EECS 598-005: Theoretical Foundations of Machine Learning **Fall 2015**

Lecture 14: Growth Function Generalization Bound and Online Learning

Lecturer: Jacob Abernethy *Scribes: Shun Zhang*

14.1 Growth Function Generalization Bound

Recall Massart's Lemma from the last lecture.

Lemma 14.1. *(Massart's Lemma) Let A be finite. $A \subseteq \mathbb{R}^n$. $\max_{\mathbf{x}\in A} ||\mathbf{x}||_2 = r$. Then*

$$\mathbb{E}[\sup_{\mathbf{x}\in A} \sum_{i=1}^{m} \sigma_i x_i] \leq \sqrt{2\log|A|}r$$

where σ_i's are Rademacher variables.

Corollary 14.2. *Let G be a function class taking values in $\{-1, 1\}$.*

$$\mathfrak{R}_m(G) \leq \sqrt{\frac{2\log \Pi_G(m)}{m}}$$

Recall that

$$\Pi_G(m) = \max_{S\subseteq X, |S|=m} |\{(h(x_1), h(x_2), \cdots, h(x_m)) : h \in G\}|$$

where $S = (x_1, x_2, \cdots, x_m)$.

Proof: By the definition of Rademacher complexity,

$$\mathfrak{R}_m(G) = \mathbb{E}_{S\sim D^m}\Big[\mathbb{E}_\sigma[\sup_{g\in G|_S} \frac{1}{m}\sum_{i=1}^{m} g(x_i)\sigma_i]\Big]$$

where σ is a Rademacher vector.

Let $A_S = \{(g(x_1), \cdots, g(x_m)) : g \in G\}$. Define the L-2 norm of a set as $||X||_2 = \max_{x\in X} ||x||_2$. We notice that $||A_S||_2 = \sqrt{m}$. Therefore,

$$\begin{aligned}
\mathfrak{R}_m(G) &= \mathbb{E}_{S\sim D^m}\mathbb{E}_\sigma[\sup_{z\in A_S} \frac{1}{m}\sum_{i=1}^{m} \sigma_i z_i] \\
&\leq \frac{1}{m}\mathbb{E}_S\Big[\sqrt{m}\sqrt{2\log|A_S|}\Big] && \text{(Massart's Lemma)} \\
&\leq \frac{1}{\sqrt{m}}\sqrt{2\log \Pi_G(m)}
\end{aligned}$$

■

Now we can bound $R(h_S^{ERM}) - R(h^*)$ using Massart's Lemma and the corollary above.

Theorem 14.3. *(Big Theorem) Let $D \in \Delta(X \times \{-1, 1\})$. H is a binary function class with VC-dimension of d. G is a loss class such that $G = \{g(x, y) = 1_{h(x)\neq y} : h \in H\}$. For a sample $S \sim D^m$, h_S^{ERM} minimizes $\hat{R}_S$. With probability of $1 - \delta$,*

$$R(h_S^{ERM}) - R(h^*) \leq \sqrt{\frac{8d\log m}{m}} + \sqrt{\frac{2\log\frac{1}{\sigma}}{m}}$$

Proof: Let $h^* = \arg\min_{h \in H} R(h)$.

$$\begin{aligned}
R(h_S^{ERM}) - R(h^*) &\leq R(h_S^{ERM}) - \hat{R}_S(h_S^{ERM}) + \hat{R}_S(h^*) - R(h^*) && (\hat{R}_S(h_S^{ERM}) \leq \hat{R}_S(h^*)) \\
&\leq 2 \sup_{h \in H} |R(h) - \hat{R}_S(h)| && \text{(Uniform Deviation Bound)} \\
&\leq 4\mathfrak{R}_m(G) + \sqrt{\frac{2\log\frac{1}{\delta}}{m}} && \text{(Symmetrization trick)} \\
&= 2\mathfrak{R}_m(H) + \sqrt{\frac{2\log\frac{1}{\delta}}{m}} && (\mathfrak{R}_m(G) = \tfrac{1}{2}\mathfrak{R}_m(H)) \\
&\leq 2\sqrt{\frac{2\log \Pi_G(m)}{m}} + \sqrt{\frac{2\log\frac{1}{\delta}}{m}} && \text{(Massart's Lemma and Corollary)} \\
&\leq \sqrt{\frac{8d\log m}{m}} + \sqrt{\frac{2\log\frac{1}{\delta}}{m}} && \text{(Sauer's Lemma)}
\end{aligned}$$

■

Recall that the error rate is $\frac{d}{m}$ for non-noisy setting. Here we have $\sqrt{\frac{d\log m}{m}}$ for noisy setting. This is a nice bound since the right-hand-side only grows when VC-dimension grows.

14.2 Online Learning

Online learning assumes that the learning algorithm learns and tests by observing a sequence of data in real time. This is a more realistic setting than having a batch of data and training on them. Popular settings in online learning include regret minimization, no regret setting and expert's setting. Some key facts in online learning are:

- You do not need i.i.d. any more.
- Bounds are often the same.

14.2.1 "Noise-Free" Learning from Experts

We assume that there is a binary outcome $y_t \in \{0, 1\}$ on each round. There are N experts. Expert i on round t predicts $f_{i,t} \in \{0, 1\}$. Our algorithm interacts with the environment in the following way.

Algorithm 1 Learning form Experts

```
#mistakes ← 0
for t = 1, ··· , T do
    Algorithm observes predictions of experts f_{1,t}, f_{2,t}, ··· , f_{n,t}
    Algorithm outputs a guess ŷ_t ∈ {0, 1}
    Nature reveals y_t
    if y_t ≠ ŷ_t then
        #mistakes ← #mistakes + 1
    end if
end for
```

We can make the following claim on the number of mistakes ($\#mistakes$) an algorithm can make.

Claim 14.4. *There exists an algorithm such that as long as there is a "perfect expert", i.e.,* $\exists_i f_{i,t} = y_t \forall_t$*, the perfect algorithm makes number of mistakes fewer than* $\log_2 N$*.*

Proof: We construct an algorithm as follows.

First, we define "good experts" at time step t as $c_t = \{i : f_{i,t'} = y_{t'}, \forall_{t'=1,\cdots,t-1}\}$. In each round, the algorithm makes a prediction using the majority votes of good experts.

Notice that if the algorithm predicts $\hat{y}_t \neq y_t$, then at least half of good experts are ruled out. $\frac{|c_{t+1}|}{|c_t|} \leq \frac{1}{2}$. Therefore,

$$1 \leq |c_t| \leq (\frac{1}{2})^{\#mistakes}|c_1| = (\frac{1}{2})^{\#mistakes}N$$

Take the log of the inequality. We have

$$\#mistakes \leq \log_2 N$$

■

Note that this is analogous to PAC learning, where we have error rate of $\frac{\log |H|}{m}$.

14.2.2 Gambling Example

Imagine N teams on week t, Team i plays Team j for some pair $i, j \in [N]$. Assume there exists a perfect permutation of $[N]$, denoted by Π, such that i beats j if and only if $\Pi(i) > \Pi(j)$.

We need an algorithm that "wins money from gambling." There are $N!$ possible permutations. From Claim 14.4, we know that there exists an algorithm that makes number of mistakes fewer than $\log_2 N! \leq N \log N$. This means that we only need $\log N$ samples per team to make a good prediction. But still we lack efficient ways to realize such algorithm.