COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire — Lecture #21
Scribe: Soner Sevinc — April 23, 2008

1 Review of the Bayes Algorithm

Last time we talked about the Bayes algorithm, in which we give priors π_i to each expert. The algorithm maintains weights $w_{t,i}$ for each expert. The π_i values serve as the initial weights for the experts. Experts predict the distributions $p_{t,i}$ over the same set X, and the algorithm predicts the q_t distribution as a mixture of those distributions. To restate it mathematically:

```
$N$ experts
$\pi_i =$ prior,   $\pi_i \geq 0$,   $\sum_i \pi_i = 1$
$w_{1,i} = \pi_i$
for $t = 1, \ldots T$
    expert $i$ predicts $p_{t,i}$ (distribution over $X$)
    master predicts $q_t$
              $q_t(x) = \sum_{i=1}^{N} w_{t,i} p_{t,i}(x)$
    observe $x_t$
    $\forall i : w_{t+1,i} = \frac{w_{t,i} p_{t,i}(x_t)}{\text{normalization}}$
```

We showed:

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

We derived the algorithm and its analysis by pretending the data was generated by a random process in which

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

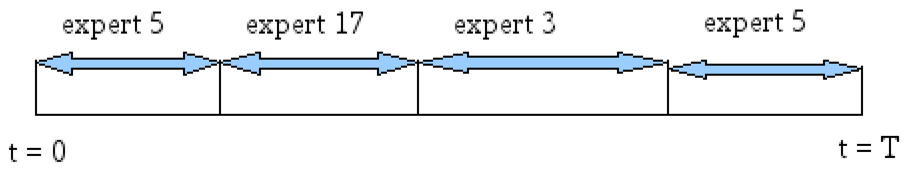
We then defined:

$$q_t(x_t) = q(x_t | x_1^{t-1}) = Pr[x_t | x_1^{t-1}].$$

In the last part, we have shown the upper bound for the loss of the algorithm by pretending that everything is random. Also we note that the algorithm's loss is bounded with respect to the best expert.

2 Switching Experts

Now, we will look at the case where the error is not with respect to a single best expert, but a switching sequence of a subset of the experts.



In the above figure we see that at the beginning expert 5 performs the best, then expert 17, etc. for some number of rounds. We let k denote the number of switches. Such a setting can be interpreted as running several algorithms one after the other because each of them is good at predicting different things. For example, the first algorithm is good for images, the other one is good for text, etc.

We will first look at a quick and dirty way of deriving a bound for the loss with respect to the best switching sequence. The underlying idea in doing this will be regarding each sequence as individual experts.

2.1 Error Bound

The idea is to create a "meta-expert" for every "switching sequence" with k switches. We then apply the Bayes algorithm to the entire family of meta-experts, giving uniform prior π_i to each meta-expert. In the Bayes error bound formula, we see that the error is at most $\ln M$ more than the best expert, where M is the number of experts. How many experts do we have in this case? Base experts can be assigned to the $k+1$ blocks in N^{k+1} many ways. We also take into account the time slots dividing those blocks which can be chosen in $\binom{T-1}{k}$ ways. Hence, the total number of sequences is $N^{k+1}\binom{T-1}{k}$. Thus,

$$\ln M \approx (k+1)\ln N + k\ln(T/k).$$

Ignoring division by k since it is small, we have an additional cost of roughly

$$\ln N + \ln T$$

per switch. This is a nice bound since everything is logarithmic, so that we have additional cost terms only logarithmic in N and T, the number of experts and rounds, respectively. But, the disadvantage is that the implementation of the algortithm is very expensive since we have so many meta-experts. As a next step, we will try to find an efficient way of implementing such an algorithm.

2.2 Weight Share Algorithm

We apply a trick to reduce the computational overhead caused by the large number of experts in the previous algorithm. This trick resembles the kernel trick idea where we had so many dimensions but in some way we find a short path of arriving at the same result. Here instead of dimensions, we have many meta-experts. We define one meta-expert for *every* expert sequence possible rather than ones using just k switches. Thus, the number of meta-experts is N^T. Then we choose a nice prior π in order to get a nice bound.

Every meta-expert corresponds to a unique sequence of experts and in general can be denoted by

$$\mathbf{e} = \langle e_1, \ldots, e_T \rangle$$

where each e_i is the predicting base expert at trial i.

Now, π is the distribution over the sequences. We apply the current parameters to the Bayes algoriothm with the following mapping between the parameters:

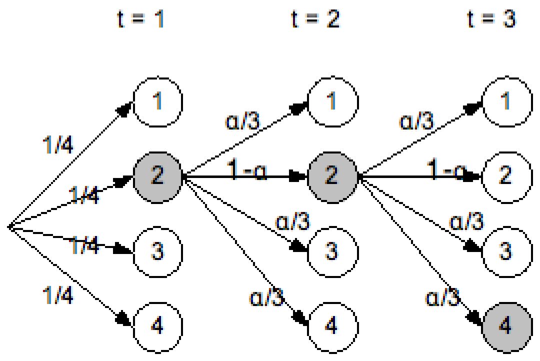
$i \longrightarrow \mathbf{e}$
$i^* \longrightarrow \mathbf{e}^*$
$\pi_i \longrightarrow \pi(\mathbf{e})$

Now we describe the process for choosing $\mathbf{e}^*$.

