COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Kevin Lee

Lecture #23
April 29, 2014

1 Recap

Last lecture, we briefly introduced the setup for portfolio selection. We assume that every time period, there are N stocks available to us, and we want to figure out how to best allocate our money among the stocks at the beginning of every investment period. We define:

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

as the price relative which is how much a stock goes up or down in a single day.

S_t denotes the amount of wealth we have at the start of day t and we assume $S_1 = 1$. We denote $w_t(i)$ to be the fraction of our wealth that we have in stock i at the beginning of day t which can be viewed as a probability distribution as $\forall i, w_t(i) \geq 0$ and $\sum_i w_t(i) = 1$. We can then derive the total wealth in stock i at the start of day t to be $S_t w_t(i)$ and the total wealth in stock i at the end of day t to be $S_t w_t(i) p_t(i)$. We can use a simple summation over the i stocks to find our total wealth at the end of day t as:

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

The total wealth after T time periods is then:

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

2 Portfolio Selection

Our goal is to make as much money as possible. This is done by maximizing $\prod_{t=1}^{T}(\mathbf{w}_t \cdot \mathbf{p}_t)$. This is the same as maximizing the log of the expression, which is $\sum_{t=1}^{T} \ln(\mathbf{w}_t \cdot \mathbf{p}_t)$. This is also the same as minimizing the negative of the log of the expression, $\sum_{t=1}^{T} -\ln(\mathbf{w}_t \cdot \mathbf{p}_t)$. We thus notice that maximizing wealth is equivalent to minimizing an expression that looks like log loss, so we can view investing as an online learning problem:

for $t = 1, ..., T$:
  learner/investor chooses $\mathbf{w}_t$
  stock market/nature chooses $\mathbf{p}_t$
  loss $= -\ln(\mathbf{w}_t \cdot \mathbf{p}_t)$

In each round, the learner chooses w_t which is how he invests money, and the stock market responds with each stock going up or down that day as represented by $\mathbf{p}_t$. Our goal is then to minimize the cumulative loss.

A natural starting point is to aim to do almost as well as the best individual stock. We can do this by massaging our problem so that we can apply Bayes algorithm which is designed around log loss. We define the Bayes algorithm outcome space to be $X = \{0, 1\}$ and choose $C \geq p_t(i)\ \forall t, i$. For each timestep t of the algorithm, each expert i needs to come up with a probability distribution over the outcome space X by coming up with a probability for outcome 1, $p_{t,i}(1)$, and trivially $p_{t,i}(0) = 1 - p_{t,i}(1)$. Let the outcomes $x_t = 1\ \forall t$. We can then let $p_{t,i}(1) = \frac{p_t(i)}{C}$. Applying Bayes algorithm will give us back a weight vector $w_{t,i}$, and we use these weights for investing by setting $w_t(i) = w_{t,i}$. Now observe that:

$$q_t(x_t) = q_t(1) = \sum_i w_{t,i} p_{t,i}(1) = \sum_i \frac{w_{t,i} p_t(i)}{C} = \frac{\mathbf{w}_t \cdot \mathbf{p}_t}{C}$$

The bound on log loss guaranteed by Bayes algorithm says:

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

$$-\sum_t \ln(\frac{\mathbf{w}_t \cdot \mathbf{p}_t}{C}) \leq \min_i -\sum_t \ln \frac{p_t(i)}{C} + \ln N$$

$$-\sum_t \ln(\mathbf{w}_t \cdot \mathbf{p}_t) \leq \min_i -\sum_t \ln p_t(i) + \ln N$$

This essentially says:

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

We then remove the logs to find:

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

The algorithm is actually equivalent to the "buy and hold" strategy where on day 1 we invest $\frac{1}{N}$ of our wealth into each stock and then just leave it there. Since this means we will invest $\frac{1}{N}$ of our wealth into the best stock, the bound on the wealth of the algorithm naturally says that we will make at least $\frac{1}{N}$ of the wealth of the best stock, even in the case that we lose all our money in the other stocks. When we invest our money, ideally we want the money to grow exponentially at the rate c^t where c is a constant that is to be maximized. The bound on the wealth of the algorithm implies that the constant c that we get from the algorithm will be asymptotically at least as good as that of the best underlying stock.

