COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Jeffrey Helt

Lecture #23
April 29, 2019

1 Review of Bayes Algorithm

A few classes ago, we saw Bayes algorithm for online learning with log loss. Recall the algorithm with uniform priors.

- N experts
- $w_{1,i} = \frac{1}{N}$ for each of our N experts
- for $t = 1, \ldots, T$:

  expert i predicts distribution $p_{t,i}$
  master predicts distribution q_t where $q_t(x) = \sum_{i=1}^{N} w_{t,i} \cdot p_{t,i}(x)$
  observe $x_t \in \mathcal{X}$
  for all i, update $w_{t+1,i} = \frac{w_{t,i} p_{t,i}(x_t)}{Z}$ where Z is the normalization constant.

We previously proved the following regret bound for the Bayes algorithm:

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

2 Portfolio selection

Last class, we started looking at the portfolio selection problem in which we want to use the framework of online learning to develop a good investment strategy. Recall the setup for the problem:

- N stocks
- Let $p_t(i)$, the price relative, be defined as

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

- Define S_t as the wealth at the start of day t, and assume $S_1 = 1$.
- Define $w_t(i)$ as the fraction of wealth invested in stock i at the start of day t, so $\sum_i w_t(i) = 1$.

Observe (from last time) $S_{t+1} = S_t(\mathbf{w}_t \cdot \mathbf{p}_t)$ and $S_{T+1} = \prod_{t=1}^{T} (\mathbf{w}_t \cdot \mathbf{p}_t)$. Our objective is to maximize S_{T+1}, which we can rewrite in an equivalent form:

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

Then the outline of the learning problem is as follows:

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

  choose $\mathbf{w}_t$

  observe $\mathbf{p}_t$

  loss $= -\ln(\mathbf{w}_t \cdot \mathbf{p}_t)$

As in all online learning problems, we need to decide what we want to compare our algorithm's performance to (i.e., what we're competing with). In past problems, it has been natural to benchmark our algorithm's performance against the best expert. Here we ask, can we devise an algorithm that does (nearly) as well as a strategy in which we had invested all of our money in the best stock (which we only know in retrospect)?

3 A first cut: Applying Bayes algorithm

Given the similarities of the portfolio selection problem to previous problems, one natural idea is try to retrofit our problem into the Bayes framework, which will hopefully allow us to reuse our analysis of Bayes algorithm to get a regret bound. Since we will be referring to values from both the Bayes and the portfolio selection algorithms and their notation is similar, we use a tilde to denote values in the Bayes algorithm to avoid confusion.

We start by assuming there exists some C such that $\forall t, \forall i : C \geq p_t(i)$, which implies $p_t(i) \in [0, C]$ for all t and i.

We then need to define the inputs to the Bayes algorithm and how we will use the outputs. We define the domain of our experts (for Bayes) as $\tilde{\mathcal{X}} = \{0, 1\}$. We then define the N experts:

$$\tilde{p}_{t,i}(1) = \frac{p_t(i)}{C} \qquad \tilde{p}_{t,i}(0) = 1 - \tilde{p}_{t,i}(1)$$

Let $\tilde{x}_t = 1$ for all t. On every round, we pass this input to the Bayes algorithm and get back $\tilde{w}_{t,i} \forall i$. Since there is a one-to-one correspondence between the experts in the Bayes world and our stocks, we can just use directly these weight vectors:

$$\forall i : w_t(i) = \tilde{w}_{t,i}$$

In order to compute the regret bound, we need to calculate $\tilde{q}_t(\tilde{x}_t)$.

$$\begin{aligned}
\tilde{q}_t(\tilde{x}_t) &= \tilde{q}_t(1) && \text{since } \forall t : \tilde{x}_t = 1 \\
&= \sum_{i=1}^{N} \tilde{w}_{t,i} \tilde{p}_{t,i}(1) \\
&= \sum_{i=1}^{N} \frac{\tilde{w}_{t,i} p_t(i)}{C} \\
&= \frac{\mathbf{w}_t \cdot \mathbf{p}_t}{C}
\end{aligned}$$

