COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: (James) Zhen XIANG

Lecture #15
April 2, 2008

1 Review of the Perceptron Algorithm

In the last few lectures, we talked about various kinds of online learning problems. We started with the simplest case in which there is a perfect expert, who is always right. In the last lecture, we generalize this into the case in which we might not have a "perfect expert", but instead we have a panel or subcommittee of experts, whose combined prediction is always right. We also introduced the perceptron algorithm to find the correct combination of experts.

The perceptron algorithm is an example of a weight-update algorithm, which has the general framework as follows:

Initialize $\mathbf{w}_1$
for $t = 1, 2, \cdots, T$
  predict $\hat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$
  update $\mathbf{w}_{t+1} = F(\mathbf{w}_t, \mathbf{x}_t, y_t)$.

In general, the updating rule can be dependent on the whole observation history, but we'll only consider those F that have the form $F(\mathbf{w}_t, \mathbf{x}_t, y_t)$, i.e. those updating rules that only use information of the previous round.

The perceptron algorithm uses the initialization and updating rules as follows:

Initialize $\mathbf{w}_1 = \mathbf{0}$
update:
  if $\hat{y}_t \neq y_t$, where $\hat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$
    $\mathbf{w}_{t+1} = \mathbf{w}_t + y_t \mathbf{x}_t$
  else $\mathbf{w}_{t+1} = \mathbf{w}_t$.

In the last lecture, we have proved that under assumptions:

- $||\mathbf{x}_t||_2 \leq 1$
- $\exists\ \delta, \mathbf{u}$ s.t. $||\mathbf{u}||_2 = 1, y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq \delta > 0$.

the number of mistakes made by perceptron algorithm has an upper bound:

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

2 The Performance of Perceptron Algorithm on an Example

Now let's apply perceptron algorithm and its performance bound to a specific example. In this example, we have N experts. Each experts predicts $+1$ or -1 in each round. There is a panel of k experts; when we take the majority vote of these k experts, we always get the right prediction. Now we would like to use perceptron algorithm to this online learning problem. We are interested in how many mistakes perceptron algorithm will make in this case.

Using the notation above, in each round t, the output of experts is a vector $\mathbf{x}_t \in \{-1,+1\}^N$, for example:

$$\mathbf{x}_t = (-1,+1,+1,-1,\ldots,+1). \tag{2}$$

We assume there is a vector $\mathbf{u}$, which has k 1's and $N-k$ 0's as its elements, for example, $\mathbf{u}$ might look like:

$$\mathbf{u} = (0,1,0,1,\ldots,1). \tag{3}$$

When we take the majority vote of those experts that correspond to positions of 1's in $\mathbf{u}$, the prediction is always right. i.e. $y_t = sign(\mathbf{u}\cdot\mathbf{x}_t)$.

To use the bound of perceptron algorithm, we first have to normalize $\mathbf{x}_t$ and $\mathbf{u}$ such that they have unit lengths:

$$\mathbf{x}_t = \frac{1}{\sqrt{N}}(-1,+1,+1,-1,\ldots,+1) \tag{4}$$

$$\mathbf{u} = \frac{1}{\sqrt{k}}(0,1,0,1,\ldots,1). \tag{5}$$

After normalization, we see that $y_t(\mathbf{u}\cdot\mathbf{x}_t)$ is always an integer times $\frac{1}{\sqrt{Nk}}$ (we assume here that k is odd). So

$$y_t(\mathbf{u}\cdot\mathbf{x}_t) \geq \frac{1}{\sqrt{Nk}} = \delta. \tag{6}$$

Therefore we have a positive margin

$$\delta = \frac{1}{\sqrt{Nk}} \tag{7}$$

so according to theorem (1), the number of mistakes is bounded by

$$\#mistakes \leq \frac{1}{\delta^2} = Nk. \tag{8}$$

which is the product of the number of experts and the size of the panel of correct experts. What we don't like about this bound is that it's linearly increasing with the total number of experts, which means we can't afford to have too many experts. In practice, we often have a lot of experts, or a lot of features/attributes to select from. So it's important for our algorithm to have good performance even when the number of experts is huge. Today, we'll introduce another algorithm (Winnow Algorithm), which gives a bound that is more tolerant of the total number of experts.

3 Winnow Algorithm

We'll introduce Winnow algorithm, which is another algorithm to find a perfect subcommittee of experts. The Winnow algorithm will make fewer mistakes than the perceptron algorithm, especially when the number of experts increases.

