COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Xinyi Fan

Lecture 15
April 02, 2013

1 Connection with PAC learning

1.1 Brief review of last lecture

Last time we started talking about one particular form of the online learning model - learning with experts' advice, and looked at a simple case, where we assumed that a "perfect" expert who never makes mistake exists. The halving algorithm suggests that we simply need to keep track of experts who haven't made any mistakes so far and take majority vote among their predictions. The number of mistakes that the algorithm makes is bounded by $\lg N$, where N is the total number of experts.

1.2 An example

We start with a concrete example which relates the halving algorithm to the PAC-learning setting discussed earlier. Suppose the unknown target concept c is chosen by an adversary from some known finite hypothesis space $\mathcal{H} = \{h_1, ..., h_N\}$. The experts are hypotheses from the same finite hypothesis space and there exists one "perfect" expert. We try to use the halving algorithm to learn c.

- target function $c \in \mathcal{H} = \{h_1, ..., h_N\}$ (chosen by an adversary)
- on each round
  - the learner observes an example x (chosen by an adversary)
  - $\forall i$: expert i predicts on x: $\xi_i = h_i(x)$
  - using the halving algorithm the learner predicts $\hat{y}$
  - the learner observes $y = c(x)$

If we apply the halving algorithm to this setting, since we know that there exists a "perfect" expert, according to the argument we made in last lecture, the number of mistakes made here is at most $\lg |\mathcal{H}|$, where $|\mathcal{H}|$ is the number of hypotheses. Note that $\lg |\mathcal{H}|$ is also a complexity measure of the hypothesis space.

1.3 Bounding the number of mistakes by complexity measure of $\mathcal{H}$

The halving algorithm is actually not the best possible algorithm for the setting above. We assume A to be some deterministic algorithm and define the following notations. In the worst case, the number of mistakes made by A is denoted by

$$M_A(\mathcal{H}) = \max_{\textbf{adversary } (c, x\textbf{'s})} (\#\ \textbf{mistakes made by } A) \tag{1}$$

where the maximization is over the adversary choices of c and x's. And we formulate the optimization as

$$opt(\mathcal{H}) = \min_A M_A(\mathcal{H}) \tag{2}$$

It turns out that the optimal number of mistakes made by any deterministic algorithm A is bounded between the *VC-dimention* and $\lg|\mathcal{H}|$. Combined with what we obtained in Section 1.2, a theorem is given below.

Theorem 1.

$$VCdim(\mathcal{H}) \leq opt(\mathcal{H}) \leq M_{halving}(\mathcal{H}) \leq \lg|\mathcal{H}| \tag{3}$$

Proof. We only need to prove $VCdim(\mathcal{H}) \leq opt(\mathcal{H})$. Pick an algorithm A and fix it, and let $d = VCdim(\mathcal{H})$. We need to show that it's possible for A to make d mistakes. According to the definition of VC-dimension, let $x_1, ..., x_d$ be d examples shattered by $\mathcal{H}$. We'll construct an adversary to choose $x_1, ..., x_d$ and force A to make mistake on each of them.

- for $t = 1, ..., d$
  - the algorithm observes x_t
  - the algorithm predicts as $\hat{y}_t$
  - choose $y_t = c(x_t) \neq \hat{y}_t$

In fact, there has to exist a target concept $c \in \mathcal{H}$, which disagrees with all the d predictions of the algorithm, because the d examples are shattered by $\mathcal{H}$. Besides, since the algorithm A is deterministic, it can be simulated ahead of time, which then makes the adversary construction possible. Therefore, it's possible for A to make d mistakes. In particular, it is possible for the adversary to choose c ahead of time, before the presentation of $x_1, ..., x_d$, as required by the model. □

2 Weighted majority algorithm

Now consider a more realistic situation. Let's assume that, instead of a "perfect" expert, there exists a "pretty good" expert in our setting, who might make mistakes, but is still good. We hope to modify the halving algorithm, and make it still work well in the new setting. The strategy here is to maintain a weight for each expert, and when an expert makes a mistake, down-weigh its advice, instead of discarding it. This leads to the weighted majority algorithm, which is described as Algorithm 1. Notice that if we choose β to be 0, the weighted majority algorithm degenerates to the halving algorithm.

Now we try to use the best expert to bound the number of mistakes made by the learner according to the weighted majority algorithm.

