COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Joe Wenjie Jiang

Lecture #20
April 21, 2008

1 Motivation: log loss in online learning.

In the last lecture, we started to discuss the following model of online learning, in which our goal is to minimize the total log loss:

Let X be the space of all possible outcomes in any time step. There are N experts from whom we can consult. At each time step $t = 1, \dots, T$:

- Each expert i predicts $p_{t,i}$, which is a distribution over all possible outcomes X.
- The master algorithm combines $p_{t,i}$ from all experts into a prediction q_t, which is a distribution over X.
- Observe $x_t \in X$.
- We suffer a loss as $-\ln q_t(x_t)$.

Our goal is to come up with a master algorithm which has the following property:

$$-\sum_{t=1}^{T} \ln q_t(x_t) \le \min_i \Big[-\sum_t \ln p_{t,i}(x_t) \Big] + [\text{small amount}]. \tag{1}$$

Take the case of horse racing as an example, we can think of each expert predicting the probability that each horse will win. Another way to motivate this model is to consider coding theory. Suppose Alice wants to send Bob a file consisting of messages $x_1, \dots, x_t, \dots x_T$, where $x_t \in X$, the set of all possible messages. Alice can wait until she sees all messages, which is an offline approach. Now we want to find an online method.

Consider there are N coding methods (experts). We want to combine them into a compression algorithm that is as nearly optimal as possible for our file. Note that we don't have any statistical assumption on the distribution of x_t, i.e., they can be arbitrary.

As we've discussed in defining relative entropy, if $p(x)$ is the probability of x chosen from X, then the optimal code length, i.e., the number of bits, is $-\lg p(x)$ for message x . If the compression algorithm (expert) i believes $p_{t,i}$ is the distribution of x_t over X, $-\lg p_{t,i}(x_t)$ is the number of bits used by method i to encode x_t. Similarly, our master algorithm uses $-\lg q_t(x_t)$ number of bits to encode x_t.

The total number of bits used by method i is $-\sum_t \lg p_{t,i}(x_t)$, and the total number of bits used by our master algorithm is $-\sum_t \lg q_t(x_t)$. Our goal is to construct q_t such that the length of our coding scheme is at most the length by the *best* coding scheme plus some small amount, as described in Eq. (1). Note that the master algorithm should be a deterministic algorithm, so Bob can correctly decode the message in the same way as Alice encodes it. This approach has a merit that it has no statistical assumption on the distribution of X, and can perform almost as well as the best coding scheme. Hence, it is called "universal compression".

2 Bayes Algorithm

Although we have no statistical assumption on x_t, in order to derive the process of constructing q_t, we pretend that x_t's are generated by some random process. We can make this assumption because the relevant data are not actually random, which allows us to perform probabilistic calculation on them. From here, we introduce the following notations, e.g., we write $p_{t,i}(x_t) = p_i(x_t|x_1^{t-1})$ and $q_t(x_t) = q(x_t|x_1^{t-1})$, where x_1^{t-1} denotes the sequence $x_1, \ldots, x_{t-1}$. This allows us to think of $p_{t,i}(x_t)$ and $q_t(x_t)$ as the conditional probability of x_t, given the occurrence of the events $x_1, \ldots, x_{t-1}$.

Now pretend that we know the probability distribution of X, and x_t is generated in the following probabilistic process. First, one expert i^* is chosen uniformly at random among all experts, i.e., $\Pr[i^* = i] = \frac{1}{N}$. Then, we generate the sequence $x_1, x_2, \ldots, x_T$ according to p_{i^*}:

$$\Pr[x_t|x_1^{t-1},\ i^* = i] = p_i(x_t|x_1^{t-1}). \tag{2}$$

Then, q_t can be constructed in the following manner. Let $q(x_t|x_1^{t-1}) = \Pr[x_t|x_1^{t-1}]$. By marginalization, we have

$$\begin{aligned} q(x_t|x_1^{t-1}) &= \Pr[x_t|x_1^{t-1}] \\ &= \sum_{i=1}^{N} \Pr[i^* = i|x_1^{t-1}] \cdot \Pr[x_t|x_1^{t-1}, i^* = i] \\ &= \sum_{i=1}^{N} \Pr[i^* = i|x_1^{t-1}] \cdot p_i(x_t|x_1^{t-1}). \end{aligned}$$

