COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Nicolas Crowell

Lecture 22
April 24, 2014

1 Bayes' algorithm review

Last lecture, we derived an on-line algorithm for estimating a probability distribution for a sequence of elements. We developed the algorithm as a general framework that can incorporate and choose among distinct individual methods for estimating distributions, so as to take into account the diversity of problem-specific methods that exist for estimating probability distributions in particular settings.

To repeat the formal problem setting: we are given a set of methods for estimating a probability distribution over a space. We often call these methods "experts". We observe a sequence of elements from the space (which may be chosen adversarially), and estimate on-line a probability distribution based on everything we've seen so far. Our goal is to develop an on-line algorithm that estimates a distribution by incorporating the outputs of the predetermined set of estimation methods. We will want the algorithm to perform almost as well as the best *single* expert on the observed sequence of elements, with a corresponding theoretical bound proving this "almost as good" comparison.

In this setting, we will measure performance by minimizing *log loss*. Given a probability distribution and an observed element, we define loss to be the negative of the logarithm of the probability of the observed element under the probability distribution, i.e. $-\log p(x)$.

With these goals in mind, we derived the Bayes algorithm, together with the following performance comparison bound bound:

$$-\sum_{t=1}^{T} \ln q_t(x_t) \leq \left(-\sum_{t=1}^{T} \ln p_{i,t}(x_t) \right) + \ln N, \ \forall \text{ experts } i \tag{1}$$

where q_t is the probability distribution chosen by Bayes' algorithm at time t, $p_{i,t}$ is the probability distribution chosen by expert i at time t, and N is the number of experts. We also showed that the $\ln N$ term corresponded to the special case of a slightly more general derivation, where instead of assuming a uniform distribution over the experts at the beginning of the derivation, we use a different distribution that incorporates prior knowledge about the different choices of expert, where π_i is the new prior probability of expert i. The more general bound was then the following:

$$-\sum_{t=1}^{T} \ln q_t(x_t) \leq \left(-\sum_{t=1}^{T} \ln p_{i,t}(x_t) \right) - \ln \pi_i, \ \forall \text{ experts } i \tag{2}$$

Switching experts

A more general problem

Now we consider a slightly more general problem: what if different methods of estimating the distribution perform better during different parts of the sequence? For example, for

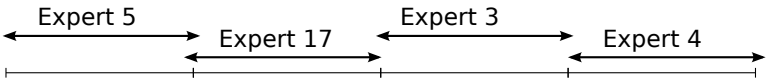

Figure 1: An observation sequence generated by 4 experts respectively choosing probability distributions at different parts of the sequence.

the first 83 time steps of the on-line sequence, expert 1 might have the best guesses, while expert 2 might be more accurate during the next 47 observations.

In the derivation of the bound for Bayes' algorithm, we pretended that the data was generated randomly according to some expert's distribution at every time step, and the expert whose distribution was followed was chosen randomly at the *beginning* of the process. Now we change the model by further pretending that at arbitrary times t, the chosen expert is *switched* to a different expert. In this new setting, algorithms that do comparably well to the best *single* expert are no longer satisfying: we want to do almost as well as the best *switching sequence* of experts.

Our original motivating example from coding theory hypothesized that someone wanted to send a message using as few bits as possible, so they needed to find an encoding that would take advantage of the implicit probability distribution of the possible elements of the message to minimize the number of bits used to encode the message. In this new setting, we can imagine that the content of the message takes several different forms. For example, the first part of the message might be written in English, and the second part of the message might be written in Spanish, which presumably have different distributions over characters. Generalizing the example even further, the rest of the message might switch between sending pictures, and then maybe music, or even completely random bits. The point is that these different kinds of message content will likely have very different distributions of elements, so we will want to switch between different methods of encoding in order to take advantage of each distribution separately.

A strategy for solving the switching experts problem

So we want to do almost as well as the best switching sequence of experts, and we already have a way to do almost as well as a single best expert. Our idea for an algorithm to address this new problem setting will be to instead consider "meta-experts" representing a switching sequence of *base* experts. A meta-expert makes predictions based on which base expert it considers to be the "true" expert at a given time. Specifically, we can have a meta-expert for *every* switching sequence with k switches. Then we could just apply Bayes' algorithm to this huge family of meta-experts built from the N base experts. We can count these meta experts:

$$M \triangleq \#\text{ of meta experts} = N^{k+1}\binom{T+k}{k}$$

And if we want to plug this size into our error term in the original bound for Bayes' algorithm with a uniform prior, we just need to compute $\ln M$:

$$\ln M \approx \ln N + k\ln N + k\ln(T/k) \approx \ln N + [k(\ln N + \ln T)] \tag{3}$$

The inner term $(\ln N + \ln T)$, which is multiplied by the number of switches k, conceptually corresponds to paying a price in loss for every time a meta-expert switches base experts, in addition to the original cost of Bayes' algorithm.

