COS 511: Foundations of Machine Learning

Rob Schapire
Scribe: Qian Xi

Lecture #14
March 30, 2006

In the previous lecture, we introduced a new learning model, *the Online Learning Model*. After seeing a concrete example in stock market, we set up the model for *Learning With Expert Advice*. First of all, let's restate the procedure of this online model formally.

N = number of experts

For $t = 1, 2, ..., T$ trials

each expert i predicts $\xi_i \in \{0, 1\}$
learner predicts $\hat{y} \in \{0, 1\}$
observe outcome $y \in \{0, 1\}$

Then we looked at a particular algorithm for it, *the Weighted Majority Algorithm (WMA)*. In this algorithm, we maintain a set of weights over all experts. So w_i is the weight on expert i.

Initially $w_i = 1$

on each round

$q_0 = \sum_{i:\xi_i=0} w_i \qquad q_1 = \sum_{i:\xi_i=1} w_i$

$$\text{predict } \hat{y} = \begin{cases} 1 & \text{if } q_1 > q_0 \\ 0 & \text{else} \end{cases}$$

$\forall i$: if $\xi_i \neq y$, then $w_i \leftarrow \beta \cdot w_i$, where $\beta \in [0, 1)$

We also proved a theorem bounding the number of mistakes the learner(master) makes.

Theorem 1 *The number of mistakes made by the Weighted Majority Algorithm is bounded as:*

$$(\#\ \textit{mistakes of the learner}) \leq a_\beta \cdot (\#\ \textit{mistakes of the best expert}) + c_\beta \cdot \lg N$$

where

$$a_\beta = \frac{\lg \frac{1}{\beta}}{\lg(\frac{2}{1+\beta})} \qquad \textit{and} \qquad c_\beta = \frac{1}{\lg(\frac{2}{1+\beta})}.$$

1 Randomized Weighted Majority Algorithm

1.1 Some Argument About The Theorem

We talked about these constants a_β and c_β. What we really want is that a_β gets as close as possible to 1. But instead of this, $a_\beta \to 2$ as $\beta \to 1$. Last time we started talking about whether we could do better than that to make the constant a_β close to 1. So we're looking for how to improve this algorithm.

A student suggested an algorithm last time. The idea is that we keep track of which expert is the best at any time, and then we make the prediction by following the expert doing the best so far. That approach is called *Follow the Leader*, which recently has been the subject of very interesting theoretical work, but we won't talk about it in class.

1.2 Randomized Weighted Majority Algorithm

A different idea is to introduce some kind of randomization. So here we're just doing a straightforward majority vote. If the 51% of the experts predict a positive example, then you predict positive. So we're thinking about the situation that the fraction of the experts who are predicting positive and negative, counting the weights, are quite close to 1/2.

So there's a different variant of *the Weighted Majority Algorithm*, *Randomized Weighted Majority Algorithm*. The predictions that are made by the algorithm are randomized. What we do is to compute the fraction of the experts predicting positive or negative, and predict randomly according to that fraction:

$$\text{predict } \hat{y} = \begin{cases} 1 & \text{with probability } \frac{q_1}{W} \\ 0 & \text{otherwise} \end{cases}$$

where

$$W = \textstyle\sum_i w_i = q_0 + q_1.$$

One equivalent way of saying that is we choose one of the experts, expert i, with probability w_i/W and we predict whatever that expert says:

$$\text{choose expert } i \text{ with probability } w_i/W, \text{ let } \hat{y} = \xi_i.$$

We have the similar theorem bounding the number of mistakes *Randomized Weighted Majority Algorithm* makes.

Theorem 2 *The number of mistakes made by the Randomized Weighted Majority Algorithm is bounded as:*

$$E[(\#\textit{ mistakes of the learner})] \leq a_\beta \cdot (\#\textit{ mistakes of the best expert}) + c_\beta \cdot \ln N$$

where

$$a_\beta = \frac{\ln \frac{1}{\beta}}{1-\beta} \qquad \textit{and} \qquad c_\beta = \frac{1}{1-\beta}.$$

Notice that the expectation is only in terms over the randomization of the learning algorithm. We still assume that the examples and the experts' predictions are not random. The only randomness is the randomness when the learner makes his own prediction.

