COS 511: Foundations of Machine Learning

Rob Schapire
Scribe: Atoshi Chowdhury

Lecture #20
25 April 2006

1 Log loss in online learning: motivation

Last time, we began discussing the following model of online learning, in which the goal was to minimize the log loss:

Let X be the space of possible outcomes in any time step, and suppose we have N experts to consult. During each time step $t = 1, ..., T$,

- each expert i provides a predicted distribution $p_{t,i}$ over X;
- the master algorithm combines the distributions $p_{t,i}$ into a distribution q_t over X;
- we observe the realization of an event $x_t \in X$;
- we suffer a loss recorded as $-\ln q_t(x_t)$.

We motivated this model using the example of a horse race, in which each expert's prediction gives the probability that each horse will win. Another motivation comes from coding theory: the problem of universal compression. Suppose we want to encode a file consisting of characters x_t (where $t = 1, ..., T$) taken from the space X of all possible characters, and we have N compression programs ('experts') which we'd like to combine into a compression algorithm that is as nearly optimal as possible for our file. Note that this motivation is natural because we don't want to make any statistical assumptions about the characters x_t in the file; they are completely arbitrary.

As we've discussed before, if $p(x)$ is the probability of choosing x from X, then the optimal number of bits to use is $-\lg p(x)$. So we can let the ith compression program encode x_t using $-\lg p_{t,i}(x_t)$ bits, and have the master combine this information to encode x_t in $-\lg q_t(x_t)$ bits.

The total length of the file after compression by the master is $-\sum_t \lg q_t(x_t)$; the total length after compression by the ith expert is $-\sum_t \lg p_{t,i}(x_t)$. Our problem is to construct q_t so that the former length is at most the length after compression by the best compression program, plus some small amount (i.e., so that the loss of the master is at most the loss of the best expert): we want q_t to satisfy

$$-\sum_t \lg q_t(x_t) \le \min_i \left(-\sum_t \lg p_{t,i}(x_t) \right) + \text{ [small amount]}.$$

Up to a constant factor, this is the same problem we were trying to solve in the example of the horse race.

2 Bayes' algorithm

Although we don't make any statistical assumptions on the x_t, in order to derive the algorithm constructing q_t we'll *pretend* the x_t are generated by a random process (to be described below). (We can do this because while the relevant data are not actually random, they have the properties that allow us to perform probabilistic calculations on them.) To that end, we introduce the following suggestive notation: write $p_{t,i}(x_t) = p_i(x_t \mid x_1^{t-1})$ and $q_t(x_t) = q(x_t \mid x_1^{t-1})$, where x_1^{t-1} denotes the sequence $x_1, ..., x_{t-1}$. This lets us think of $p_{t,i}(x_t)$ and $q_t(x_t)$ as the probability of the event x_t conditional upon the occurrence of the events $x_1, ..., x_{t-1}$.

Now, we pretend the x_t are generated as follows. First, one expert i^* is chosen uniformly at random from among all the experts: $\Pr[i^* = i] = \frac{1}{N}$. Then the sequence $x_1, ..., x_T$ is generated according to p_{i^*}:

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

To construct q_t, we simply let $q_t(x_t) = \Pr[x_t \mid x_1^{t-1}]$. This quantity can be computed by marginalization:

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

By Bayes' rule,

$$\begin{aligned}
\Pr[i^* = i \mid x_1^{t-1}] &= \frac{\Pr[x_1^{t-1} \mid i^* = i] \Pr[i^* = i]}{\Pr[x_1^{t-1}]} \\
&= \frac{\frac{1}{N} \Pr[x_1^{t-1} \mid i^* = i]}{\Pr[x_1^{t-1}]} \\
&= \frac{\frac{1}{N} \prod_{t'=1}^{t-1} \Pr[x_{t'} \mid i^* = i, x_1^{t'-1}]}{\Pr[x_1^{t-1}]} \\
&= \frac{\frac{1}{N} \prod_{t'=1}^{t-1} p_i(x_{t'} \mid x_1^{t'-1})}{\Pr[x_1^{t-1}]}.
\end{aligned}$$

Let $w_{t,i} = \displaystyle\prod_{t'=1}^{t-1} p_i(x_{t'} \mid x_1^{t'-1})$; then

$$\Pr[i^* = i \mid x_1^{t-1}] = \frac{1}{N} \frac{w_{t,i}}{\Pr[x_1^{t-1}]},$$

and

$$\begin{aligned}
\Pr[x_1^{t-1}] &= \sum_i \Pr[x_1^{t-1} \mid i^* = i] \Pr[i^* = i] \\
&= \frac{1}{N} \sum_i \prod_{t'=1}^{t-1} \Pr[x_{t'} \mid x_1^{t'-1}, i^* = i] \\
&= \frac{1}{N} \sum_i \prod_{t'=1}^{t-1} p_i(x_{t'}|x_1^{t'-1}) \\
&= \frac{1}{N} \sum_i w_{t,i}.
\end{aligned}$$

