COS 511: Foundations of Machine Learning

Rob Schapire
Scribe: Joseph Bradley

Lecture # 13
March 28, 2006

So far, we have been studying learning algorithms which:

1 Get a batch of training examples

2 Choose a hypothesis, which is then fixed

3 Get a batch of test examples

In the models and algorithms we have studied, we have assumed that all examples were randomly generated from a fixed target distribution.

1 Online Learning Model

We now turn to a new learning model in which learning algorithms get 1 example at a time, make a prediction on that example's label, find out if the prediction was right or wrong, and repeat. For example, we might want to predict each day if the stock market will go up or down, or if it will rain or not. The goal of the learning algorithm is to minimize the number of mistakes it makes in these predictions. There are several points which distinguish this model from previous models (e.g. the PAC model):

- We train and test the algorithm at the same time.
- Online learning algorithms are often very simple.
- We no longer assume that the data is randomly generated from a fixed distribution. Rather, the data is entirely arbitrary.

1.1 Example of Online Learning

Say we want to predict if the S&P 500 will go up or down each day. We could look at the predictions made by some investing experts and try to base our prediction on the predictions of the experts. In this example, there are 4 such experts, each of which makes a prediction for the current day. The *learner* (or *master algorithm*) then makes its own prediction. Finally, the experts and learner get to see the outcome and find out if their predictions were correct.

	Experts					
	1	2	3	4	Learner	Outcome
day 1	$\uparrow$	$\uparrow$	$\downarrow$	$\uparrow$	$\uparrow$	$\uparrow$
day 2	$\downarrow$	$\uparrow$	$\uparrow$	$\downarrow$	$\downarrow$	$\uparrow$
$\vdots$					$\vdots$	
total # mistakes	37	12	62	51	17	

You can see that, for example, expert 1 made a correct prediction on day 1 but made a mistake on day 2. After some number of days, we get the above totals for the number of mistakes made by the experts and the learner. The totals for the experts give us an idea of how well the learner can do. Ideally, we would like the number of mistakes made by the learner to be close to that of the best expert in hindsight at the end of this sequence of trials. This particular setup is called *learning with expert advice.*

Note that, since we do not make statistical assumptions, our choice of experts is not limited. They could be people, some fixed set of functions, a computer model with various parameter settings, a set of learning algorithms, etc.

1.2 Learning with Expert Advice

To restate this setup for online learning a bit more formally,

```
$N$ = number of experts

For $t = 1, 2, \ldots, T$ trials

    Each expert $i$ makes prediction $\xi_i \in \{0, 1\}$
    Learner makes prediction $\hat{y} \in \{0, 1\}$
    Experts and learner observe outcome $y \in \{0, 1\}$
    (Learner makes a mistake if $y \neq \hat{y}$)

Goal: (# mistakes by learner) $\leq \min_i$ (# mistakes by expert $i$) + (small amount)
```

2 Models for Learning with Expert Advice

First, we will look at a simple model in which we can assume that at least one expert is perfect and never makes mistakes. As stated above, we want to guarantee that our learning algorithm will make at most a small number of mistakes.

One possible learning algorithm in this setting is as follows:

```
Maintain a list of experts which have not yet made mistakes, initially including all experts.

On each trial,

    Choose an arbitrary expert $i$ from the list, and use its prediction as the learner's prediction: $\hat{y} = \xi_i$
    Eliminate experts $i$ in list which made mistakes ($\xi_i \neq y$).
```

Every time the learner makes a mistake, the expert it chose to follow will be thrown out since that expert must have made a mistake as well. Since there are N experts, with at least one perfect one, the algorithm will make $\leq N - 1$ mistakes.

It is possible, however, to do better than this with the Halving Algorithm.

2.1 Halving Algorithm

Algorithm:

> Maintain a list of experts which have not yet made mistakes, initially including all experts.
>
> On each trial,
>
> > Make prediction based on majority vote of experts in list:
> >
> > $$\hat{y} = 1 \,\text{if}\, |\{i : \xi_i = 1\}| > |\{i : \xi_i = 0\}|, \,\text{else}\, 0$$
> >
> > Eliminate experts i in list which made mistakes ($\xi_i \neq y$).

