COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Max Goer

Lecture #10
March 07, 2013

1 Rademacher Complexity Bounds

Recall the following theorem from last lecture:

Theorem 1. *With probability $\geq 1-\delta$, the following two inequalities hold $\forall h \in \mathcal{H}$:*

$$err(h) \leq \hat{err}(h) + \mathfrak{R}_m(\mathcal{H}) + O\left(\sqrt{\frac{\ln(\frac{1}{\delta})}{m}}\right)$$

$$err(h) \leq \hat{err}(h) + \hat{\mathfrak{R}}_S(\mathcal{H}) + O\left(\sqrt{\frac{\ln(\frac{1}{\delta})}{m}}\right) \tag{1}$$

1.1 Bound with respect to $|\mathcal{H}|$

Theorem 2. *For $|\mathcal{H}| < \infty$:*

$$\hat{\mathfrak{R}}_S(\mathcal{H}) \leq \sqrt{\frac{2\ln(|\mathcal{H}|)}{m}}$$

Proof. We will prove this as a corollary of another theorem later in the course. □

This shows that we can bound the Rademacher complexity with respect to $|\mathcal{H}|$, which was the first measure of complexity introduced in this course. Note that if we plug this into (1) we recover the error bound from previous lectures (up to constant factors).

1.2 Bound with respect to $\Pi_{\mathcal{H}}(S)$

We would like to drop the assumption that $|\mathcal{H}| < \infty$. Note that $\hat{\mathfrak{R}}_S(\mathcal{H})$ depends on the $x_i \in S$ only. Thus we only need to see how all $h \in \mathcal{H}$ depend on S.

Theorem 3.

$$\hat{\mathfrak{R}}_S(\mathcal{H}) \leq \sqrt{\frac{2\ln(|\Pi_{\mathcal{H}}(S)|)}{m}}$$

Proof. As noted above, we only need to consider behaviors of the hypotheses on S. Let:

$$\mathcal{H}' = \{\text{one representative from } \mathcal{H} \text{ for each behavior on } S\}$$

Note that $\mathcal{H}' \subset \mathcal{H}$ and $|\mathcal{H}'| = |\Pi_{\mathcal{H}}(S)| \leq \Pi_{\mathcal{H}}(m) \leq 2^m < \infty$.

$$\begin{aligned}
\hat{\mathfrak{R}}_S(\mathcal{H}) &= \mathbb{E}_\sigma\left[\sup_{h\in\mathcal{H}}\frac{1}{m}\sum_{i=1}^{m}\sigma_i h(x_i)\right] \\
&= \mathbb{E}_\sigma\left[\sup_{h\in\mathcal{H}'}\frac{1}{m}\sum_{i=1}^{m}\sigma_i h(x_i)\right] \\
&= \hat{\mathfrak{R}}_S(\mathcal{H}') \\
&\leq \sqrt{\frac{2\ln(|\mathcal{H}'|)}{m}} \\
&= \sqrt{\frac{2\ln(|\Pi_{\mathcal{H}}(S)|)}{m}}
\end{aligned}$$

The second equality follows from the fact that for the $h \in \mathcal{H}$ that attains the first supremum, there exists $h' \in \mathcal{H}'$ that attains the same value. This implies that the supremum over $\mathcal{H}$ is no greater than the supremum over $\mathcal{H}'$. Furthermore, the supremum over $\mathcal{H}'$ is no greater than the supremum over $\mathcal{H}$, as $\mathcal{H}' \subset \mathcal{H}$. These two observations imply that the suprema are equal. The inequality on the 4^{th} line follows from theorem 2. The last equality follows by the definition of the growth function and construction of $\mathcal{H}'$. □

Hence we can bound the Rademacher complexity with respect to the growth function, the second measure of complexity introduced in this course.

1.3 Bound with respect to the VC Dimension

We can also bound the Rademacher complexity by the third complexity measure — the VC dimension.

Theorem 4. *Let $d = VCdim(\mathcal{H})$, then for $m \geq d \geq 1$:*

$$\hat{\mathfrak{R}}_S(\mathcal{H}) \leq \sqrt{\frac{2d\ln\left(\frac{em}{d}\right)}{m}}$$

Proof. This follows immediately from Sauer's lemma (see lecture #6) and theorem 3. □

2 Boosting

Boosting has its origins in PAC learning. Recall the definition of (strong) PAC learning:

Definition 1. We say that $\mathcal{C}$ is *(strongly) PAC learnable* if there exists an algorithm A such that for all $c \in \mathcal{C}$, and for all true distributions $\mathcal{D}$, for all $\epsilon > 0$, for all $\delta > 0$, A gets $m = poly(\frac{1}{\epsilon}, \frac{1}{\delta})$ examples and finds a hypothesis h_A such that:

$$\Pr\left[err(h_A) \leq \epsilon\right] \geq 1 - \delta.$$

But what happens if we can't get the error arbitrarily close to 0? Is learning all or none? To answer these questions, we introduce the notion of weak PAC learning.