Making meta-experts tractable

The problem with this approach is that the number of meta-experts is exponential in N. We need a computational trick that will allow us to consider the very large space of meta-experts without computationally interacting with each one directly.

Another inelegant part of our meta-expert idea as stated is that it uses an external parameter, namely the precise number of times that the base expert switches. To deal with both this rather arbitrary parameter and the computational problem posed by the large number of meta-experts, we will take advantage of the flexibility of Bayes' algorithm, which works for *any* prior distribution over the space of experts, along with a clever choice of prior distribution.

Let's think of a meta-expert as a vector $\mathbf{e}$ of T elements representing which base expert is predicting at time t.

$$e_t \in \{1, \ldots, N\} \quad \mathbf{e} \in \mathbb{M} \triangleq \{1, \ldots, N\}^T$$

Every meta-expert corresponds to a vector in the space $\mathbb{M}$. Notice that we've represented *every* possible meta expert, including one that switches at every time step! This seems undesirable, as such a meta-expert represents a very complicated way of generating data. Reflecting our general preference in machine learning for simpler hypotheses, we will want to focus on "nice" blocky sequences, where the meta-expert doesn't switch very often. To embody this preference, we can weight the prior distribution over meta-experts to give higher weight to meta-experts with fewer switches.

To find such a prior, we will come up with a random process to generate meta-experts in such a way as to embody the simplicity preference. The random process will then exactly define our prior distribution over the meta-experts, and we can write:

$$\Pr[\mathbf{e}^* = \mathbf{e}] = \pi(\mathbf{e})$$

where $\pi(\mathbf{e})$ is the probability of a meta-expert under the random process, and $\mathbf{e}^*$ is the chosen "true" meta-expert.

A prior distribution over meta-experts

The basic heuristic we want to follow is that a higher number of switches should correspond to a lower probability. Our random process will work by generating one component of a vector representing a meta-expert at a time. Let the initial component be chosen uniformly at random.

$$e_1^* = \text{uniform} \implies \Pr[e_1^* = i] = 1/N$$

And the key step is how to find component $t+1$ in accordance with our heuristic, which is that we want consecutive components to be the same with relatively high probability:

$$e_{t+1}^* = \begin{cases} e_t^* \text{ with probability } 1 - \alpha \\ \text{some other expert chosen uniformly, with probability } \alpha \end{cases}$$

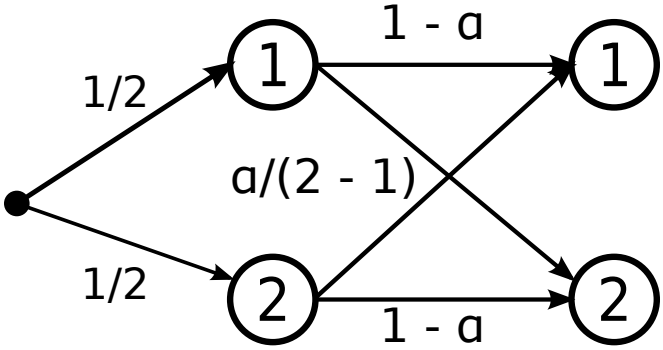

Figure 2: We randomly generate the generating meta-expert via a Markov chain of base experts.

Given this, we can compute the corresponding conditional probability that some base expert is the true expert at time $t+1$ given the identity of the true expert at time t:

$$\Pr[e^*_{t+1} \mid e^*_t] = \begin{cases} 1-\alpha & \text{if } e^*_{t+1} = e^*_t \\ \alpha/(N-1) & \text{otherwise} \end{cases}$$

This means that we defined a Markov chain of base experts, which we illustrate in Figure (2).

So meta-experts are defined by a path in the Markov sequence graph, and their probability is just the product of the edge probabilities. This defines our prior distribution on meta-experts, so we can now analyze our original bound.