The difference between Winnow algorithm and perceptron algorithm is how they update weights $\mathbf{w}_t$. Recall that in perceptron algorithm, we're adjusting the weight by adding $y_t\mathbf{x}_t$ to $\mathbf{w}_t$: $\mathbf{w}_{t+1} = \mathbf{w}_t + y_t\mathbf{x}_t$, that is to increase the weights of those experts who made a right prediction by adding a number. Instead of adding, we could also try multiplication. That is the main idea of "Winnow" algorithm: we multiply the weights of those experts who were right by a number greater than 1, and reduce the weights of those experts who made

a mistake by a ratio smaller than one. In detail, let the weight $w_{t,i}$ denote the weight of the i^{th} expert in the t^{th} round; for convenience, we assume $\sum_i w_{t,i} = 1$. The "Winnow" algorithm initializes and updates weights as follows:

Initialize $w_{1,i} = \frac{1}{N}$

In each round:

- if the prediction of the master is right, i.e. $y_t = sign(\mathbf{w}_t \cdot \mathbf{x}_t)$, then don't change weights
- if the prediction of the master is wrong, then change weights by: $w_{t+1,i} = w_{t,i} e^{\eta y_t x_{t,i}} / Z_t$, where Z_t is the normalization factor to make $\sum_i w_{t,i} = 1$.

This update rule reminds us of the update rule for probability distribution in Adaboost. Actually, this also looks like the WMA(Weighted Majority Algorithm) that we have covered before: notice that if we let $\beta = e^{-2\eta}$, and assume $x_{t,i}$ only takes value $\{-1, +1\}$, then the update rule becomes:

$$w_{t+1,i} = \frac{e^{\eta} w_{t,x}}{Z_t} \times \begin{cases} 1, & \text{if } x_{t,i} = y_t \\ \beta, & \text{otherwise} \end{cases} \tag{9}$$

This update rule is the same as the update rule of WMA. Actually Winnow and WMA are the same kind of algorithm, but we analyzed them in different ways. We should still be aware of the difference between this algorithm and WMA: WMA updates the weights of experts in every round; if an expert made a mistake, his weight is adjusted. But in this Winnow algorithm, the weights are updated only when the master (the majority vote of the weighted experts) makes a mistake.

4 Proving the Performance of Winnow Algorithm

We now analyze the performance of this algorithm. First of all, we make the following assumptions:

- a mistake is made at every round
- $\forall t, ||\mathbf{x}_t||_\infty = 1$
- $\exists\ \delta, \mathbf{u}$, s.t.
  - $\forall t, y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq \delta > 0$.
  - $||\mathbf{u}||_1 = 1$
  - $u_i \geq 0$ *i.e.* each element of $\mathbf{u}$ is nonnegative; we'll see later that this assumption can be removed.

Notice that we use L_1 and L_∞ norm here instead of the L_2 norm that we used in perceptron algorithm.

We have the following theorem:

Theorem: under previous assumptions, we have the following upper bound on the number of mistakes :

$$\#mistakes \leq \frac{\ln N}{\eta\delta + \ln(\frac{2}{e^{\eta} + e^{-\eta}})}. \tag{10}$$

If we choose an optimal η to minimize the bound, we get when $\eta = \frac{1}{2}\ln(\frac{1+\delta}{1-\delta})$,

$$\#mistakes \leq \frac{2\ln N}{\delta^2}. \tag{11}$$

Before proving this theorem, we would first like to apply this theorem to the previous example in section 2. Since we are using L_1 and L_∞ norm here instead of L_2 norm, we have to renormalize $\mathbf{x}_t$ and $\mathbf{u}$ as follows:

$$\mathbf{x}_t = (-1, +1, +1, -1, \ldots, +1),\ ||\mathbf{x}_t||_\infty = 1 \tag{12}$$

$$\mathbf{u} = \frac{1}{k}(0, 1, 0, 1, \ldots, 1),\ ||\mathbf{u}||_1 = 1. \tag{13}$$

We have $y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq \frac{1}{k} = \delta$, so by the theorem, the number of mistakes made by Winnow algorithm satisfies:

$$\#mistakes \leq \frac{2\ln N}{\delta^2} = 2k^2 \ln N. \tag{14}$$

So the number of mistakes only depends on the logarithm of N. This bound is much better than the bound of perceptron algorithm (Nk) when N is large. This indicates that Winnow algorithm is more tolerant of the total number of experts.

We now prove this theorem. The idea is similar to the proof we used for perceptron algorithm: we choose a quantity and watch it in each round. If we can show that the quantity is decreased by a certain amount at each round and this quantity can never go below 0, then we can get a bound on the total number of rounds.