To analyze the Halving Algorithm, let $W = \#$ of surviving experts (experts left in list). Initially, $W = N$. If the learner makes a mistake, then $\geq 1/2$ of the remaining experts are eliminated since the learner makes a prediction based on a majority vote of the experts. So,

$$\begin{array}{llll} \text{After} & 1 & \text{mistake,} & W \leq \frac{1}{2}N \\ \text{After} & 2 & \text{mistakes,} & W \leq \frac{1}{4}N \\ & & \vdots & \\ \text{After} & m & \text{mistakes,} & W \leq 2^{-m}N \end{array}$$

We know that one expert is perfect and will never be thrown out, so $W \geq 1$. Therefore, we have $1 \leq W \leq 2^{-m}N$, which implies the learner makes $m \leq \lg N$ mistakes.

2.2 Relating These Ideas to PAC Model

We now relate the ideas we have covered in online learning with expert advice to ideas we have seen previously in the PAC learning model:

- Suppose that each expert is associated with a hypothesis in the space $\mathscr{H} = \{h_1, h_2, \ldots, h_N\}$.
- Since one expert is perfect, we know the target concept c is in $\mathscr{H}$.
- On each round,
  - We get an example x.
  - We have predictions from experts/hypotheses: $\xi_i = h_i(x)$
  - We make a prediction $\hat{y}$.
  - We observe the outcome $y = c(x)$.

In this special case of the learning with expert advice model, we have seen that the number of mistakes made by the Halving Algorithm is $\leq \lg |\mathscr{H}|$. The term $\lg |\mathscr{H}|$ is the same measure of hypothesis space complexity that we have seen in the PAC model! Now, let A be a deterministic online learning algorithm, and let $M_A(\mathscr{H}) = \max_{\text{adversary}}$ (# mistakes made by A when learning $c \in \mathscr{H}$). With these definitions, $M_{\text{halving}}(\mathscr{H}) \leq \lg |\mathscr{H}|$.

So, a natural question to ask is: Is there a better algorithm for this setting than the Halving Algorithm? The answer is that there are better algorithms in some cases in which the learning algorithms can do a complicated and expensive recursive look-ahead over all possible outcomes. Still, the Halving Algorithm gives us a bound on the performance of reasonable learning algorithms in this setting:

$$\min_A M_A(\mathscr{H}) \leq M_{\text{halving}}(\mathscr{H}) \leq \lg |\mathscr{H}|$$

Continuing, we can also ask how VC-dim($\mathscr{H}$) fits in here. If $\mathscr{H}$ has a VC-dimension of d, this means that $\exists x_1, x_2, \ldots, x_d$ which are shattered by $\mathscr{H}$. In the online learning model, an adversary could force the learning algorithm A to make at least d mistakes by giving it this shattered set as follows:

- Give A example x_1.
- A makes prediction $\hat{y}$.
- Choose to label x_1 with the opposite label $y \neq \hat{y}$. (See note below.)
- Do the same for the other $d-1$ examples in the shattered set.

Note: We can assign an arbitrary labeling to the d examples since they are shattered. Moreover, the adversary can "choose" this labeling because the data is arbitrary; we make no statistical assumptions about the data. To phrase this more rigorously, since A is deterministic, an adversary could simulate how A would run beforehand and choose examples and labels in a way such that A would always make mistakes on the d examples; then, when A actually ran, it would be ensured to make at least d mistakes. This differs from the PAC model, where the adversary could choose the target concept c and distribution D beforehand but could not choose the actual examples.

Up until now, in our discussion of online learning with expert advice, we have assumed that at least one expert is perfect. This might not be the case, however, so we now turn to a model in which we do not make this assumption.