Substituting these expressions in equation 1 yields

$$q_t(x_t) = \frac{\sum_i w_{t,i} p_i(x_t \mid x_1^{t-1})}{\sum_i w_{t,i}}. \tag{2}$$

This construction is known as Bayes' algorithm. From a probabilistic point of view, it carries the following terminology: $\Pr[i^* = i]$ is called the *prior*, $\Pr[i^* = i \mid x_1^{t-1}]$ is the *posterior*, and $\Pr[x_1^{t-1} \mid i^* = i]$ is the *likelihood*.

If we view Bayes' algorithm as an algorithm for updating the weights $w_{t,i}$ of the N experts (before taking a weighted average of their predictions at each time step), the update rule for the weights is $w_{t+1,i} = w_{t,i} p_i(x_t \mid x_1^{t-1})$.

How does Bayes' algorithm compare to the algorithm we used in the original scenario of learning with expert advice? There, the update rule for the weights was $w_{t+1,i} = w_{t,i}\beta^{\mathrm{loss}(i)}$, where β was a constant and $\mathrm{loss}(i)$ was 1 if expert i had made a mistake and 0 otherwise. In the log loss model, we have $\mathrm{loss}(i) = -\ln p_i(x_t \mid x_1^{t-1})$; taking $\beta = e^{-1}$ yields the same update rule. So Bayes' algorithm in fact has the same form as the earlier algorithm.

3 Analysis of Bayes' algorithm

To prove that Bayes' algorithm yields an effective learner, we first extend the analogy to conditional probability by defining

$$p_i(x_1^t) = \Pr[x_1^t \mid i^* = i] = \prod_{t'=1}^{t} p_i(x_{t'} \mid x_1^{t'-1}),$$
$$q(x_1^t) = \Pr[x_1^t] = \prod_{t'=1}^{t} q(x_{t'} \mid x_1^{t'-1}).$$

Now the cumulative log loss of the master algorithm can be written

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

Similarly, expert i's cumulative log loss is

$$-\sum_{t=1}^{T} \ln p_{t,i}(x_t) = -\sum_{t=1}^{T} \ln p_i(x_t \mid x_1^{t-1}) = -\ln \prod_{t=1}^{T} p_i(x_t \mid x_1^{t-1}) = -\ln p_i(x_1^T).$$

To bound the master's cumulative log loss, note that for any i,

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

since each $p_i(x_1^T)$ is nonnegative. Hence

$$-\ln q(x_1^T) \leq -\ln p_i(x_1^T) + \ln N.$$

In particular,

$$-\ln q(x_1^T) \leq \min_i \left(-\ln p_i(x_1^T)\right) + \ln N,$$

or equivalently,

$$-\sum_t \ln q_t(x_t) \leq \min_i \left(-\sum_t \ln p_{t,i}(x_t)\right) + \ln N,$$

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

(In the context of universal compression, we could achieve the same bound by using $\lg N$ bits to say which expert is best, then encode the file in the same way as the best expert. But Bayes' algorithm gives us an effective way to do this with all coding occurring online.)

We can also set up this algorithm choosing $\Pr[i^* = i] = \pi_i$ where the π_i's form a given prior probability distribution over the experts. In this case, by a direct generalization of the argument above, the bound on the loss of the master will be

$$-\sum_t \ln q_t(x_t) \leq \min_i \left(-\sum_t \ln p_{t,i}(x_t) - \ln \pi_i\right).$$

4 Example: a continuum of experts

Let $X = \{0,1\}$, and suppose we have a continuum $[0,1]$ of experts: for each $p \in [0,1]$, there is an expert p that predicts $\Pr[1] = p$, $\Pr[0] = 1-p$ at every time step. Then we can apply a continuous analogue of Bayes' algorithm, with the sums replaced by integrals. Suppose that h of the first t bits observed were 1; then for each expert p, we have $w_{t+1,p} = p^h(1-p)^{t-h}$, so the continuous version of equation (2) gives us

$$q_{t+1}(1) = \frac{\int_0^1 p^h(1-p)^{t-h} \cdot p \; dp}{\int_0^1 p^h(1-p)^{t-h} \; dp} = \frac{h+1}{t+2}.$$

Note that this is not the probability we would assign to the outcome 1 based on naïve counting (that would be $\frac{h}{t}$); instead, it's what we would get by applying Laplace smoothing.