We can now apply our (previously proved) regret bound for Bayes with uniform priors.

$$\begin{aligned}
-\sum_{t=1}^{T} \ln\left(\frac{\mathbf{w}_t \cdot \mathbf{p}_t}{C}\right) &= -\sum_{t=1}^{T} \ln \tilde{q}_t(\tilde{x}_t) \\
&\leq \min_{1 \leq i \leq N} \left[-\sum_{t=1}^{T} \ln \tilde{p}_{t,i}(\tilde{x}_t)\right] + \ln N \qquad \text{using Bayes regret bound} \\
&\leq \min_{1 \leq i \leq N} \left[-\sum_{t=1}^{T} \ln \frac{p_t(i)}{C}\right] + \ln N \qquad \text{by definition}
\end{aligned}$$

Notice we can further simplify this bound by canceling the C's on both sides, which leaves us with

$$-\sum_{t=1}^{T} \ln\left(\mathbf{w}_t \cdot \mathbf{p}_t\right) \leq \min_{1 \leq i \leq N} \left[-\sum_{t=1}^{T} \ln p_t(i)\right] + \ln N \tag{2}$$

We showed previously the left-hand side of this inequality is equivalent to the negative log of the amount of money the algorithm has accumulated over T rounds. Similarly, for a fixed stock i, $-\sum_{t=1}^{T} \ln p_t(i)$ is the wealth accumulated over T rounds by a strategy that only invested in stock i. This leads us to the following interpretation of the bound above.

$$-\ln(\text{wealth of algorithm}) \leq -\ln(\text{wealth of best stock}) + \ln N$$

We can then raise e by both sides and simplify to get

$$(\text{wealth of algorithm}) \geq \left(\frac{1}{N}\right)(\text{wealth of best stock})$$

As it turns out, this bound is somewhat trivial. To see this, let's examine our returns when using a simple "buy and hold" strategy, where we simply divide our initial wealth evenly among the N stocks and make no further adjustments on any following rounds.

What would our return be under the buy and hold strategy? Well, the $1/N$th of our wealth that we invested in the best stock would yield the same returns as the best stock. Thus, even if we lost all of our other money, the simple buy and hold strategy would satisfy our lower bound on returns above. In fact, it is possible to show, through simplification and expanding out definitions, that the algorithm developed above is exactly equivalent to the buy and hold strategy.

4 Portfolio rebalancing

Our somewhat trivial bound above for our investing strategy based on Bayes algorithm motivates us to try to find something better. A common idea used in investing is the idea of "rebalancing," where a person's stock holdings are shifted at the end of every day (or more generally, some time interval) to keep the *proportion* of their wealth invested in each stock constant over time. The general term for this class of strategies is called a "constant rebalanced portfolio" or CRP, and when the proportions of each stock are equal, it is referred to as a uniform CRP.

Intuitively, rebalancing causes us to buy low and sell high. Observe the following toy example with 2 stocks, one with static prices and the other with volatile prices.

Prices at start of day:

Day	Stock 1	Stock 2
1	1	1
2	1	0.5
3	1	1
4	1	0.5
5	1	1
6	1	0.5
	⋮	

Price relatives at end of day:

Day	Stock 1	Stock 2
1	1	1/2
2	1	2
3	1	1/2
4	1	2
5	1	1/2
6	1	2
	⋮	

Our buy and hold strategy has a return of zero with these two stocks because over the long-run their prices don't change. However, observe the performance of a uniform CRP in which we rebalance at the end of every day.

$$\begin{aligned} S_1 &= 1 \\ S_2 &= S_1 \left(\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{2} \right) = \frac{3}{4} S_1 \\ S_3 &= S_2 \left(\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2 \right) = \frac{3}{2} S_2 \\ &\vdots \\ S_{t+2} &= \frac{3}{4} \cdot \frac{3}{2} S_t = \frac{9}{8} S_t \end{aligned}$$

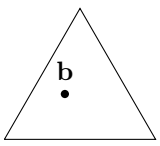
The last equation shows that every two days, our wealth increases by 12.5% $\left(\text{i.e., } \frac{9}{8} - 1\right)$ using the uniform CRP strategy.