Definition 2. We say that $\mathcal{C}$ is *weakly PAC learnable* if there exists an algorithm A and there exists $\gamma > 0$ such that for all $c \in \mathcal{C}$, and for all true distributions $\mathcal{D}$, for all $\delta > 0$, A gets $m = poly(\frac{1}{\delta})$ examples and finds a hypothesis h_A such that:

$$\Pr\left[err(h_A) \leq \frac{1}{2} - \gamma\right] \geq 1 - \delta.$$

Note the absence of ϵ and the presence of γ in the definition of weak PAC learning. In strong PAC learning, we need to be able to make errors arbitrarily small. In weak PAC learning we just require that the error can be brought down to some threshold. As $\gamma > 0$, we still require that we can do better than arbitrary guessing.

A natural question to ask is whether strong and weak PAC learning algorithms are equivalent. Moreover, if this is true, we would like to have an algorithm to convert a weak PAC learning algorithm into a strong PAC learning algorithm. We will see that boosting accomplishes this.

Definition 3. A *boosting algorithm* is an algorithm that converts a weak learning algorithm into a strong learning algorithm.

It is important to note that both strong and weak PAC learning are distribution-free. The following example will shed more light on the importance of this.

2.1 An example for learning with a fixed distribution

Let $\mathcal{C}$ be the set of *all* concepts over $\{0,1\}^n \cup \{z\}$, where $\{z\} \notin \{0,1\}^n$. Let D be the distribution that assigns mass $\frac{1}{4}$ to the point z and has mass $\frac{3}{4}$ uniformly distributed over $\{0,1\}^n$. That is:

$$Pr_{x\sim D}[x = k] = \begin{cases} \frac{1}{4} & \text{if } k = z \\ \frac{3}{4 \cdot 2^n} & \text{if } k \in \{0,1\}^n \end{cases}$$

Consider the hypothesis that predicts $c(z)$ if $x = z$ and simply flips a coin otherwise. Eventually we will get a sample of z and thus learn $c(z)$ (we can identify such an example when $x_i \notin \{0,1\}^n$). This hypothesis will always correctly predict all points $x_i = z$ and predict with 50% accuracy otherwise. Thus its error is:

$$err_D(h_a) = \frac{3}{4} \cdot \frac{1}{2} = \frac{3}{8} < \frac{1}{2}$$

Hence, (if we drop the distribution-freeness from the definition) $\mathcal{C}$ is weakly PAC learnable for the fixed distribution D. However, $VCdim(\mathcal{C}) = 2^n$ hence by theorem 1 of lecture #7 $\mathcal{C}$ is <u>not</u> strongly PAC learnable (again modifying the definition to a fixed distribution) using any algorithm. This is because we would need at least $m = \Omega(2^n)$ examples which is not polynomial.

Hence we cannot necessarily convert a weak into a strong learning algorithm if we fix the distribution.

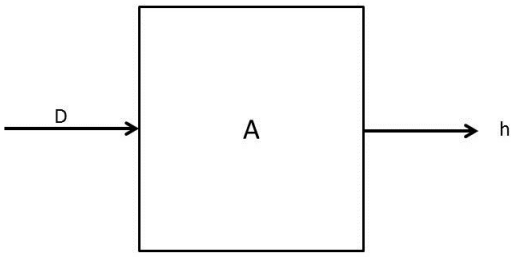
2.2 The setup

We are given:

- $S = \{(x_1, y_1), ..., (x_m, y_m)\}$ drawn from the true distribution $\mathcal{D}$, where $x_i \in X$, $y_i \in \{-1, 1\}$.
- access to a weak learner A which:
  $\forall D$ (not necessarily the same as $\mathcal{D}$)
  given examples drawn from D
  computes $h \in \mathcal{H}$ (the hypothesis space of the weak learner) such that:

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

The following diagram illustrates what the weak learner does.