We introduce a new notation,

$$w_{t,i} = \Pr[i^* = i|x_1^{t-1}].$$

We have, $w_{1,i} = \Pr[i^* = i|x_1^0] = \Pr[i^* = i] = \frac{1}{N}$. Note that $w_{t+1,i} = \Pr[i^* = i|x_1^t]$ is called *posterior*, since we are predicting the probability of choosing expert i after we see the sequence of events. By Bayes rule,

$$\begin{aligned} w_{t+1,i} &= \Pr[i^* = i|x_1^t] \\ &= \frac{\Pr[i^* = i] \cdot \Pr[x_1^t|i^* = i]}{\Pr[x_1^t]}. \end{aligned}$$

Here, $\Pr[i^* = i]$ is called *prior*, and $\Pr[x_1^t|i^* = i]$ is the *likelihood* of the sequence. Since the denominator $\Pr[x_1^t]$ is a constant independent of i, we can further expand the equation by

$$\begin{aligned} w_{t+1,i} &\propto \Pr[i^* = i] \cdot \Pr[x_1^t|i^* = i] \\ &= \Pr[i^* = i] \cdot \Pr[x_1^{t-1}|i^* = i] \cdot \Pr[x_t|x_1^{t-1},\ i^* = i] \\ &= \Pr[i^* = i|x_1^{t-1}] \cdot \Pr[x_1^{t-1}] \cdot p_i(x_t|x_1^{t-1}) \\ &\propto w_{t,i} \cdot p_i(x_t|x_1^{t-1}). \end{aligned}$$

This suggests that we can recursively compute $w_{t+1,i}$ as

$$w_{t+1,i} = \frac{w_{t,i} \cdot p_i(x_t|x_1^{t-1})}{\sum_i w_{t,i} \cdot p_i(x_t|x_1^{t-1})}. \tag{3}$$

In summary, we can calculate $q_t(x_t)$ as

$$q_t(x_t) \quad = \quad \sum_{i=1}^{N} w_{t,i} \cdot p_i(x_t|x_1^{t-1}). \tag{4}$$

where $w_{t,i}$ is updated the way as (3) suggests.

This construction is known as the *Bayes algorithm*. If we view Bayes algorithm as an algorithm for updating the weights $w_{t,i}$ of N experts, the update rule is

$$w_{t+1,i} \quad = \quad \frac{w_{t,i} \cdot p_i(x_t|x_1^{t-1})}{\text{normalization}}.$$

An interesting question is how does Bayes algorithm relate to online learning with expert advice, as we have seen before? We can think of the update rule for the weights as

$$w_{t+1,i} \quad \propto \quad w_{t,i} \cdot \beta^{\text{loss}(i)}. \tag{5}$$

where β is a constant and loss(i) is 1 if expert i made a mistake and 0 otherwise. In the log loss model, $\text{loss}(i) = -\ln p_i(x_t|x_1^{t-1})$ and $\beta = e^{-1}$, which yields the same update rule as (3). In other words, Bayes algorithm is essentially the same form as the earlier algorithm.

3 Analysis of Bayes Algorithm

To prove that Bayes algorithm achieves the goal we previously stated in (1), we first extend the analogy to the probability of an entire sequence by defining

$$\begin{aligned} q(x_1^T) &= q(x_1)q(x_2|x_1)q(x_3|x_1,x_2)\dots \\ &= \prod_{t=1}^{T} q(x_t|x_1^{t-1}) \\ &= \prod_{t=1}^{T} \Pr[x_t|x_1^{t-1}] \\ &= \Pr[x_1^T]. \end{aligned}$$

which is the probability of the entire sequence. Similarly, $p_i(x_1^T)$ can be defined so that

$$p_i(x_1^T) = \Pr[x_1^T|i^* = i].$$

Now we can rewrite the cumulative log loss of our master algorithm as:

$$-\sum_{t=1}^{T} \ln q_t(x_t) = -\sum_{t=1}^{T} \ln q(x_t|x_1^{t-1}) = -\ln \prod_{t=1}^{T} q(x_t|x_1^{t-1}) = -\ln q(x_1^T).$$