Analyzing loss of meta-expert algorithm

Say that some meta-expert $\mathbf{e}$ has k switches. Then we can compute the prior probability that it is the generating expert, and thus compute the error term in our original bound on the error of Bayes' algorithm compared to any expert, including the best one:

$$\begin{aligned} -\ln(\pi(\mathbf{e})) &= -\ln\left[(1/N)\left(\frac{\alpha}{N-1}\right)^k (1-\alpha)^{T-1-k}\right] \\ &= \ln N + k\ln\frac{N-1}{k} + (T-k-1)\ln(1/(1-\alpha)) \end{aligned}$$

If we "approximately minimize" this with respect to α, we find $\alpha = k/(T-1)$. Then we have:

$$-\ln(\pi(\mathbf{e})) \approx \ln N + k(\ln N + \ln T)$$

which is (roughly) the same bound as Equation (3), which was for the space of all meta-experts with exactly k experts. It's a nice result that in moving from considering only experts with precisely k switches between base experts to those with any number of switches, we can still bound the loss in terms of the number of switches made by the generating expert!

Efficient implementation of meta-expert algorithm

The number of meta-experts with *any* number of switches is much larger than just those with k switches for some k, so we need to find a way of dealing with all of them at once without performing computation directly proportional to the population size.

Recall that at every time step of Bayes' algorithm, the "master" receives probability distributions from each expert and combines them into a single distribution before observing the next data point. The combined distribution is produced by weighting each expert's distribution and normalizing. The weights are updated every round by multiplying each expert's old weight by the probability of the observed data point under its chosen distribution for that round.

Before diving into how Bayes' algorithm will work with meta-experts, we define some notation and make explicit what is signified in the terms of the problem by a few specific random events.

$$x_1^t \triangleq \langle x_1, \ldots, x_t \rangle \quad \text{the sequence of observations up to time } t \tag{4}$$

$$\Pr[\mathbf{e}^* = \mathbf{e}] = \pi(\mathbf{e}) \quad \text{the prior probabilty that } \mathbf{e} \text{ is the generating meta-expert} \tag{5}$$

$$\Pr[x_t | x_1^{t-1}, \mathbf{e}^* = \mathbf{e}] \quad \text{the prediction of } \mathbf{e} \text{ on } x_t, \text{ i.e. the prediction of base expert } \mathbf{e}_t \tag{6}$$

$$\Pr[x_t | x_1^{t-1}, e_t^* = i] \triangleq p_i(x_t | x_1^{t-1}) \tag{7}$$
(prediction of base expert selected by generating meta-expert at time t)

Also recall that when we derived/analyzed Bayes' algorithm, we pretended that one of the experts is chosen at the beginning of the observation sequence by a random process, and that at every time step the observed data point is generated according to the generating expert, which was randomly chosen to begin with. We also define the probability distribution produced by the master to be the probability of the next data point under this pretend random process given all of the observed data so far, which is implicitly marginalized with respect to the experts, one of which was randomly chosen to be the "generating" expert.

$$\begin{aligned} q(x_t \mid x_1^{t-1}) &= \Pr[x_t \mid x_1^{t-1}] \\ &= \sum_{i=1}^{N} \Pr[x_t, e_t^* = i \mid x_1^{t-1}] && \text{(marginalization)} \\ &= \underbrace{\Pr[e_t^* = i \mid x_1^{t-1}]}_{v_{t,i}} \cdot \underbrace{\Pr[x_t \mid e_t^* = i, x_1^{t-1}]}_{p_i(x_t \mid x_1^{t-1})} && \text{(defn. of conditional probability)} \end{aligned}$$

We've separated the two quantities above to deal with them separately below. The probability $v_{t,i}$ is the probability of the event that the t-th generating expert is base expert i, given all of the observations through x_{t-1}, which can also be described as the distribution of base experts at time t given the previous observations. The probability $p_i(x_t \mid x_1^{t-1})$ is the distribution of observations at time t, given all of the previous observations *and* the identity of the generating base expert at time t.