3 Constant Rebalanced Portfolio

Instead of comparing with the best individual stock, we now look towards comparing with constant rebalanced portfolios (CRP). In a CRP, we decide ahead of time on fixed allocations among the different stocks and, everyday, rebalance the different portfolios so that they always have those fixed allocations. The simplest kind of CRP is a uniform CRP (UCRP) where everyday we rebalance equally among the N stocks. CRP is a very common strategy, as it is natural to constantly rebalance your portfolio so that you have, for example, 60% in

day	stock 1 price	stock 2 price	stock 1 price relative	stock 2 price relative
1	1	1	1	0.5
2	1	0.5	1	2
3	1	1	1	0.5
4	1	0.5	1	2
5	1	1	1	0.5

Figure 1: Stock 1 and 2 behavior

stock, 30% in bonds, and 10% in cash. If stocks go up and bonds go down, then CRP will have you sell stock to buy more bonds. This thus encourages buying low and selling high.

We present a concrete example where CRP is a good idea. Imagine there is stock 1 and stock 2 that both start at \$1. The price of stock 1 never changes, and the price of stock 2 is \$1 on odd days and \$0.50 on even days. The behavior of these two stocks is depicted for 5 days in Figure 1.

The buy and hold strategy will never earn money when applied to these two stocks. We then look at how UCRP performs:

$$\begin{aligned} S_1 &= 1 \\ S_2 &= S_1(\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{2}) = S_1 \cdot \frac{3}{4} \\ S_3 &= S_2(\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2) = S_2 \cdot \frac{3}{2} = S_1 \cdot \frac{3}{4} \cdot \frac{3}{2} \end{aligned}$$

More generally, if we have S_t on day t then $S_{t+2} = S_t \cdot \frac{3}{4} \cdot \frac{3}{2} = S_t \cdot \frac{9}{8}$. Thus every two days our wealth grows by 12.5%, so it grows exponentially.

4 Universal Portfolio Algorithm

We now return to making an algorithm to try to do almost as well as the best CRP instead of the best individual stock. Let us say that each CRP is a vector $\mathbf{b} = \langle b_1, ..., b_N \rangle$ which forms a valid distribution over the N stocks, and using the CRP means using $w_t(i) = b_i$. We then would want to reapply Bayes algorithm as we previously did, but instead of splitting wealth amongst stocks, we split it amongst all possible CRP's. There are uncountably infinite possible CRPs, so for each CRP, $\mathbf{b}$, we give it an infinitesimally small piece of our wealth, $d\mu(\mathbf{b})$. At the start of day t:

$$\text{wealth in CRP } \mathbf{b} = \prod_{s=1}^{t-1} (\mathbf{b} \cdot \mathbf{p}_s) d\mu(\mathbf{b})$$

$\prod_{s=1}^{t-1}(\mathbf{b} \cdot \mathbf{p}_s)$ is how much wealth we would have at the start of day t if we had started with \$1 invested in the CRP, and $d\mu(\mathbf{b})$ scales this wealth down as we only in fact invested an infinitesimally small amount into the CRP. We can simply integrate over the set of all possible CRP's $\mathbf{b}$ to find the total wealth at the start of day t:

$$\text{total wealth} = S_t = \int \prod_{s=1}^{t-1} (\mathbf{b} \cdot \mathbf{p}_s) d\mu(\mathbf{b})$$

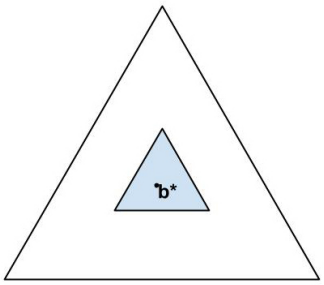

Figure 2: Simplex

We can use another integral over the set of all possible CRP's $\mathbf{b}$ to find the total wealth invested in stock i at the start of day t, where b_i is the fraction of our wealth in i, as:

$$\text{total wealth in stock } i = \int b_i \prod_{s=1}^{t-1} (\mathbf{b} \cdot \mathbf{p}_s) d\mu(\mathbf{b})$$

We can then calculate the fraction of our wealth we need to invest in stock i, $w_t(i)$ as:

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

Using this $w_t(i)$ to rebalance our portfolio at each time step is known as the Universal Portfolio (UP) Algorithm or Cover's algorithm.

5 Bounds on UP Algorithm

Theorem 1. *Wealth of UP algorithm* $\geq (\frac{1}{(T+1)^{N-1}})$*(Wealth of best CRP)*

While $(\frac{1}{(T+1)^{N-1}})$ may appear to be a small fraction, it holds for any stock market and still says the rate of exponential growth of this algorithm will eventually match the rate of exponential growth of the best CRP.