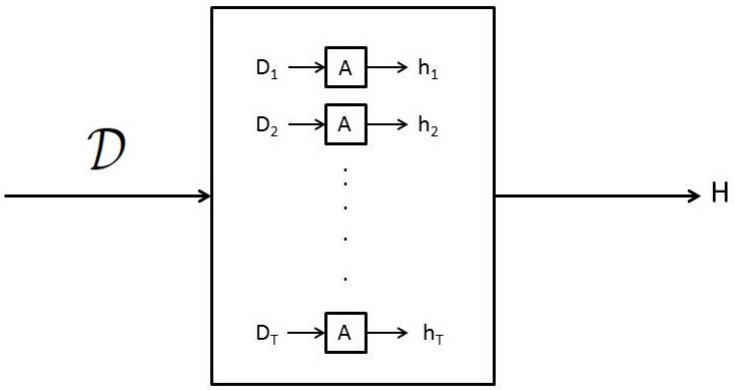
Our goal is to find a final hypothesis H – we don't require that $H \in \mathcal{H}$ – such that:

$$\Pr[err_{\mathcal{D}}(H) \leq \epsilon] \geq 1 - \delta$$

Note that we use the true distribution, $\mathcal{D}$ and not D for the last probability.

2.3 The main idea

The main idea behind boosting is to run the weak learning algorithm several times and combine the hypotheses from each run. To do this effectively, we need to force the weak algorithm to learn by giving it a different D on every run. The following diagram illustrates this:



2.4 The AdaBoost algorithm

We will now analyze the AdaBoost algorithm. The pseudocode is given below.

Algorithm 1 AdaBoost

procedure AdaBoost(S, T)
  $D_1(i) = \frac{1}{m}\ \forall i$
  for $t = 1, ..., T$ **do**
    Construct D_t
    Run A on D_t (sample from D_t)
    Get h_t from A
    $\epsilon_t = err_{D_t}(h_t) = \frac{1}{2} - \gamma_t$
    Choose $\alpha_t > 0$
    $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \\ e^{-\alpha_t} & \text{else} \end{cases}$
  Output $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

In the above algorithm, $D_t(i) =$ weight on (x_i, y_i) under D_t. We can think of this as a distribution of weight or importance of the $x_i \in S$. Z_t is simply a normalizing factor to make D_{t+1} a probability distribution. Note that the update places more weight on previously misclassified examples and less weight on previously correctly classified examples. α_t is unspecified in the code above. We will determine it later.

AdaBoost relies on the weak learning assumption:

$$\gamma_t \geq \gamma > 0$$

2.5 Example 2

See slides on course website.

2.6 Bound on the empirical error of AdaBoost

Theorem 5. *The final hypothesis H output by Adaboost satisfies the following:*

$$\begin{aligned} \hat{err}(H) &\leq \prod_{t=1}^{T} \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] \\ &= \exp\left(-\sum_{t=1}^{T} RE\left(\frac{1}{2}||\epsilon_t\right)\right) \\ &= \prod_{t=1}^{T} \left[\sqrt{1-4\gamma_t^2}\right] \\ &\leq \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right) \\ &\quad \textit{(Additionally, if the weak learning assumption holds)} \\ &\leq e^{-2\gamma^2 T} \end{aligned}$$

Proof. The second line holds by definition. The third line holds as $\epsilon_t = \frac{1}{2} - \gamma_t$. The fourth line follows from the fact that $1 + x \leq e^x$. The final line follows by the weak learning assumption. Thus it is sufficient to show that the first line holds, which follows from lemma 2 and 3 (stated later). □

Lemma 1.

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

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

Proof. Note $D_{t+1}(i) = \frac{D_t(i)}{Z_t} e^{-\alpha_t y_i h_t(x_i)}$ by definition.

We can now solve for D_{T+1} recursively.

$$\begin{aligned} D_{T+1}(i) &= D_1(i) \cdot \frac{\exp\left(-\alpha_1 y_i h_1(x_i)\right)}{Z_1} \cdot \frac{\exp\left(-\alpha_2 y_i h_2(x_i)\right)}{Z_2} \cdots \frac{\exp\left(-\alpha_T y_i h_T(x_i)\right)}{Z_T} \\ &= \frac{1}{m} \frac{\exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)\right)}{\prod_{t=1}^{T} Z_t} \\ &= \frac{\exp\left[-y_i F(x_i)\right]}{m \prod_{t=1}^{T} Z_t} \end{aligned}$$

□

Lemma 2. $\hat{err}(H) \leq \prod_{t=1}^{T} Z_t$

Proof.

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

Line 3 follows from the fact that $e^{-y_i F(x_i)} > 0$ if $y_i F(x_i) > 0$ and $e^{-y_i F(x_i)} \geq 1$ if $y_i F(x_i) \leq 0$. Line 4 follows from lemma 1. The last line follows from the fact that D_{T+1} is a probability distribution. □

Lemma 3. $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$

Proof.

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

The last equality follows because $\sum_{i:y_i \neq h_t(x_i)} D_t(i) = \Pr[err_{D_t}(h_t)] = \epsilon_t$.

We choose an α_t so that the empirical error is minimized. By lemma 2, this corresponds to minimizing Z_t. This yields:

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

This is also the α_t we use in the algorithm.

Plugging this into equation (2) we get the desired result. □