5 Universal Portfolio Algorithm (UP)

In general, a CRP is defined by a vector of length N, where N is the number of stocks. More formally, a CRP is defined by

$$\mathbf{b} = \langle b_1, \ldots, b_N \rangle \text{ where } \forall i : b_i \geq 0 \text{ and } \sum_{i=1}^{N} b_i = 1$$

For a fixed number of stocks N, there are infinitely-many CRPs, one for each value of $\mathbf{b}$. In fact, since $b_i \in \mathbb{R}$, there are uncountably-many CRPs. However, since we have the constraint that $\sum_i b_i = 1$, the CRPs form an $(N-1)$-dimensional simplex in $\mathbb{R}^N$. For instance, with $N = 3$, we can visualize the space of CRPs as a two-dimensional equilateral triangle in $\mathbb{R}^3$.



Given that there are infinitely-many CRPs, can we ever hope to have an online algorithm that does almost as well as the best CRP, which we only know in hindsight? It turns out the answer is yes. The key idea is to simultaneously invest an (infinitesimally) small portion of

our total wealth according to each possible CRP. This is the key idea behind the universal portfolio (UP) algorithm.

More concretely, we want to calculate $w_t(i)$ for all t and i to serve as the weights in our online investment algorithm. We are using the same notation here as we did when applying Bayes algorithm. For a fixed N, define Δ as the set of all CRPs. That is,

$$\Delta = \{\text{all CRPs}\} = \left\{ \mathbf{b} \in [0,1]^N : \sum_{i=1}^{N} b_i = 1 \right\}$$

Let $d\mu(\mathbf{b})$ denote the portion of our wealth that we're investing according to $\mathbf{b}$. Then for a fixed $\mathbf{b}$, observe that $\prod_{s=1}^{t-1}(\mathbf{b}\cdot\mathbf{p}_s)\cdot d\mu(\mathbf{b})$ is the amount of wealth invested according to $\mathbf{b}$ after $t-1$ days, since we started with $d\mu(\mathbf{b})$ and rebalanced according to $\mathbf{b}$ at the start of every day. Similarly, the fraction of stock i that we hold due to $\mathbf{b}$ at the beginning of day t is $b_i \prod_{s=1}^{t-1}(\mathbf{b}\cdot\mathbf{p}_s)\cdot d\mu(\mathbf{b})$. Using these two ideas, we can now write down $w_t(i)$. Let

$$w_t(i) = \frac{\int_{\mathbf{b}\in\Delta} b_i \prod_{s=1}^{t-1}(\mathbf{b}\cdot\mathbf{p}_s) d\mu(\mathbf{b})}{\int_{\mathbf{b}\in\Delta} \prod_{s=1}^{t-1}(\mathbf{b}\cdot\mathbf{p}_s) d\mu(\mathbf{b})}$$

where the integrals are taken over the space of all possible CRPs (i.e., the simplex). The numerator is the total wealth invested in stock i on day t, and the denominator is the total wealth of the algorithm at the start of day t.

5.1 Regret bounds

We now prove regret bounds for the UP algorithm.

Theorem 1. *After T rounds,*

$$(\textit{wealth of UP}) \geq \frac{1}{(T+1)^{N-1}} (\textit{wealth of best CRP})$$

We will actually prove a slightly weaker version of this regret bound because the proof is simpler and more intuitive. Namely, we will show, after T rounds,

$$(\text{wealth of UP}) \geq \frac{1}{e(T+1)^{N-1}} (\text{wealth of best CRP})$$

Proof. Let $\mathbf{b}^*$ be the best CRP in hindsight. The outline of the proof is as follows. We start by arguing that the CRPs $\mathbf{b}$ that are close to $\mathbf{b}^*$ in the simplex, which we call $\mathbf{b}^*$'s neighborhood, make almost as much money as $\mathbf{b}^*$. Then we argue that we invested a decent amount of our total wealth in strategies that lie within the neighborhood. Finally, we combine these two ideas to prove our regret bound.