What's important in this algorithm is that $a_\beta \to 1$ as $\beta \to 1$. So by introducing the randomization, we halve the number of mistakes this algorithm is going to make. To prove this theorem, we use the similar technique as we used last time, which is to keep track of the sum of the weights.

Proof: Let's think about one particular round of the algorithm.

On round t:

$$l = \text{probability of learner making a mistake} = \frac{\sum_{i:\xi_i \neq y} w_i}{W}$$

$$W_{new} = \textstyle\sum_{i:\xi_i \neq y} w_i \cdot \beta + \sum_{i:\xi_i = y} w_i = lW\beta + (W - lW) = W(1 - l(1-\beta)).$$

So at the end of T steps,

$$\begin{aligned} W_{final} &= N \cdot (1 - l_1(1-\beta)) \cdot (1 - l_2(1-\beta)) \cdots (1 - l_T(1-\beta)) \\ &= N \cdot \prod_t (1 - l_t(1-\beta)) \\ &\leq N \cdot \exp(-\sum_{t=1}^{T} l_t(1-\beta)) \\ &= N \cdot \exp(-(1-\beta)\sum_{t=1}^{T} l_t). \end{aligned}$$

Let $L_A = \sum_{t=1}^{T} l_t$, so $L_A = E[(\#$ mistakes of the learner)].
We can get the lower bound of W_{final} just as we did in the proof of the previous lecture. So $\forall i$:

$$\beta^{L_i} = w_i \leq W_{final} \leq N \cdot e^{(-(1-\beta)L_A)}.$$

Then it's easy to get L_A out of the inequality:

$$L_A \leq a_\beta \cdot L_i + c_\beta \cdot \ln N.$$

So

$$L_A \leq a_\beta \cdot \min_i L_i + c_\beta \cdot \ln N$$

where $\min_i L_i$ is # of mistakes of the best expert.

1.3 Discussion About the Value of β

Let's talk a little bit more about this bound. First of all, how do we tune β? The best way of tuning β is to depend on the number of mistakes made by the best expert. Let's say you just have an upper bound on that and you know $min_i L_i \leq K$ ahead of time. Set

$$\beta = \frac{1}{1+\sqrt{\frac{2\ln N}{K}}}.$$

By plugging in β,

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

It turns out that you can improve the last two constants.

As shown in Figure 1, given $\frac{q_1}{W} = \frac{\sum_{i:\xi_i=1} w_i}{W}$, we want to see what is the probability of $\hat{y} = 1$. So the red line describes *the Weighted Majority Algorithm*, in which if $\frac{q_1}{W} < \frac{1}{2}$, $Pr[\hat{y} = 1] = 0$, otherwise $Pr[\hat{y} = 1] = 1$. The blue line depicts *the Randomized Weighted Majority Algorithm*, in which $Pr[\hat{y} = 1]$ is always equal to $\frac{q_1}{W}$. The yellow line depicts the predictions of a different randomized algorithm for which it can be shown that we can improve the bound to obtain:

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

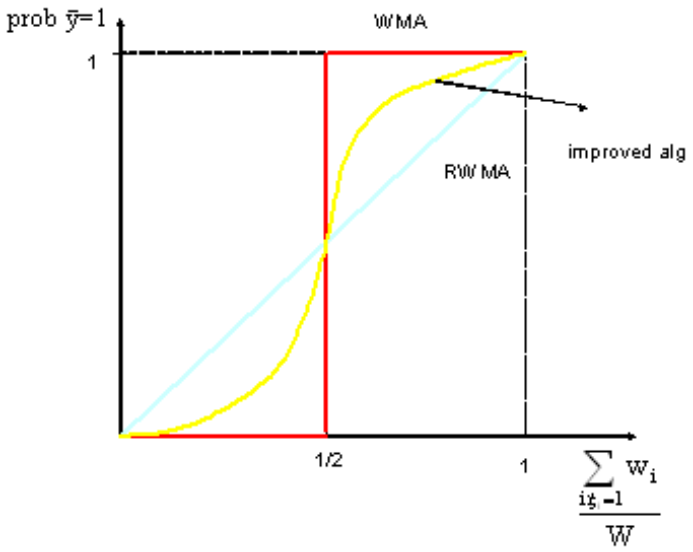

Figure 1: The Comparison of WMA, RWMA and the Improved Algorithm

When $K = 0$, there exists one perfect expert who always predicts correctly as we talked about last time. We got the bound with $\lg N$ by Halving Algorithm. But here in fact, we get improved by half and the bound is $\frac{\lg N}{2}$.

Another thing to do is we can suppose that $K = rT$, where r can be regarded as the likely rate of the best expert making mistakes. By plugging in K and dividing both sides of the inequality by T, we have:

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

The last two constants indicate how fast the bound is converging:

$$\begin{array}{ll} \text{if} \quad r = o(1) & \text{converging rate} \sim \frac{1}{T} \\ \text{otherwise} & \text{converging rate} \sim \sqrt{\frac{1}{T}} \end{array}$$

Let's think about one more thing: what is the trivial bound on K? In almost any case, K is going to be at most $T/2$. Why? For instance, we have one expert that always predicts 0 and another that always predicts 1. Then the number of mistakes of one of them will never go beyond $T/2$. So by plugging in $K \leq T/2$,

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

By giving up our assumption of random data, you might have a feeling that we are losing something. But in fact we're not. We'll prove it by looking at the worst case in which the data is random, and we also suppose every expert just guesses randomly.

$$\xi_i = \begin{cases} 1 & \text{with probability } \frac{1}{2} \\ 0 & \text{with probability } \frac{1}{2} \end{cases}$$

$$y = \begin{cases} 1 & \text{with probability } \frac{1}{2} \\ 0 & \text{with probability } \frac{1}{2} \end{cases}$$

For *any* learning algorithm,

$$\text{E[\# of mistakes of the learner]} = \frac{T}{2}.$$

This is because no matter what you predict, the chance of getting right answer is 1/2. This holds for the experts as well:

$$E[L_i] = \frac{T}{2}.$$

However, the expected number of mistakes of the *best* expert will not be $\frac{T}{2}$, but instead, will be a little better than $\frac{T}{2}$. Specifically, it can be shown that:

$$E[\min_i L_i] \approx \frac{T}{2} - \sqrt{\frac{T \ln N}{2}}.$$

So for any learning algorithm,

$$E[L_A] \gtrsim E[\min_i L_i] + \sqrt{\frac{T \ln N}{2}}.$$

By comparing the upper bound and the approximate lower bound, we can see that they're quite close, so we're not losing anything.

2 The Perceptron Algorithm

Let's come back to *the Weighted Majority Algorithm* first. Think about a question one student asked last time. A set of experts are making weather predictions, and a subcommittee of them are good. Can we do well by combining the results of these experts rather than identifying a single good expert. So we assume that there's some combination of the experts that are making really good predictions and we want to do almost as well as that subset of the experts.

Let's formalize the thought and we're going to change the notation a little bit just for mathematical convenience:

N = number of experts

For $t = 1, 2, ..., T$ trials

get $\mathbf{x}_t \in \{-1, +1\}^N$ (or more generally, $\mathbf{x}_t \in \mathbb{R}^N$)
learner predicts $\hat{y} \in \{-1, +1\}$
observe outcome $y \in \{-1, +1\}$

The expert predictions are now identified with components of $\mathbf{x}_t$, although generally, the vectors $\mathbf{x}_t$ can be any points in $\mathbb{R}^N$.

We assume that there's a set of weights over the experts. The learner's prediction is the sign of the weighted majority sum.

$\exists \mathbf{u} \in \mathbb{R}^N$ s.t.