And the cumulative log loss of expert i is:

$$-\sum_{t=1}^{T} \ln p_t(x_t) = -\sum_{t=1}^{T} \ln p_i(x_t|x_1^{t-1}) = -\ln \prod_{t=1}^{T} p_i(x_t|x_1^{t-1}) = -\ln p_i(x_1^T).$$

To bound the cumulative log loss of the master algorithm, note that

$$\begin{aligned} q(x_1^T) &= \Pr[x_1^T] \\ &= \sum_{i=1}^{N} \Pr[i^* = i] \cdot \Pr[x_1^T | i^* = i] \\ &= \sum_{i=1}^{N} \frac{1}{N} \cdot \Pr[x_1^T | i^* = i] \\ &= \frac{1}{N} \sum_{i=1}^{N} p_i(x_1^T). \end{aligned}$$

Hence, we have

$$\begin{aligned} -\ln q(x_1^T) &= -\ln\left(\frac{1}{N}\sum_{i=1}^{T} p_i(x_1^T)\right) \\ &\le -\ln \frac{p_i(x_1^T)}{N} \\ &= -\ln p_i(x_1^T) + \ln N. \end{aligned}$$

Since the above inequality holds for any i,

$$\begin{aligned} -\sum_{t=1}^{T} \ln q_t(x_t) &= -\ln q(x_1^T) \\ &\le \min_i \Big(-\ln p_i(x_1^T) \Big) + \ln N \\ &= \min_i \left(-\sum_{t=1}^{T} \ln p_{t,i}(x_t) \right) + \ln N \end{aligned}$$

which shows that the master algorithm does no worse than the best expert plus $\ln N$.

What is worth noting is that, in the context of universal compression, the same bound can be achieved. For the obvious off-line algorithm, the number of bits needed to indicate which coding scheme is the best is $\lg N$, and the file can then be encoded in the best way. Bayes algorithm achieves this identical bound while coding the stream of messages entirely online.

To extend this model, we can set up the algorithm such that we choose $\Pr[i^* = i] = \pi_i$, where π_i is a given prior distribution over the experts. In this case, we can obtain a similar argument in the following form:

$$-\sum_{t=1}^{T} \ln q_t(x_t) \le \min_i \left[-\sum_{t=1}^{T} \ln p_{t,i}(x_t) + \ln \frac{1}{\pi_i} \right].$$

4 Example: experts in the continuous space

Let $X = \{0, 1\}$, and the experts predict the probability $p \in [0, 1]$, such that $\Pr[x = 1] = p$ and $\Pr[x = 0] = 1 - p$. Here, we have an infinite number of experts. We can apply a continuous analogue of Bayes algorithm, where the sum is replaced by integrals. Suppose up to the first t rounds, we have observed h 1's, i.e., $h = \sum_{t'=1}^{t} x_{t'}$. Recall (3), $w_{t,p}$ becomes

$$w_{t,p} = \frac{p^h(1-p)^{t-1-h}dp}{\int_0^1 p^h(1-p)^{t-1-h}dp}.$$

So the continuous version of equation (4) gives us

$$\begin{aligned} q_{t+1}(1) &= \frac{\int_0^1 p^h(1-p)^{t-h} \cdot p\ dp}{\int_0^1 p^h(1-p)^{t-h}dp} \\ &= \frac{h+1}{t+1} \\ &= \frac{\#\text{ of 1's} + 1}{\#\text{ of 1's} + \#\text{ of 0's} + 2}. \end{aligned}$$

This is called *Laplace smoothing*. Note its difference to the naive Bayes algorithm, in which this quantity is $\frac{h}{t}$ by applying the maximum likelihood estimation.

5 Switching experts

In the next class, we will consider the case of switching experts, i.e., the distribution of X varies over the timeline $t = 1, 2, \ldots, T$, and one particular expert is doing better than others in a certain period of time. There are in total k number of switches such that each switch corresponds to the entering of another period in which a new expert does a good job. Our goal is to find out a master algorithm which performs almost as well as a sequence of best experts, plus some small amount. However, the previous theorem cannot be directly applied, since there is a large number of "experts" as in the original definition, each of which corresponds to a sequence of $k+1$ experts in this case. In fact, our algorithm should compete against the best sequence of experts, assuming k is not too large.