In this case we watch the distance between the updated weight in each round $\mathbf{w}_t$ and our target weight: the ground truth weight $\mathbf{u}$. Both $\mathbf{u}$ and $\mathbf{w}_t$ are probability distributions. A natural way to measure their distance would be the relative entropy:

$$\Phi_t = RE(\mathbf{u}||\mathbf{w}_t) \tag{15}$$

We want to show that at each round, this quantity is decreased by at least C, where C is some constant we're going to compute later. We now compute the difference between Φ_{t+1} and Φ_t:

$$\begin{aligned}
\Phi_{t+1} - \Phi_t &= \sum_i u_i \ln \frac{u_i}{w_{t+1,i}} - \sum_i u_i \ln \frac{u_i}{w_{t,i}} \\
&= \sum_i u_i \ln \frac{w_{t,i}}{w_{t+1,i}} \\
&= \sum_i u_i \ln \left(\frac{Z_t}{e^{\eta y_t x_{t,i}}} \right) \\
&= \sum_i u_i (\ln Z_t - \eta y_t x_{t,i}) \\
&= \ln Z_t - \sum_i u_i \eta y_t x_{t,i} \\
&= \ln Z_t - \eta y_t (\mathbf{u} \cdot \mathbf{x}_{t,}) \\
&\leq \ln Z_t - \eta\delta.
\end{aligned}$$

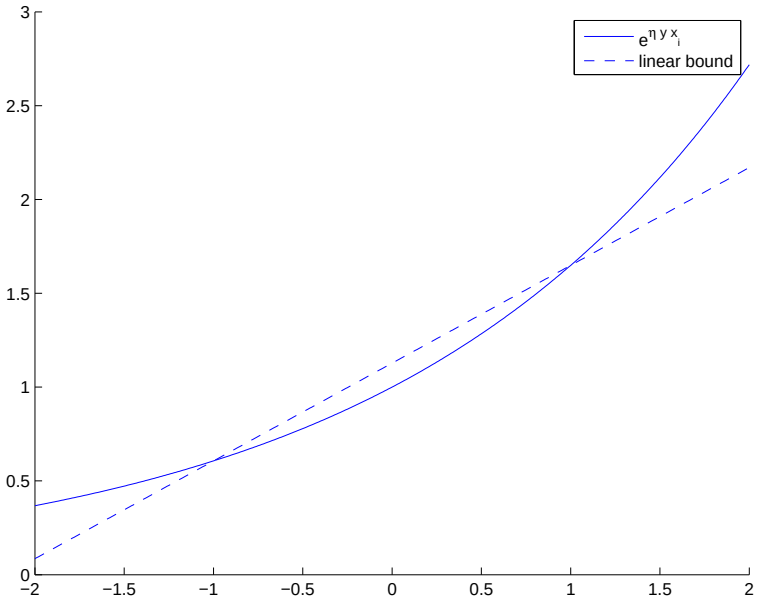

Figure 1: Using Linear Function to Bound Exponential Function

The last inequality is because δ is the margin. Now we want to make estimations on Z_t. Let's omit subscript t for simplicity. So Z means Z_t, w_i means $w_{t,i}$, etc. We know that Z is the normalization factor and can be computed as:

$$Z = \sum_i w_i e^{\eta y x_i}. \tag{16}$$

We'll bound the exponential term by a linear function, as illustrated in figure (1):

$$e^{\eta x} \leq \left(\frac{1+x}{2}\right) e^{\eta} + \left(\frac{1-x}{2}\right) e^{-\eta}, \text{ for } -1 \leq x \leq 1. \tag{17}$$

Using this bound, we have:

$$\begin{aligned} Z &= \sum_i w_i e^{\eta y x_i} \\ &\leq \sum_i w_i \left(\frac{1+yx_i}{2}\right) e^{\eta} + \sum_i w_i \left(\frac{1-yx_i}{2}\right) e^{-\eta} \\ &= \frac{e^{\eta}+e^{-\eta}}{2} \sum_i w_i + \frac{e^{\eta}-e^{-\eta}}{2} y \sum_i w_i x_i \\ &= \frac{e^{\eta}+e^{-\eta}}{2} + \frac{e^{\eta}-e^{-\eta}}{2} y(\mathbf{w}\cdot\mathbf{x}) \\ &\leq \frac{e^{\eta}+e^{-\eta}}{2}. \end{aligned}$$

The last inequality comes from the assumption that the master makes a wrong prediction every time, so the second term is less than 0.

