COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Alexander Strzalkowski

Lecture #22
April 24, 2019

1 Review of Bayes Algorithm

Last class we saw an online algorithm to minimize log-loss in the online setting known as Bayes algorithm:

- N: Number of experts
- $\mathcal{X}$: The space of outcomes
- π: A prior distribution over experts with $\pi_i \geq 0$ and $\sum_{i=1}^{N} \pi_i = 1$
- Initialize weights: $w_{1,i} = \pi_i$ for all $i \in \{1, \ldots, N\}$.
- for $t = 1, \ldots, T$:
  - Expert i predicts distribution $p_{t,i}$
  - Learner predicts q_t where $q_t(x) = \sum_{i=1}^{N} w_{t,i} p_{t,i}(x)$
  - Observe $x_t \in \mathcal{X}$
  - $\forall i$ update the weights: $w_{t+1,i} = \frac{w_{t,i} p_{t,i}(x_t)}{q_t(x_t)}$

We were able to show the following regret bound for the algorithm above, namely

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

The analysis for (1) followed from pretending that our data was generated by the following random process:

1. Choose one expert i^* with probability: $Pr[i^* = i] = \pi_i$.
2. x_t is generated probabilistically from the prediction of expert i^* conditioned on the history: $Pr[x_t | x_1^{t-1}, i^* = i] = p_i(x_t | x_1^{t-1}) = p_{t,i}(x_t)$.
3. The learner predicts x_t according to: $q_t(x_t) = q(x_t | x_1^{t-1}) = Pr[x_t | x_1^{t-1}]$.

As a reminder $x_1^{t-1} = \langle x_1, x_2, \ldots, x_{t-1} \rangle$. Specifically, it denotes the vector that holds all observed data points up to round $t-1$. Note that the above framework considers only the advice of a single expert i^* for all T rounds. We next review and extend the case in which it might be better to consider the advice of more than one expert over all T rounds.

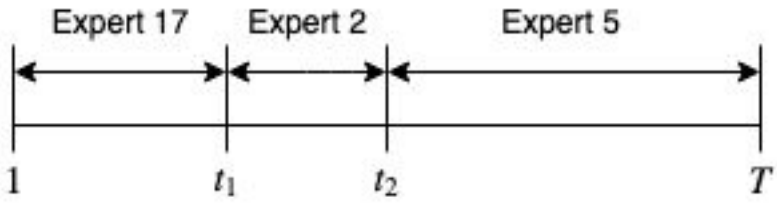

Figure 1: An illustration of why switching experts would be better than choosing one expert for all T rounds.

2 Applying Bayes Algorithm to Switching Experts

As we started exploring last class, what happens if some experts outperform other experts during certain sub-intervals in $[1, T]$? If this were the case, then it would not be ideal to choose a single expert for all T rounds. Intuitively, it makes sense that if we reach round t and another expert performs better than the one's advice we have been using we should *switch* to that other expert.

To be more concrete, consider Figure 1 which shows that expert 17 performs the best from rounds 1 to t_1, while expert 2 performs the best from t_1 to t_2 and expert 5 does the best from t_2 until the last round T. In this example, it would be best to use expert 17's advice for $t \in [1, t_1)$, then *switch* to expert 2's advice for $t \in [t_1, t_2)$, and then *switch* to expert 5's advice for $t \in [t_2, T]$. This example illustrates that we are aiming for an algorithm that performs almost as well as the best "switching sequence" of experts.

This motivates the switching experts framework where we have:

- "base" experts: $i \in \{1, \ldots, N\}$;
- "meta"-experts: $\mathbf{e} = \langle e_1, \ldots, e_T \rangle$, where each $e_t \in \{1, \ldots, N\}$.

Observe that the number of meta-experts is exponential in N, since for each element of a meta-expert we have N different choices for each round t resulting in a total possible selection of N^T meta-experts! It is important to see that a meta-expert is simply a way to keep track of which base experts we would like to use on which round. For example, if $e_5 = 4$, then our meta-expert would use the prediction of base expert 4 on round 5.