Random Process for $\mathbf{e}^*$

$$e_1^* : \Pr[e_1^* = i] = 1/N \quad \text{(uniform)}$$
$$e_{t+1} : \Pr[e_{t+1}^* | e_t^*] = \begin{cases} 1-\alpha & \text{if } e_{t+1}^* = e_t^* \\ \frac{\alpha}{N-1} & \text{else.} \end{cases}$$

The above process tells that the base experts have equal probability of being selected in the first round. In any round $t+1$, where $t \geq 1$ the expert that was previously chosen in round t will be selected with probability $1-\alpha$. Otherwise, one of the other experts is chosen uniformly at random. This process can be depicted with the following figure.



In the figure we have four experts, and in the first and second rounds expert 2 is selected. The arrows show the probability of selection in the consecutive rounds. In the third round expert 4 was selected. So, $e_1^* = 2, e_2^* = 2, e_3^* = 4$. If we make the α value small, the sequences with small number of switches will be more likely.

Now let's look at the bound before looking at the computational cost. As before, the additional loss is given in the Bayes bound as $-\ln \pi_i$. Assuming there are k switches in $\mathbf{e}$, we have that

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

The derivation follows from the fact that there are k switches each with probability $\frac{\alpha}{N-1}$, and $T-k-1$ non-switching expert selections with $1-\alpha$ probability. In addition, since we are looking at a fixed $\mathbf{e}$ the switching times are fixed. We optimize the expression and set $\alpha = \frac{k}{T-1}$, and arrive at:

$$= \ln N + k \ln \left[\frac{(N-1)(T-1)}{k} \right] + (\text{some term } \leq k).$$

So, we arrived at almost the same bound as in the previous algorithm.

Now we focus on how to compute this efficiently. When we look at the Bayes algorithm, basically what it does is to calculate the term $\Pr[x_t | x_1^{t-1}]$ at each iteration. Let's look how to compute this term.

$$\begin{aligned} q_t(x_t) &= \Pr[x_t | x_1^{t-1}] \\ &= \sum_{i=1}^{N} \Pr[x_t \wedge e_t^* = i | x_1^{t-1}] \\ &= \sum_{i=1}^{N} \Pr[e_t^* = i | x_i^{t-1}] \Pr[x_t | x_1^{t-1}, e_t^* = i]. \end{aligned}$$

Here we see that x_t and $e_t^* = i$ are disjoint events and their union is the x_t event. Let $v_{t,i} = \Pr[e_t^* = i | x_i^{t-1}]$. This is the probability that base expert i was chosen given what came before. If we computed the $v_{t,i}$ values, then we have an algorithm. How to compute $v_{t,i}$'s ?

$$\begin{aligned} v_{t+1,i} &= \Pr[e_{t+1}^* = i | x_1^t] \\ &= \sum_{j=1}^{N} \Pr[e_{t+1}^* = i \wedge e_t^* = j | x_1^t] \\ &= \sum_{j} \Pr[e_t^* = j | x_1^*] \Pr[e_{t+1}^* = i | e_t^* = j, x_1^t]. \end{aligned}$$

x_1^t can be omitted from the second term since e_{t+1}^* is conditionally independent of x_1^t, given e_t^*, so

$$\Pr[e_{t+1}^* = i | e_t^* = j, x_1^t] = \Pr[e_{t+1}^* = i | e_t^* = j] = \begin{cases} 1 - \alpha & \text{if } i = j \\ \frac{\alpha}{N-1} & \text{else.} \end{cases}$$

For the first term, we can apply Bayes rule:

$$\Pr[e_t^* = j | x_1^*] = \frac{\Pr[x_t | e_t^* = j, x_1^{t-1}] \Pr[e_t^* = j | x_1^{t-1}]}{\Pr[x_t | x_1^{t-1}]}. \tag{1}$$

The first term of the numerator is equal to $p_{t,j}(x_t)$, the second term is equal to $v_{t,j}$, and the denominator is a constant, or $q_t(x_t)$, using the previous notation. Therefore, Eq. (1) is equal to

$$\frac{p_{t,j}(x_t) v_{t,j}}{q_t(x_t)} = c_{t,j}.$$

Thus, the algorithm is

$$q_t(x_t) = \sum_i v_{t,i} p_{t,i}(x_t)$$

for prediction. The weights are updated using the rule, for all $i = 1, \dots N$,

$$v_{t+1,i} = \sum_{j=1}^{N} c_{t,j} \times \begin{cases} 1-\alpha & \text{if } i = j \\ \frac{\alpha}{N-1} & \text{else.} \end{cases}$$

What is the running time of this algorithm per round? Since we have a for loop and a summation from 1 to N, the complexity is $O(N^2)$. This is clearly much better than the previous case where we had experts on the order of $O(N^k)$. We can still improve the running time bound by rewriting the weight updates as

$$S + (1 - \alpha - \frac{\alpha}{N-1})c_{t,i}$$

where $S = \frac{\alpha}{N-1}\sum_{i=1}^{N} c_{t,i}$, and in fact, $S = \frac{\alpha}{N-1}$ since $\sum_{i=1}^{N} c_{t,i} = 1$. That means, for each i, we just make a correction rather than calculating from scratch. The running time is now $O(N)$. This algorithm is called the "weight-share algorithm".