$\forall t, y_t = sign(\mathbf{u} \cdot \mathbf{x}_t)$ (i.e. $y_t(\mathbf{u} \cdot \mathbf{x}_t) > 0$)

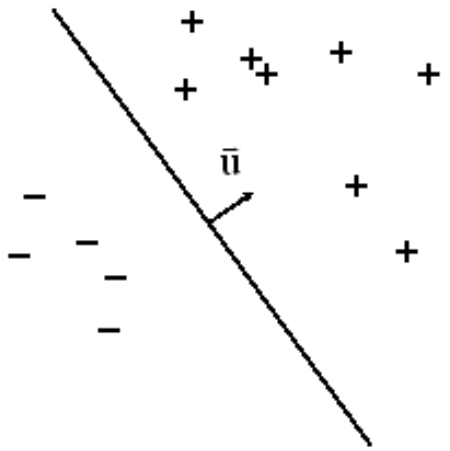

Figure 2: The Hyperplane

Here we're actually allowing some of the experts to have negative weights.

Think about it geometrically. Assume that we have a separating hyperplane $\mathbf{u}$. The points above are positive examples and the ones below are negative examples. We're trying to learn the hyperplane that best divides the positive and negative examples as shown in Figure 2.

We're going to talk about algorithms making weight vectors over experts.

Initialize $\mathbf{w}_1$

for $t = 1, 2, ..., T$

predict $\hat{y}_t = sign(\mathbf{w}_t \cdot \mathbf{x}_t)$
update $\mathbf{w}_{t+1} = F(\mathbf{w}_t, \mathbf{x}_t, y_t)$

We update the weight vector in terms of the value of itself and the current example as opposed to all examples we've received at that point.

Now we introduce an algorithm called *the Perceptron Algorithm.*

$\mathbf{w}_1 = \mathbf{0}$

update:

if the learner makes a mistake ($y_t(\mathbf{w}_t \cdot \mathbf{x}_t) \leq 0$)
$\mathbf{w}_{t+1} = \mathbf{w}_t + y_t \cdot \mathbf{x}_t$
else $\mathbf{w}_{t+1} = \mathbf{w}_t$

How does the update work? Think about it geometrically again. Here is an example, shown in Figure 3. Let's suppose we make a mistake on this example. Making a mistake means our algorithm predicts that $\mathbf{x}_t$ is below the hyperplane, so it's a negative example. But in fact, it's positive. So this time $y_t = 1$. What does the update step do? It adds $y_t \cdot \mathbf{x}_t$ to the new vector $\mathbf{w}_{t+1}$, which tips the direction of the hyperplane to get close to $\mathbf{x}_t$. So it's more likely to predict it correctly next time.

Next, we add some assumptions. First of all, we assume that:

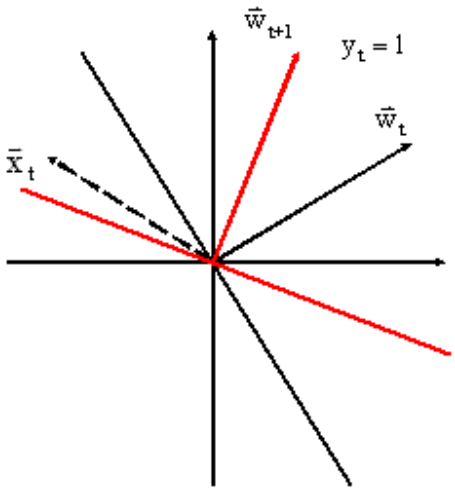

Figure 3: An Example of the Update Step In *the Perceptron Algorithm*

$$\forall t: \quad \|\mathbf{x}_t\|_2 \leq 1$$

We're not violating generality by assuming this, because it doesn't change the prediction of any expert.

The second assumption is that:

$$\exists \mathbf{u}, \quad \delta > 0: \qquad y_t(\mathbf{u} \cdot \mathbf{x}_t \geq \delta > 0)$$

$$\|\mathbf{u}\|_2 = 1$$

We're making a stronger assumption. It geometrically means that there exists a hyperplane separating the examples by margin of δ as *Support Vector Machine*, shown in Figure 4.

