(x_i) > 0$), then $1\{y_i F(x_i) \leq 0\} = 0 \leq e^{(\text{something negative})}$.

Using the above, we can reason:

$$\begin{aligned}
\widehat{err}(H) &= \frac{1}{m} \sum_{i=1}^{m} 1\{H(x_i) \neq y_i\} \\
&\leq \frac{1}{m} \sum_{i=1}^{m} e^{-y_i F(x_i)} \\
&= \frac{1}{m} \cdot \sum_{i=1}^{m} D_{T+1}(i) \cdot m \prod_t Z_t \\
&= \prod_t Z_t \quad \text{since} \ \sum_{i=1}^{m} D_{T+1}(i) = 1
\end{aligned}$$

2.3 Proof, Step 3

We would like to show that $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$.

Proof. We use the fact that the normalization factor Z_t is equal to the sum of the weights (since we want all weights to sum to 1). We split apart the terms into two groups of samples: where the hypothesis is wrong and where the hypothesis is correct.

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

We used the fact that $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$ and $(1-\epsilon_t) = \sum_{i:h_t(x_i) = y_i} D_t(i)$. This follows from (2) in the AdaBoost algorithm.

We can minimize this expression with respect to α_t (take the derivative), which yields the choice of α_t given in the algorithm. Plugging into the expression for Z_t above, we get $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$. We want to minimize Z_t because this gives us the tightest upper bound on $\widehat{\text{err}}(H)$ (see step 2). This makes training error go down as fast as possible.

3 Discussion of AdaBoost's generalization error

Our previous discussion is about the training error for AdaBoost. However, just because the algorithm achieves a low training error, this does not mean that the algorithm will generalize well on unseen data. As a result, we would like to characterize the generalization error of

AdaBoost using the general tools we previously developed in class and in the homework assignments.

Recall that H has the form $\text{sign}(\sum_t \alpha_t h_t)$ and so we can consider H as a function g of functions h_t. We can rewrite $H(x) = g(h_1(x), ..., h_T(x))$ where $g(z_1, ..., z_T) = \text{sign}\left(\sum_t \alpha_t z_t\right)$. The vectorized form of this is: $g(\mathbf{z}) = \text{sign}(\mathbf{w} \cdot \mathbf{z})$ where $\mathbf{w} = \langle \alpha_1, ..., \alpha_T \rangle$. The outer function g is a linear threshold function (LTF) in $\mathbb{R}^T$.

Now, consider the space $\mathcal{F}$ of all functions of this form $g(f_1, ..., f_T)$ where $g \in \mathcal{G}$, $h_t \in \mathcal{H}$. Let d be the VC-dimension of $\mathcal{H}$. We know that VC-dim$(\mathcal{G}) = T$ because $\mathcal{G}$ is the space of LTFs whose VC-dimension is the dimensionality (the number of dimensions) of the input domain.

We recall from HW2, that for a composition of functions of this kind:

$$\Pi_{\mathcal{F}}(m) \leq \Pi_{\mathcal{G}}(m) \prod_{t=1}^{T} \Pi_{\mathcal{H}}(m) = \Pi_{\mathcal{G}}(m) \cdot (\Pi_{\mathcal{H}}(m))^T$$

We also remember that $\Pi_{\mathcal{H}}(m) \leq \Phi_d(m) \leq \left(\frac{em}{d}\right)^d$. We can then reason that $\Pi_{\mathcal{F}}(m) \leq \Pi_{\mathcal{G}}(m) \left[\Pi_{\mathcal{H}}(m)\right]^T \leq \left(\frac{me}{T}\right)^T \left(\frac{me}{d}\right)^{dT}$

Then, using the bound using the growth function we previously saw:

$$\text{err}(H) \leq \widehat{\text{err}}(H) + \tilde{O}\left(\sqrt{\frac{Td + \ln 1/\delta}{m}}\right)$$

We remark that Td is a measure of the complexity of H which is reasonable since H is the combination of T weak hypotheses each with complexity d.