First, we examine $v_{t,i}$. At the first time step, we have $v_{1,i} = \Pr[e_1^* = i] = 1/N$. For time step $t+1$, we will first explicitly marginalize with respect to the experts at the previous

time step.

$$\forall i,\ v_{t+1,i} = \Pr[e^*_{t+1} = i \mid x_1^t] = \sum_{j=1}^{N} \Pr[e^*_{t+1} = i, e^*_t = j \mid x_1^t]$$
$$= \sum_{j=1}^{N} \underset{(B)}{\Pr[e^*_{t+1} = i \mid e^*_t = j,\ x_1^t]} \cdot \underset{(A)}{\Pr[e^*_t = j \mid x_1^t]} \tag{8}$$

Note that (B) is defined directly by the Markovian process we defined early:

$$\Pr[e^*_{t+1} = i \mid e^*_t = j, x_1^t] = \Pr[e^*_{t+1} = i \mid e^*_t = j] = \begin{cases} 1-\alpha \text{ when } i = j \\ \alpha/(N-1) \text{ otherwise} \end{cases}$$

Diving into (A):

$$\Pr[e^*_t = j \mid x_t, x_1^{t-1}] = \frac{\Pr[x_t \mid e^*_t = j, x_1^{t-1}] \cdot \Pr[e^*_t = j, x_1^{t-1}]}{\Pr[x_t \mid x_1^{t-1}]} \qquad \text{(Bayes' rule)}$$
$$= \frac{v_{t,j} \cdot p_j(x_t \mid x_1^{t-1})}{q(x_t \mid x_1^{t-1})} \tag{9}$$

Given these simplifications for the quantities (A) and (B), we can write a computable expression for $v_{t+1,i}$

$$v_{t+1,i} = \sum_j \frac{v_{t,j} \cdot p_j(x_t \mid x_1^{t-1})}{q_t(x_t \mid x_1^{t-1})} \cdot \begin{cases} 1-\alpha \text{ when } i = j \\ \alpha/(N-1) \text{ otherwise} \end{cases} \tag{10}$$

So we see that $v_{t+1,i}$ can be computed using the sum of N terms from Equation (10). This means we can compute $v_{t+1,i}$ for all $i \in \{1, \ldots, N\}$ in $\mathcal{O}(N^2)$ time. In fact, we can algebraically simplify to compute this in $\mathcal{O}(N)$ time without much more work:

$$v_{t,j} = \Pr[e^*_t = j \mid x_1^{t-1}] = \sum_j \frac{v_{t,j} p_j(x_t \mid x_1^{t-1})}{q_t(x_t \mid x_1^{t-1})} \cdot \begin{cases} 1-\alpha \text{ when } i = j \\ \alpha/(N-1) \text{ otherwise} \end{cases}$$
$$= \sum_j c_j \left[\frac{\alpha}{N-1} + \left(1 - \alpha - \frac{\alpha}{N-1}\right) \cdot \mathbb{1}\{i = j\} \right]$$
$$= \left(1 - \alpha - \frac{\alpha}{N-1}\right) c_i + \frac{\alpha}{N-1} \sum_{j=1}^{N} c_j$$
$$= \frac{\alpha}{N-1} + \left(1 - \alpha - \frac{\alpha}{N-1}\right) c_i \tag{11}$$

where c_i is the quantity (A), from equations (8) and (9), and is the posterior probability of the generating expert being base expert i at time t. This means we can compute the weight update for one of the experts at one time step in constant time, so each time step will only need $\mathcal{O}(N)$ computation to update a set weights over the base experts, even though there are a huge number of meta-experts.

2 Stock investment, game theory

How do you choose to apportion money between different investment instruments? We will set up some notation and a simple mathematical framework to be expanded upon during the next lecture.

There are N stocks. Every "day" (pick your favorite unit of time), you need to decide how to reapportion wealth among the stocks during the morning. At the end of each day t, we find out the price shift for each stock i:

$$p_t(i) = \frac{\text{price at end of day } t}{\text{price at beginning of day } t}$$

For example, if stock i goes up 5% on day t, then $p_t(i) = 1.05$.

Let S_t be our total wealth at the start of day t, and let $S_1 = 1$. Let $w_t(i) \triangleq$ the fraction of our wealth which we choose to invest in stock i on day t. Then our notation allows us to write the amount of our wealth at the beginning of day t invested in a particular stock i as $S_t \cdot w_t(i)$, and the corresponding amount for the end of day t is $S_t \cdot w_t(i) \cdot p_t(i)$.

Our total wealth at the end of day t is the following:

$$S_{t+1} = \sum_{i=1}^{N} S_t w_t(i) p_t(i) = S_t \cdot (\mathbf{w}_t \cdot \mathbf{p}_t)$$

And at the end of T time steps:

$$S_T = S_1 \cdot \prod_{t=1}^{T} (\mathbf{w}_t \cdot \mathbf{p}_t) = \prod_{t=1}^{T} (\mathbf{w}_t \cdot \mathbf{p}_t)$$