2.3 Weighted Majority Algorithm

In the previous model, when at least one expert was perfect, we threw out experts which made mistakes. If no experts are perfect, we can use a less extreme but analogous method which gives less importance to experts which make mistakes. The Weighted Majority Algorithm does this by maintaining weight w_i for each expert i, where higher weights correspond to greater importance. On each round, an expert's weight (importance) is decreased iff the expert makes a mistake. The learner's prediction is a weighted majority vote by all of the experts.

Algorithm:

```
Let $\beta \in [0, 1)$

Maintain weights: $w_i$ = weight of expert $i$, initialized to $w_i = 1, \forall i$

On each round,

    $q_0 = \sum_{i:\xi_i=0} w_i$        $q_1 = \sum_{i:\xi_i=1} w_i$
    If $q_1 > q_0$, predict $\hat{y} = 1$, else predict $\hat{y} = 0$
    $\forall i$: if $\xi_i \neq y$ then $w_i \leftarrow \beta \cdot w_i$
```

Theorem 1 *Given the above learning setting, the number of mistakes made by the Weighted Majority Algorithm is bounded as:*

$$(\#\, mistakes\, of\, learner) \leq a_\beta \cdot (\#\, mistakes\, of\, best\, expert) + c_\beta \cdot \lg N$$

where

$$a_\beta = \frac{\lg \frac{1}{\beta}}{\lg\left(\frac{2}{1+\beta}\right)} \quad and \quad c_\beta = \frac{1}{\lg\left(\frac{2}{1+\beta}\right)}$$

Before we prove this, let us look at the meaning of this bound. If we divide all of the terms by T, the number of trials, then we get a bound which looks like:

$$(rate\, at\, which\, learner\, makes\, mistakes) \leq a_\beta(rate\, at\, which\, best\, expert\, makes\, mistakes) + \frac{c_\beta \lg N}{T}$$

The last term approaches 0 as $T \to \infty$. Also, note that the bound's dependence on $\lg N$ means that we can use a very large number of experts and still have a reasonable bound. To give an idea of the relative importance of the terms in the bound, the table below shows how a_β and c_β vary as functions of β. When we penalize experts a lot for making mistakes ($\beta \to 0$), the first term becomes large, and when we penalize experts very little for making mistakes ($\beta \to 1$), the second term becomes very large.

β	a_β	c_β
1/2	≈ 2.4	≈ 2.4
$\to 0$	$\to \infty$	$\to 1$
$\to 1$	$\to 2$	$\to \infty$

This is a decent bound, but we do not like the fact that $a_\beta \to 2$ as $\beta \to 1$ since this means we have a poor bound when no expert is very good. For example, if the best expert has 30% error, then this bound only guarantees the learner has at most (about) 60% error, which is worse than guessing randomly.

Just as the Weighted Majority Algorithm is, in a sense, analogous to the Halving Algorithm from the previous learning model, we can analyze the Weighted Majority Algorithm in almost the same way to get the bounds in the above theorem.

Proof of Theorem 1

Let $W = \sum_i w_i$.
On some round, say $y = 0$. Then

$$\begin{aligned} W_{new} &= q_1\beta + q_0 \\ &= q_1\beta + (W - q_1) \\ &= W - (1-\beta)q_1 \end{aligned}$$

Now, suppose the learner made a mistake. Then the learner predicted 1, meaning $q_1 \geq \frac{1}{2}W$. This implies

$$W_{new} \leq W - (1-\beta) \cdot \frac{1}{2}W = \left(\frac{1+\beta}{2}\right)W$$

A similar analysis holds if we say $y = 1$ and suppose the learner makes a mistake.

So, after m mistakes, $W \leq \left(\frac{1+\beta}{2}\right)^m N$ since $W = N$ initially.
Let $L_i = \#$ mistakes of expert i. Then $w_i = \beta^{L_i}$.
Combining these, for any expert i, we know that

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

and solving for m gives

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

This holds for all experts and, in particular, for the best expert, so we have the bound from our theorem:

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

□