We want to use Bayes algorithm on these meta-experts, but with an appropriate prior $\pi(\mathbf{e})$. To define this prior, we describe a "pretend" process for generating a random meta-expert, with higher probability under that prior being given to meta-experts with fewer switches, namely:

1. Choose e_1^* uniformly at random: $Pr[e_1^* = i] = \frac{1}{N}$.
2. Choose e_{t+1}^* according to

$$e_{t+1}^* = \begin{cases} e_t^* & \text{with probability } 1 - \alpha \\ \text{Choose any other } e_i \text{ uniformly} & \text{with probability } \alpha \end{cases}$$

The second step can be more compactly summarized in the following conditional probability:

$$Pr[e_{t+1}^* | e_t^*] = \begin{cases} 1 - \alpha & \text{if } e_{t+1}^* = e_t^* \\ \frac{\alpha}{N-1} & \text{otherwise.} \end{cases} \tag{2}$$

We can now directly apply Bayes algorithm with the aforementioned prior to the switching experts case, and can obtain a regret bound directly from (1) that looks roughly like

$$(\text{loss of algorithm}) \leq (\text{loss of } \mathbf{e}) - \ln \pi(\mathbf{e}),$$

where loss here refers to log-loss and $\pi(\mathbf{e}) = Pr[\mathbf{e}^* = \mathbf{e}]$ denotes the prior probability sampled from the aforementioned process for the meta-expert. In particular, we would like the right-hand side to hold for the best meta-expert in hindsight. If our meta-expert $\mathbf{e}$ switches k times, then the negative logarithm of its prior is given by

$$-\ln \pi(\mathbf{e}) = \ln N + k \ln \frac{N-1}{\alpha} + (T-k-1) \ln \left(\frac{1}{1-\alpha} \right). \tag{3}$$

To see an explicit derivation of (3), see lecture 21 notes. The minimum of (3) is attained at $\alpha^* = \frac{k}{T-1}$. This can be found by differentiating (3) with respect to α, setting the resulting equation to zero and solving for α. Note that this result makes intuitive sense: if we expect that our meta-expert switches k times, then the minimum probability is given by the rate it switches over the whole period excluding the first round since it can't switch then, namely $k/(T-1)$. Plugging α^* into (3) yields

$$-\ln \pi(\mathbf{e}) = \overbrace{\ln N}^{(*)} + \overbrace{k \ln(N-1)}^{(**)} + \overbrace{(T-1) H\left(\frac{k}{T-1} \right)}^{(***)}, \tag{4}$$

where H denotes the entropy function. We originally defined the entropy as a function that takes as input a random variable — not a real number. However, when we take the entropy of a real number we mean it to represent the entropy of a Bernoulli random variable with bias equal to the real number. Concretely, if $b \in \mathbb{R}$ and $X \sim \text{Bernoulli}(b)$, then $H(b) := H(X)$.

If we replaced all of the natural logarithms with logarithms of base 2 in (4), we would be able to interpret each term in the language of coding that was discussed last class. Specifically in (4),

(*) Number of bits needed to encode the expert chosen on the first round.

(**) Number of bits needed to encode switching to another expert ($N-1$ other experts) and this happens k times.

(***) The uncertainty at each of the $T-1$ time points that a switch will occur, each with probability $k/(T-1)$. This is the additional number of bits one needs to specify the switches.

By plugging $-\ln \pi(\mathbf{e})$ directly into inequality (1), the bound derived for Bayes algorithm, we obtain the kind of regret bound we were looking for in the first place, where every switch only incurs an additional regret of about $\ln N + \ln T$. This bound is reasonable if there aren't too many switches. However, if we just naively applied Bayes algorithm to the case of switching experts it would be computationally infeasible. Recall that the number of total possible meta-experts is N^T, and thus our algorithm would have to maintain weights for *each* meta-expert. This would imply that our algorithm runs in time linear in N^T! If there were no structure in the meta-experts' predictions we would be unable to do better. Fortunately, the structure given to our meta-experts with the prior generation process will provide us a means to be able to modify Bayes algorithm for the switching experts case that will run drastically faster.