Theorem 2.

$$(\#mistakes\ of\ learner) \leq a_\beta\,(\#mistakes\ of\ best\ expert) + c_\beta \lg N \tag{4}$$

where $a_\beta = \dfrac{\lg(1/\beta)}{\lg\left(\dfrac{2}{1+\beta}\right)}$, $c_\beta = \dfrac{1}{\lg\left(\dfrac{2}{1+\beta}\right)}$

To get some intuition for the constants, let's look at some of their sample values, as given in Table 1. The settings of constants are reasonable, and by examining the extreme cases we see that there's a trade off between a_β and c_β.

Algorithm 1 Weighted Majority Algorithm

choose down-weigh parameter $\beta \in [0, 1)$
denote the i-th expert's weight at the t-th round by $w_{t,i}$
$\forall i$: $w_{1,i} = 1$
on each round t

$\forall i$:

get the i-th expert's prediction $\xi_{t,i}$

$q_{t,0} = \sum_{i:\xi_{t,i}=0} w_{t,i}, \quad q_{t,1} = \sum_{i:\xi_{t,i}=1} w_{t,i}$

the prediction at this round is $\hat{y}_t = \begin{cases} 1 & if \quad q_{t,1} > q_{t,0} \\ 0 & else \end{cases}$

$\forall i$: update the weight

if $\xi_{t,i} \neq y_t$ then

$w_{t+1,i} \leftarrow \beta w_{t,i}$

else

$w_{t+1,i} \leftarrow w_{t,i}$

Table 1: Sample values of constants

β	a_β	c_β
0.5	≈ 2.4	≈ 2.4
$\to 0$	∞	1
$\to 1$	2	∞

We can also divide both sides of (4) by the number of rounds T and get

$$\frac{(\#mistakes\ of\ learner)}{T} \leq \frac{a_\beta\,(\#mistakes\ of\ best\ expert)}{T} + \frac{c_\beta \lg N}{T} \tag{5}$$

When T gets large, the term $\frac{c_\beta \lg N}{T} \to 0$, and the bound can be explained as the rate at which the learner is making mistakes is bounded by the rate at which the best expert is making mistakes multiplied by some constant.

To prove the theorem, we adopt a method which is quite similar to the proof of the halving algorithm.

Proof. We keep track of the sum of the weights of all of the N experts $W_t = \sum_{i=1}^{N} w_{t,i}$ on each round t. Initially we set $W_1 = N$, and on each round t we assume $y_t = 0$.

$$\begin{aligned} W_{t+1} &= \sum_{i:\xi_{t,i}=1} w_{t+1,i} + \sum_{i:\xi_{t,i}=0} w_{t+1,i} \\ &= \sum_{i:\xi_{t,i}=1} w_{t,i}\beta + \sum_{i:\xi_{t,i}=0} w_{t,i} \\ &= q_{t,1}\beta + q_{t,0} \\ &= q_{t,1}\beta + (W_t - q_{t,1}) \\ &= W_t - (1-\beta)q_{t,1} \end{aligned}$$

If the learner makes a mistake ($\hat{y}_t \neq y_t$),

$$q_{t,1} \geq q_{t,0}$$

$$\Rightarrow \quad q_{t,1} \geq \frac{1}{2} W_t$$

$$\Rightarrow \quad W_{t+1} \leq W_t - (1-\beta) \cdot \frac{1}{2} W_t = \frac{1+\beta}{2} W_t$$

If the learner makes a mistake, the sum of the weights will decrease by being multiplied by a factor $\frac{1+\beta}{2}$. Therefore, after m mistakes, the sum of the weights of all the experts is

$$W_{new} \leq \left(\frac{1+\beta}{2}\right)^m W_1 \tag{6}$$

That's how we get the upper bound. To get a lower bound on W, we define L_i to be the number of mistakes of expert i. Then the weight of any individual expert i after making L_i mistakes becomes $w_i = \beta^{L_i}$, because w_i is set to be 1 initially. Therefore, we have

$$\beta^{L_i} = w_i \leq W \leq \left(\frac{1+\beta}{2}\right)^m \cdot N \tag{7}$$

where N is the number of experts. We can solve for the number of mistakes m as

$$m \leq \frac{L_i \lg\left(\frac{1}{\beta}\right) + \lg N}{\lg \frac{2}{1+\beta}} \tag{8}$$

□

3 Randomized weighted majority algorithm

For any deterministic algorithm, the constant a_β is close to 2 when β approaches 1. In fact, a_β cannot get smaller than 2 for deterministic algorithms. We hope a_β can be even smaller and close to 1, in which case, the learner can roughly do as well as the best expert. To get this result, we need to modify our algorithm to make predictions randomly. In fact, adding some randomness to an algorithm's behavior is often necessary when dealing with an adversary.