Step 1. Let $\Delta = \{\mathbf{b} \in [0,1] : \sum_i b_i = 1\}$ as above, and define $\mathcal{N}(\mathbf{b}^*)$, the neighborhood of $\mathbf{b}^*$ for some fixed $\alpha \in [0,1]$, as:

$$\mathcal{N}(\mathbf{b}^*) = \{(1-\alpha)\mathbf{b}^* + \alpha\mathbf{z} : \mathbf{z} \in \Delta\}.$$

Consider a single $\mathbf{b} \in \mathcal{N}(\mathbf{b}^*)$. Then

$$\begin{aligned}
&\mathbf{b} = (1-\alpha)\mathbf{b}^* + \alpha\mathbf{z} \\
\implies &\mathbf{b} \cdot \mathbf{p}_t = (1-\alpha)\mathbf{b}^* \cdot \mathbf{p}_t + \underbrace{\alpha\mathbf{z} \cdot \mathbf{p}_t}_{\geq 0} \\
\implies &\mathbf{b} \cdot \mathbf{p}_t \geq (1-\alpha)\mathbf{b}^* \cdot \mathbf{p}_t \\
\implies &\prod_{t=1}^{T} (\mathbf{b} \cdot \mathbf{p}_t) \geq (1-\alpha)^T \prod_{t=1}^{T} (\mathbf{b}^* \cdot \mathbf{p}_t)
\end{aligned}$$

This tells us that after T days, the wealth accumulated by investing according to strategy $\mathbf{b}$ in the neighborhood of $\mathbf{b}^*$ is at least $(1-\alpha)$ of the wealth accumulated by investing according to strategy $\mathbf{b}^*$. In other words,

$$(\text{wealth of } \mathbf{b}) \geq (1-\alpha)^T (\text{wealth of } \mathbf{b}^*).$$

Step 2. Our initial wealth will be distributed over all of the $\mathbf{b} \in \Delta$. Thus, we can measure the proportion of wealth invested in $\mathcal{N}(\mathbf{b}^*)$ by relating the volume of $\mathcal{N}(\mathbf{b}^*)$ to the volume of Δ.

$$\begin{aligned}
\text{Vol}(\mathcal{N}(\mathbf{b}^*)) &= \text{Vol}(\{(1-\alpha)\mathbf{b}^* + \alpha\mathbf{z} : \mathbf{z} \in \Delta\}) \\
&= \text{Vol}(\{\alpha\mathbf{z} : \mathbf{z} \in \Delta\}) && (1-\alpha)\mathbf{b}^* \text{ just translates the neighborhood} \\
&= \alpha^{N-1}\text{Vol}(\{\mathbf{z} : \mathbf{z} \in \Delta\}) \\
&= \alpha^{N-1}\text{Vol}(\Delta)
\end{aligned}$$

The penultimate line holds because, as mentioned earlier, Δ is an $(N-1)$-dimensional simplex in $\mathbb{R}^N$. When each $\mathbf{z} \in \Delta$ is scaled by some constant $\alpha \in [0,1]$, the volume of the resulting simplex is scaled by α^{N-1}.

Recall the fraction of wealth initially invested in $\mathcal{N}(\mathbf{b}^*)$ is given by $\text{Vol}(\mathcal{N}(\mathbf{b}^*))/\text{Vol}(\Delta)$. Thus, the equality above tells us that α^{N-1} of our wealth is initially invested in the set of strategies $\mathbf{b} \in \mathcal{N}(\mathbf{b}^*)$.