3 The Weight Share Algorithm

Recall that before when we were dealing with a single best expert our predictions looked like

$$\underline{\text{Before}}\text{: } Pr[x_t|x_1^{t-1},\ i^* = i],$$

but our predictions in the switching experts case now looks like

$$\underline{\text{Now}}\text{: } Pr[x_t|x_1^{t-1},\ \mathbf{e}^* = \mathbf{e}].$$

In the switching experts case, the above notation stands for the prediction of meta-expert $\mathbf{e}$ on x_t. However, recall that a meta-expert's prediction on round t on x_t is given by its corresponding base expert e_t on x_t. Thus, we can reduce the prediction of the switching experts case to

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

The above reduces to exactly what we had before in the single expert case!

We now will rewrite the prediction of the learning algorithm. By marginalizing we have that the prediction of our learner reduces to

$$\begin{aligned} q(x_t|x_1^{t-1}) &= Pr[x_t|x_1^{t-1}] && (5)\\ &= \sum_{i=1}^{N} \underbrace{Pr[e_t^* = i|x_1^{t-1}]}_{v_{t,i}} \cdot \underbrace{Pr[x_t|e_t^* = i,\ x_1^{t-1}]}_{p_i(x_t|x_1^{t-1})} && (6)\\ &= \sum_{i=1}^{N} v_{t,i} p_i(x_t|x_1^{t-1}). && (7) \end{aligned}$$

Observe that (7) is almost what we had in the single expert case, except now we have $v_{t,i}$'s instead of $w_{t,i}$'s. We just need to calculate the $v_{t,i}$'s before we can present our modified version of Bayes algorithm.

From the pretend prior generation process we know that

$$v_{1,i} = Pr[e_1^* = i] = \frac{1}{N}$$

for any i. Now we only need to compute $v_{t+1,i}$,

$$\begin{aligned} v_{t+1,i} &= Pr[e_{t+1}^* = i|x_1^t] && (8)\\ &= \sum_{j=1}^{N} Pr[e_t^* = j|x_1^t] \cdot Pr[e_{t+1}^* = i|e_t^* = j,\ x_1^t] && (9) \end{aligned}$$

where line (9) follows from marginalization. The first term in the sum on line (9) can be reduced to familiar terms by Bayes' theorem namely

$$\begin{aligned} Pr[e_t^* = j|x_1^t] &= Pr[e_t^* = j|x_t,\ x_1^{t-1}]\\ &= \frac{Pr[x_t|e_t^* = j, x_1^{t-1}] \cdot Pr[e_t^* = j|x_1^{t-1}]}{Pr[x_t|x_1^{t-1}]}\\ &= \frac{p_j(x_t|x_1^{t-1}) \cdot v_{t,j}}{q_t(x_t)}. \end{aligned}$$

The second term in the sum on line (9) is actually equivalent to (2), as the event $e^*_{t+1} = i$ is conditionally independent to x_1^t conditioned on $e^*_t = j$. Hence,

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

We can now present Bayes Algorithm for Switching N Base Experts, where the red text below highlights the differences between this algorithm and the original version of Bayes Algorithm:

- Initialize $v_{1,i} = \frac{1}{N}$ for all i.
- for $t = 1, \ldots, T$:
  - Base expert i predicts distribution $p_{t,i}$
  - Learner predicts q_t where $q_t(x) = \sum_{i=1}^N v_{t,i} p_{t,i}(x)$
  - Observe x_t
  - For all i, update $v_{t+1,i}$ according to:

$$v_{t+1,i} = \sum_{j=1}^N \frac{v_{t,j} \cdot p_{t,j}(x_t)}{q_t(x_t)} \cdot \begin{cases} 1-\alpha & \text{if } i = j \\ \frac{\alpha}{N-1} & \text{otherwise.} \end{cases}$$

Observe that the above algorithm has an $\mathcal{O}(N^2)$ runtime per round with the major bottleneck coming from the update step requiring $\mathcal{O}(N^2)$ computations. Although the above algorithm is already quite fast, being polynomial rather than exponential, we can do even better!