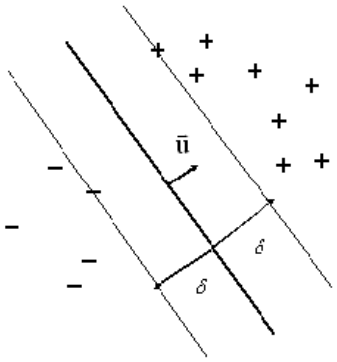

Figure 4: *Support Vector Machine*

Given those assumptions, we have a theorem.

Theorem 3 *Assume that the learner makes a mistake on every round, i.e. T = # mistakes, we have:*

$$\#\ mistakes \leq \tfrac{1}{\delta^2}.$$

Suppose that we've already proved the theorem. In the previous lecture, with A being a deterministic online learning algorithm, and $M_A(\mathscr{H}) = \max_{\text{adversary}}$ (# mistakes made by A when learning $c \in \mathscr{H}$), we have:

$$\text{VC-dim}(\mathscr{H}) \leq \min_A M_A(\mathscr{H}).$$

So:

$$\begin{aligned} \text{VC-dim}(\mathscr{H}) &\leq \text{\# mistakes for any learning algorithm} \\ &\leq \text{\# mistakes for } \textit{the Perceptron Algorithm} \\ &\leq \frac{1}{\delta^2}. \end{aligned}$$

So, this shows that the VC-dimension of hyperplanes with margin at least δ is no more than $1/\delta^2$.

Proof: In the last proof, we picked up some function to keep track of the sum of weights. This time we focus on a different quality: the potential function, which is the angle between $\mathbf{u}$ and $\mathbf{w}_t$.

$$\begin{aligned} \Phi_t &= \cos(\textit{angle between } \mathbf{u} \textit{ and } \mathbf{w}_t) \\ &= \frac{\mathbf{u} \cdot \mathbf{w}_t}{\|\mathbf{w}_t\|_2} \\ &\leq 1. \end{aligned}$$

Then we're trying to get the lower bound of Φ_t.

Step 1

$$\mathbf{w}_{T+1} \cdot \mathbf{u} \geq T\delta.$$

Proof: The proof is pretty straightforward by plugging in definitions.

$$\begin{aligned} \mathbf{w}_{t+1} \cdot \mathbf{u} &= (\mathbf{w}_t + y_t \mathbf{x}_t) \cdot \mathbf{u} \\ &= \mathbf{w}_t \cdot \mathbf{u} + y_t(\mathbf{u} \cdot \mathbf{x}_t). \end{aligned}$$

Notice that by assumption, $y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq \delta$. So $\mathbf{w}_t \cdot \mathbf{u}$ increases by at least δ on each round. Since the base case of $\mathbf{w}_t$ is $\mathbf{w}_1 = \mathbf{0}$, we have:

$$\mathbf{w}_{T+1} \cdot \mathbf{u} \geq T\delta.$$

Step 2 *By keeping track of the denominator:*

$$\|\mathbf{w}_{T+1}\|_2^2 \leq T.$$

Proof:

$$\begin{aligned}
\|\mathbf{w}_{t+1}\|_2^2 &= (\mathbf{w}_t + y_t\mathbf{x}_t)\cdot(\mathbf{w}_t + y_t\mathbf{x}_t) \\
&= \underbrace{\mathbf{w}_t\cdot\mathbf{w}_t}_{\|\mathbf{w}_t\|_2^2} + 2\underbrace{y_t(\mathbf{w}_t\cdot\mathbf{x}_t)}_{\leq 0} + \underbrace{y_t^2\mathbf{x}_t\cdot\mathbf{x}_t}_{\leq 1} \\
&\leq \|\mathbf{w}_t\|_2^2 + 1.
\end{aligned}$$

So $\|\mathbf{w}_{t+1}\|_2^2$ increases by at most 1 on every round and $\|\mathbf{w}_1\|_2^2 = \mathbf{0}$. Then:

$$\|\mathbf{w}_{T+1}\|_2^2 \leq T.$$

By plugging in the results of Step 1 and Step 2:

$$\begin{aligned}
\tfrac{T\delta}{\sqrt{T}} \leq \Phi_{T+1} &\leq 1 \\
T &\leq \tfrac{1}{\delta^2}.
\end{aligned}$$