3 Investment

In the last part of the lecture we look at how we can make use of the ideas developed so far in investment. We will basically talk about the stock market. The process with the stock market is that everyday you decide where to put your money, and according to the outcome at the end of the day you re-invest your money to increase the profit. Here we use days as the time metric, but it could also be weeks, months, hours, etc.

We will first look at how we can put the problem mathematically on the table. Next, we will make use of the Bayes algorithm in order to derive bounds on a simple algorithm.

Suppose there are N stocks, and at the beginnig we start investment with \$1. Each day the prices of the stocks change. Our aim is to invest in the stocks that maximize our wealth. On day t, the price of stock i at the end of day t relative to the price at beginning of the day is denoted by $p_t(i)$. This shows the relative value of the stock to the previous day. For example, if stock price has increased by 5%, $p_t(i) = 1.05$, if it decreased by 5%, $p_t(i) = 0.95$. Our wealth at the start of day t is denoted by S_t, and fraction of wealth in stock i at the start of day t is denoted by $w_t(i)$.

Note that the formulation also handles the case where not all of the wealth is invested. This is possible by making one stock return exactly what is input to it.

Therefore, the total wealth in stock i at start of day t becomes $S_t w_t(i)$, and the total wealth in stock i at end of day t is $S_t w_t(i) p_t(i)$. Our total wealth in day $t+1$, therefore can be calculated in terms of wealth in day t by summing up all of the wealth at each stock:

$$S_{t+1} = \sum_i S_t w_t(i) p_t(i) = S_t(\mathbf{w_t} \cdot \mathbf{p_t}).$$

Thus, the wealth at the end of T days is

$$S_{T+1} = \prod_{t=1}^{T} (\mathbf{w_t} \cdot \mathbf{p_t}).$$

We want to choose w_t's to maximize S_{T+1}:

$$\begin{aligned}\max \prod_t (\mathbf{w_t} \cdot \mathbf{p_t}) &\equiv \max \sum_t \ln(\mathbf{w_t} \cdot \mathbf{p_t}) \\ &\equiv \min \underbrace{\sum_t - \ln(\mathbf{w_t} \cdot \mathbf{p_t})}_{\text{loss function in online learning}} \quad .\end{aligned}$$

In the above equations we covert the maximization of wealth problem to minimization problem by taking the negative of the function. Also, taking the logarithm allows us to work with summation rather than multiplication. At the end we come up with a term which is very similar to the loss function in online learning. This is intuitive since actually what the algorithm does is online learning.

The question before determining the error bounds is what are we comparing the algorithm against? Let's for now assume that we are comparing against the best single stock. So, we can make use of the Bayes algorithm.

A weird way of using the Bayes algorithm:

Let $C = \max_{t,i} p_t(i)$
$X = \{0, 1\}$
Let $p_{t,i}(1) = p_t(i)/C \Rightarrow p_{t,i}(0) = 1 - p_{t,i}(1)$
Let $x_t = 1 \quad \forall t$

where $p_{t,i}(1)$ is the probability of expert i selecting 1 on time t.

Let's see why this works out. We apply Bayes algorithm, and invest according to the computed weights so that $w_t(i) = w_{t,i}$. The Bayes algorithm gives us weights $w_{t,i}$, and we invest with the weights $w_t(i)$ which are the same. Then

$$\begin{aligned}q_t(x_t) = q_t(1) &= \sum_i w_{t,i} p_{t,i}(1) \\ &= \sum_i \frac{w_{t,i} p_t(1)}{C} \\ &= \frac{\mathbf{w_t} \cdot \mathbf{p_t}}{C}.\end{aligned}$$

Now we can use the bound in the Bayes algorithm:

$$\begin{aligned}-\sum_t \ln\left(\frac{\mathbf{w_t} \cdot \mathbf{p_t}}{C}\right) = -\sum_t \ln q_t(x_t) &\le \min_i \left[-\sum_t \ln p_{t,i}(x_t)\right] + \ln N \\ &= \min_i \left[-\sum_t \ln\left(\frac{p_t(i)}{C}\right)\right] + \ln N.\end{aligned}$$

The constant C can be removed since $\ln C$ appears as an additive constant on both sides of the inequality. Thus, we have:

$$\underbrace{-\sum_t \ln \mathbf{w_t} \cdot \mathbf{p_t}}_{-\ln(\text{wealth of the alg})} = \underbrace{\min_i \left[-\sum_t \ln p_t(i)\right]}_{-\ln(\text{wealth of the best single stock})} + \ln N$$

Actually it turns out that this bound is trivial. In the next lecture we will see why this is so.