First, define c_j to be the fractional term in the update, that is

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

Rewriting the update term using indicators yields

$$\begin{aligned} v_{t+1,i} &= \sum_{j=1}^N \left[c_j \left(\frac{\alpha}{N-1} + \mathbb{1}\{i = j\} \cdot \left(1 - \alpha - \frac{\alpha}{N-1} \right) \right) \right] \\ &= \frac{\alpha}{N-1} \left(\sum_{j=1}^N c_j \right) + c_i \left(1 - \alpha - \frac{\alpha}{N-1} \right) \\ &= \frac{\alpha}{N-1} \cdot 1 + \frac{v_{t,i} p_{t,i}(x_t)}{q_t(x_t)} \cdot \left(1 - \alpha - \frac{\alpha}{N-1} \right) \end{aligned}$$

where the last line follows from the fact that the c_j's sum to 1 due to the normalizing $q_t(x_t)$ in the denominator. Consequently, for each i, the update step now takes $\mathcal{O}(1)$ time, instead of $\mathcal{O}(N)$ which means that the modified version of Bayes algorithm runs in $\mathcal{O}(N)$ time! Observe that the term c_i is identical to the weight update we would have done in the case of no switching experts. The only difference is now we redistribute the weights to the other experts on each update. In this form, the modified Bayes algorithm is a special case of a more general algorithm known as *the weight share algorithm*.

4 Preview of Portfolio Selection

Now we move our focus to applying online learning to learn strategies that are good for investing. To do so, we first introduce some notation and the general setup. Instead of N experts we now have N stocks we can invest in. Moreover, let

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

where $p_t(i)$ is known as the *price relative*. For example, if stock i increases by 5%, then $p_t(i) = 1.05$. However, if stock i decreases by 5%, then $p_t(i) = 0.95$. Moreover, we let

$$S_t = \text{ wealth at the } \underline{\text{start}} \text{ of day } t$$

and we assume that $S_1 = 1$. Furthermore, let

$$w_t(i) = \text{ fraction we invest in stock } i \text{ at the start of day } t$$

where for each t we have $\sum_{i=1}^{N} w_t(i) = 1$. Observe from these definitions that

$$S_t w_t(i) = \text{ wealth invested in stock } i \text{ at the } \underline{\text{start}} \text{ of day } t$$

and

$$S_t w_t(i) p_t(i) = \text{ wealth invested in stock } i \text{ at the } \underline{\text{end}} \text{ of day } t.$$

This implies that our total wealth at time $t+1$ is given by

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

Our objective is to maximize our total wealth at the end of T days. Specifically, we would like to maximize S_{T+1} given by

$$S_{T+1} = 1 \cdot (\mathbf{w}_1 \cdot \mathbf{p}_1) \cdots (\mathbf{w}_T \cdot \mathbf{p}_T) = \prod_{t=1}^{T} (\mathbf{w}_t \cdot \mathbf{p}_t).$$

Equivalently, we can view maximizing S_{t+1} to minimizing

$$\max \prod_{t=1}^{T} (\mathbf{w}_t \cdot \mathbf{p}_t) \equiv \min \sum_{t=1}^{T} (-\ln(\mathbf{w}_t \cdot \mathbf{p}_t)),$$

where the maximum and minimum is over the choice of $\mathbf{w}_t$'s. Looking closely, it seems that the summand on the right-hand side looks like a log-loss. With this motivation in mind, we can pose investing as an online learning problem

for $t = 1, \ldots, T$:

- Learner picks $\mathbf{w}_t$
- World chooses $\mathbf{p}_t$
- Loss $= -\ln(\mathbf{w}_t \cdot \mathbf{p}_t)$

We will analyze the case in which $\mathbf{p}_t$ is chosen adversarially. We will try to see if we can use Bayes Algorithm in this situation. Note that we can map one-to-one each expert i to a stock i. Moreover assume that we can bound the price relative over t and i by a constant, specifically

$$\max_{t,i} p_t(i) \leq C$$

which directly implies that $p_t(i) \in [0, C]$. Note that this is a perfectly reasonable assumption. For example, in the real world C might be equal to a trillion. This bound on C implies that it would be impossible for any person to invest a dollar and see it increase in value to a trillion dollars in one day. Arriving at a naming conflict, from here on out any variable with a tilde above it will refer to the corresponding variable in the Bayes algorithm setting. Let $\tilde{\mathcal{X}} = \{0, 1\}$ denote the set of outcomes for which the probability distributions will be defined. Moreover, let

$$\tilde{p}_{t,i}(1) = \frac{p_t(i)}{C}.$$

The above lets us define $\tilde{p}_{t,i}(0) = 1 - \tilde{p}_{t,i}(1)$. The rest of the reduction will be given in the next lecture.