To get the new algorithm, we only need to change the hard threshold at the learner's prediction in the weighted majority algorithm to

$$\hat{y}_t = \begin{cases} 1 & \textit{with prob. } q_{t,1}/W_t \\ 0 & \textit{with prob. } q_{t,0}/W_t \end{cases}$$

The new algorithm is called randomized weighted majority algorithm. Similarly, we can obtain a bound of the same form on the expected number of mistakes made by the learner rather than the number of mistakes made by the learner, due to the introduction of randomization during the learner's prediction. Note that examples are still chosen by an adversary, and the expectation is only over the randomization of the algorithm.

Theorem 3.

$$E\left[(\#mistakes\ of\ learner)\right] \leq a_\beta\,(\#mistakes\ of\ best\ expert) + c_\beta \lg N \tag{9}$$

where $a_\beta = \dfrac{\ln(1/\beta)}{1-\beta}$, $c_\beta = \dfrac{1}{1-\beta}$

With the new definition of constants a_β can go to 1 as closely as possible when β approaches 1. The proof of this is just a modification of the last proof.

Proof. We again keep track of the sum of the weights of all of the N experts $W_t = \sum_{i=1}^{N} w_{t,i}$ on each round t. Initially, let $W_1 = N$, and on each round t, the probability (over the algorithm's randomization) that the algorithm makes an mistake is

$$\begin{aligned}
\ell_t &= Pr\left[\hat{y}_t \neq y_t\right] = \begin{cases} q_{t,1}/W_t & if\ y_t = 0 \\ q_{t,0}/W_t & if\ y_t = 1 \end{cases} \\
&= \sum_{i:\xi_{t,i}\neq y_t} w_{t,i}/W_t
\end{aligned}$$

Therefore, we have the new sum of weights

$$\begin{aligned}
W_{t+1} &= \sum_{i:\xi_{t,i}\neq y_t} w_{t+1,i} + \sum_{i:\xi_{t,i}=y_t} w_{t+1,i} \\
&= \sum_{i:\xi_{t,i}\neq y_t} w_{t,i}\beta + \sum_{i:\xi_{t,i}=y_t} w_{t,i} \\
&= \ell_t W_t \beta + (W_t - \ell_t W_t) \qquad \left(W_t = \sum_{i:\xi_{t,i}\neq y_t} w_{t,i} + \sum_{i:\xi_{t,i}=y_t} w_{t,i},\quad \sum_{i:\xi_{t,i}\neq y_t} w_{t,i} = \ell_t W_t\right) \\
&= W_t\left(1 - \ell_t(1-\beta)\right)
\end{aligned}$$

After making m mistakes,

$$\begin{aligned}
\beta^{L_i} &= w_i \\
&\leq W_{final} \\
&\leq N\prod_{t=1}^{T}\left(1-\ell_t(1-\beta)\right) && (W_1 = N) \\
&\leq N\prod_{t=1}^{T}\exp(-\ell_t(1-\beta)) && (1+x \leq e^x) \\
&= N\cdot\exp\left(-(1-\beta)\sum_{t=1}^{T}\ell_t\right) && (10)
\end{aligned}$$

$L_A = \sum_{t=1}^{T}\ell_t$ is actually the expected number of mistakes. Therefore, after doing some algebra we have

$$L_A \leq \frac{L_i\cdot\ln\dfrac{1}{\beta} + \ln N}{1-\beta} \tag{11}$$

□

4 More discussion

Now let's look at how to choose the factor β. Say we know the loss of the best expert

$$\min_i L_i \leq K$$

β can be chosen as $\dfrac{1}{1+\sqrt{\dfrac{2\ln N}{K}}}$. Plug it into the bound (11), we have

$$L_A \leq \min_i L_i + \sqrt{2K \ln N} + \ln N \tag{12}$$

To visualize the way in which a learner predicts, let's look at Figure 1. Here the x-axis represents the weighted fraction of experts predicting 1 and the y-axis represents the probability that the algorithm predicting $\hat{y}$ as 1. The green curve shows how weighted majority algorithm predicts and the red curve shows how randomized weighted majority algorithm predicts. The blue curve represents many other ways in between, and we can tune it properly to get different algorithms with different constants.