So we have:

$$\begin{aligned}\Phi_{t+1} - \Phi_t &\leq \ln Z_t - \eta\delta \\ &\leq \ln\left(\frac{e^{\eta} + e^{-\eta}}{2}\right) - \eta\delta = -C.\end{aligned}$$

So Φ_t is decreased by at least $C = \ln\left(\frac{2}{e^{\eta}+e^{-\eta}}\right) + \eta\delta$ for each round, and $\Phi_1 = \sum_i \ln\frac{u_i}{1/N} = \sum_i u_i \ln(Nu_i) \leq \sum_i u_i \ln N = \ln N$, so the total number of mistakes is bounded by:

$$\begin{aligned}\#mistakes &\leq \frac{\Phi_1}{C} \\ &\leq \frac{\ln N}{\eta\delta + \ln\left(\frac{2}{e^{\eta}+e^{-\eta}}\right)}.\end{aligned}$$

The inequality is proved. When we choose $\eta = \frac{1}{2}\ln(\frac{1+\delta}{1-\delta})$, $C = RE(\frac{1}{2} - \frac{\delta}{2}||\frac{1}{2}) \geq \frac{\delta^2}{2}$, so the bound becomes $\frac{2\ln N}{\delta^2}$. Proved.

5 Balanced Winnow Algorithm

How can we get rid of the assumption $u_i \geq 0$? What if some components of $\mathbf{u}$ are negative? One "quick and dirty" trick is to double the size of N and add a complementary version of each expert. For example:

$$\begin{aligned}\mathbf{x}_t = (1, -0.7, 0.32) &\longrightarrow \mathbf{x}_t^{'} = (1, -0.7, 0.32 \mid -1, 0.7, -0.32) \\ \mathbf{u} = (1, 0.2, -0.2) &\longrightarrow \mathbf{u}^{'} = (1, 0.2, 0 \mid 0, 0, 0.2)\end{aligned}$$

After this transformation, we preserved the norm: $||\mathbf{x}_t||_\infty = ||\mathbf{x}_t^{'}||_\infty$, $||\mathbf{u}||_1 = ||\mathbf{u}^{'}||_1$, we also have $\mathbf{u}\cdot\mathbf{x}_t = \mathbf{u}^{'}\cdot\mathbf{x}_t^{'}$. If we use $\mathbf{w}^+$ and $\mathbf{w}^-$ to denote the original and duplicated parts of vector $\mathbf{w}$, the Winnow algorithm will have the following form:

Initialize: $w_{1,i}^+ = w_{1,i}^- = \frac{1}{2N}$

Predict: $y_t = sign(\mathbf{w}^+\cdot\mathbf{x}_t - \mathbf{w}^-\cdot\mathbf{x}_t)$

Update:

if the prediction is right, then don't change weights

if the prediction is wrong, then update as follows:

$w_{t+1,i}^+ = w_{t,i}^+\frac{e^{\eta y_t x_{t,i}}}{Z_t}$

$w_{t+1,i}^- = w_{t,i}^-\frac{e^{-\eta y_t x_{t,i}}}{Z_t}$

This algorithm is called the balanced Winnow algorithm.

6 A Comparison Between Perceptron Algorithm and Winnow Algorithm

The major difference between these two algorithms are the way they update weights. They also use different norms.

Perceptron	Winnow / WMA
additive update	multiplicative update
$\|\|\mathbf{x}_t\|\|_2 = 1$ $\|\|\mathbf{u}\|\|_2 = 1$	$\|\|\mathbf{x}_t\|\|_\infty = 1$ $\|\|\mathbf{u}\|\|_1 = 1$
analogous to SVM	analogous to Boosting

7 Regression

We have been talking about classification problems for a while. In these problems, we try to minimize the probability of our classifier making a mistake. For the next few lectures, we will begin to talk about a different kind of problem.

Let's start by looking at an example. A TV station wants to hire a meteorologist to predict the weather. Two applicants, say Alice and Bob, made predictions on the next day's weather during the interview:

Alice said there is a 70% chance of raining tomorrow.

Bob said there is an 80% chance of raining tomorrow.

On the second day, it turned out that it did rain. The question is, which one should we hire? This is a totally different problem than classification. What is complicated here is that although it rained on the next day, we still don't know the true probability of raining on that day.

We introduce the following notations to formulate this problem:

Let x denote weather conditions.

$$y = \begin{cases} 1 & \text{if rain} \\ 0 & \text{if not rain} \end{cases}$$

We assume (x, y) pairs are random according to source distribution D, so $(x, y) \sim D$. We want to estimate

$$p(x) = Pr[y = 1|x] = E(y|x). \tag{18}$$

But the question is that we can never observe $p(x)$. Alice is giving $h_A(x)$ as an estimation of $p(x)$. Bob is also giving his estimation $h_B(x)$. We can only observe $\{x, h_A(x), h_B(x), y\}$ and we want to choose one from $\{h_A(x), h_B(x)\}$ that is closer to $p(x)$. We'll discuss this in more detail next time.