We will proceed to prove a slightly weaker version of this theorem in two steps. Let the best CRP be $\mathbf{b}^*$. Fortunately in our algorithm, we put part of our money into $\mathbf{b}^*$, but unfortunately we only put in an infinitesimally small amount. We note that the CRP's are essentially just probability vectors and so they live in the space of all probability vectors which is known as the simplex. We illustrate this simplex in the case that we have 3 stocks in Figure 2. $\mathbf{b}^*$ is just a point in the simplex and we consider the neighborhood around $\mathbf{b}^*$, which consists of CRPs that are close to $\mathbf{b}^*$, as illustrated by the inner shaded triangle in Figure 2. We then want to argue two points. In step 1, we want to argue that all of the CRPs in the neighborhood of $\mathbf{b}^*$ attain wealth close to $\mathbf{b}^*$. Then in step 2, we want to show that the overall size of the neighborhood is large. Let us define

$$\Delta = \{\text{all CRPs}\} = \{\mathbf{b} : b_i \geq 0, \sum_i b_i = 1\}$$

and we define the neighborhood of $\mathbf{b}^*$ as:

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

where α is a small positive number and we are essentially mixing $\mathbf{b}^*$ with some other distribution $\mathbf{z}$.

Proof. Step 1:
Let us say that $\mathbf{b}$ is one of the points in the neighborhood of $\mathbf{b}^*$ and so
$\mathbf{b} = (1-\alpha)\mathbf{b}^* + \alpha\mathbf{z}$. We can derive the amount of wealth that $\mathbf{b}$ gains at time t to be:

$$\mathbf{b} \cdot \mathbf{p}_t = (1-\alpha)\mathbf{b}^* \cdot \mathbf{p}_t + \alpha\mathbf{z} \cdot \mathbf{p}_t$$

$\mathbf{z} \cdot \mathbf{p}_t \geq 0$ as price relatives are never negative. Thus, after T timesteps:

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

□

Proof. Step 2:

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

where Vol() denotes volume. We now note that $(1-\alpha)\mathbf{b}^*$ is a fixed quantity, so the total volume will be the same if we shift the simplex and remove this quantity:

$$\begin{aligned}\text{Vol}(N(\mathbf{b}^*)) &= \text{Vol}(\{\alpha\mathbf{z} : \mathbf{z} \in \Delta\}) \\ &= \text{Vol}(\Delta) \cdot \alpha^{N-1}\end{aligned}$$

The last equality holds as the simplex is an $N-1$ dimensional object and each dimension is being scaled by α. □

We can now combine the results of the two steps to show:

$$\begin{aligned}\text{wealth of UP algorithm} &\geq (\text{fraction of CRPs in } N(\mathbf{b}^*))(\text{minimum wealth of any CRP in } N(\mathbf{b}^*)) \\ &\geq \alpha^{N-1}(1-\alpha)^T(\text{wealth of } \mathbf{b}^*) \\ &\geq \frac{1}{e(T+1)^{N-1}}(\text{wealth of } \mathbf{b}^*)\end{aligned}$$

where the last inequality holds if we choose $\alpha = \frac{1}{T+1}$. Thus, we have proved a slightly weaker version of the theorem.

6 Game Theory

Game Theory is a field that studies games and is really about interactions between players of all kinds. There is a natural connection to learning, as in learning there is often an interaction between a teacher and a student or between a learner and nature. For our purposes, a game is defined by a matrix. The game matrix for rock, paper, scissors is shown in Figure 3. The rows of the matrix are actions that the row player Mindy can take, and the columns are actions that the column player Max can take. Mindy chooses one of the rows, and Max chooses one of the columns. The entry that is defined by these choices

	R	P	S
R	0.5	1	0
P	0	0.5	1
S	1	0	0.5

Figure 3: Rock, Paper, Scissors Game Matrix

in the matrix is the loss suffered by Mindy. Max tries to maximize this loss while Mindy tries to minimize it. In general we will always have a game matrix M. To play, Mindy chooses row i, and Max chooses column j. Individual rows i and columns j are called pure strategies. The corresponding entry in the matrix $M(i, j)$ is the loss suffered by Mindy. In principle, any two-person zero-sum game can be put into this form.