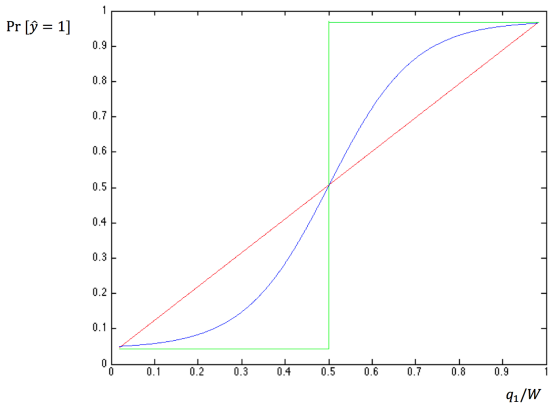

Figure 1: Comparison among predictions of different algorithms

By choosing an appropriate algorithm, which can be represented by some curve like the blue one in the figure, we can get the bound

$$L_A \leq \frac{L_i \ln(1/\beta) + \ln N}{2 \ln \dfrac{2}{1+\beta}} \tag{13}$$

And if we know that $\min_i L_i \leq K$, we can design β to obtain

$$L_A \leq \min_i L_i + \sqrt{K \ln N} + \frac{\lg N}{2} \tag{14}$$

If we set $K = 0$, which implies the existence of a "perfect" expert, the bound becomes

$$L_A \leq \frac{\lg N}{2} \tag{15}$$

This result is better than the bound of the halving algorithm.

Divide both sides of the bound (14) by the total number of rounds T,

$$\frac{L_A}{T} \leq \frac{\min_i L_i}{T} + \frac{\sqrt{K \ln N}}{T} + \frac{\lg N}{2T} \tag{16}$$

Usually we can assume $K \leq T/2$, which means that the best expert can make at most $T/2$ mistakes. This assumption is reasonable because we could take two experts who always predict as the opposite to each other, and thus there has to exist an expert between the two who makes at most $T/2$ mistakes. Plug $K = T/2$ into Equation 16, we have

$$\frac{L_A}{T} \leq \frac{\min_i L_i}{T} + \sqrt{\frac{\ln N}{2T}} + \frac{\lg N}{2T} \tag{17}$$

When T gets large, the last two terms on the right hand side is going to zero, with different rates. Since we plug $K = T/2$ into this, these different rates of convergence will depend on K, which represents the loss of the best expert.

Finally, we will show that introducing randomness to the experts' prediction and the examples will in fact not improve our result, by proving a lower bound on what is possible for any algorithm in this setting.

Assume that the experts' predictions are random, so as the labels. More specifically, the labels and experts' predictions are chosen to be 0 or 1 with equal probability. On each round, independently we have

$$\xi_{t,i} = \begin{cases} 0 & \textit{with prob. } 1/2 \\ 1 & \textit{with prob. } 1/2 \end{cases}$$

$$y_t = \begin{cases} 0 & \textit{with prob. } 1/2 \\ 1 & \textit{with prob. } 1/2 \end{cases}$$

Then for any learning algorithm, the expected number of mistakes (over randomness of experts' predictions and data) is

$$E[L_A] = T/2 \tag{18}$$

And for any expert i, we also have

$$E[L_i] = T/2 \tag{19}$$

However, if we look at the expected loss of the *best* expert, it turns out that

$$E\left[\min_i L_i\right] \approx \frac{T}{2} - \sqrt{\frac{T \ln N}{2}} \tag{20}$$

and therefore,

$$E[L_A] \gtrsim E\left[\min_i L_i\right] + \sqrt{\frac{T \ln N}{2}} \tag{21}$$

Compare (17), the upper bound, and (21), the approximate lower bound, since $1/\sqrt{T}$ dominates $1/T$, we find that the two bounds are quite close to each other and even up to the constants. Therefore, the adversary setting actually does as well as the randomized data setting.