Step 3. The total wealth of our algorithm then is lower bounded by the wealth generated by only the strategies in $\mathcal{N}(\mathbf{b}^*)$. Since we know that each of these strategies earns at least $(1-\alpha)^T(\text{wealth of } \mathbf{b}^*)$, we can then get a lower bound on the total wealth earned under UP. Combining steps 1 and 2, we see

$$\begin{aligned}
(\text{wealth of UP}) &\geq \alpha^{N-1}(1-\alpha)^T(\text{wealth of best CRP } \mathbf{b}^*) \\
&\geq \frac{1}{e(T+1)^{N-1}}(\text{wealth of best CRP } \mathbf{b}^*) && \text{taking the best } \alpha = (T+1)^{-1}
\end{aligned}$$

□

This concludes our analysis of the universal portfolio algorithm. We will now move on to our final topic of the semester, connections between machine learning and game theory.

6 Connections to game theory

Game theory is the mathematical study of games. In addition to more obvious games, such as checkers and chess, we can study many interactions through the lens of game theory.

Many interactions between people, companies, and animals can all be viewed as games. Similarly, machine learning studies the interaction between a learner and the environment, or adversary. This has been especially obvious in our recent investigation of online learning, where we've analyzed algorithms under adversarial conditions.

Let us start by developing some of the language used by game theorists. Rock paper scissors (RPS) serves as the canonical game used to introduce game theory. It involves two players. At each round, both players simultaneously select one of rock, paper, or scissors, and then the winner of the round is determined based on the players' choices. For instance, if one player chooses paper and the other chooses scissors, the second player wins because by the rules, "scissors cut paper."

As we have done previously in class, we call our players Max and Mindy. We can represent a game like RPS by a matrix in which Max's choices are denoted by columns and Mindy's choices are denoted by rows. Each entry $M(i, j)$ in the matrix gives the outcome of the game when Max chose i and Mindy chose j. For instance, for RPS we have

		Max		
		R	P	S
Mindy	R	$\frac{1}{2}$	1	0
	P	0	$\frac{1}{2}$	1
	S	1	0	$\frac{1}{2}$

where the outcomes of each instance of RPS are given in the form of Mindy's loss. A zero means Mindy won, $\frac{1}{2}$ denotes a tie, and a one means Max won. In general, we can write down any two-player, zero-sum game using a matrix like the one above, although it may require an enormous matrix as is the case for chess. "Zero-sum" here means that the rules of the game are such that if one player gains an advantage, then the other play must have lost some advantage (and vice-versa).

In general, we can write two-player, zero-sum games using a matrix $\mathbf{M}$. For now, we assume Max and Mindy make their choices simultaneously.

$\mathbf{M}$	$\ldots$	j	$\ldots$
1			
2			
$\vdots$			
i		$M(i,j)$	
$\vdots$			

where $M(i, j) \in [0, 1]$ denotes Mindy's loss. Individual columns or rows in the matrix are called pure (i.e., deterministic) strategies.

We can also define mixed (i.e., random) strategies. Mindy defines a distribution $\mathbf{P}$ over the rows in $\mathbf{M}$ and then her choice at each round is made by drawing $i \sim \mathbf{P}$. Similarly, Max defines a distribution $\mathbf{Q}$ over the columns and then chooses by drawing $j \sim \mathbf{Q}$.

We will continue our discussion of the connections between game theory and machine learning next class, but for now, we introduce some additional notation. We write Mindy's

expected loss as

$$\begin{aligned}\text{Mindy's expected loss} &= \sum_{i,j} P(i)M(i,j)Q(j) \\ &= \mathbf{P}^T\mathbf{M}\mathbf{Q}\end{aligned}$$

We will also use the notation $\mathbf{M}(\mathbf{P},\mathbf{Q})$ to denote $\mathbf{P}^T\mathbf{M}\mathbf{Q}$. Similarly, we use $\mathbf{M}(i,\mathbf{Q})$ to denote Mindy's expected loss when she plays with pure strategy i against Max's mixed strategy $\mathbf{Q}$.