COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Nadia Heninger

Lecture #22
April 28, 2008

1 Portfolio Selection

Last time we discussed our model of the stock market.

N stocks

start on day 1 with \$1

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

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

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

The idea is that we will choose w_t first and observe p_t at the end of the day.
From the definitions of the above quantities, we showed that:

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

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

Assuming your goal is to maximize wealth, this can be formulated as the following optimization problem:

$$\max \prod_t (\mathbf{w_t} \cdot \mathbf{p_t}) \equiv \min \sum_t -\ln(\mathbf{w_t} \cdot \mathbf{p_t}).$$

In this analysis, we are not going to make statistical assumptions about the movement of the stock market. Instead, we would like to find a strategy that works no matter what the stock market does, so we will use online learning.

Last time, we developed a bound for the performance of Bayes' algorithm to the best single stock:

$$-\sum_t \ln(\mathbf{w_t} \cdot \mathbf{p_t}) \leq \min_i \left[-\sum_t \ln p_t(i) \right] + \ln N.$$

Or in terms of wealth and stocks,

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

Exponentiating both sides gives

$$(\text{wealth of alg.}) \geq \frac{1}{N}(\text{wealth of best stock}).$$

This is completely trivial: in order to achieve this bound, simply divide all money evenly between the N stocks and leave it there. This is the "buy and hold" strategy.

$$\begin{aligned} \text{wealth} &= \frac{1}{N}\sum_{i=1}^{N}(\text{wealth for stock } i) \\ &\geq \frac{1}{N}(\text{wealth of best stock}). \end{aligned}$$

If you go back and unravel Bayes' algorithm, it is doing exactly buy-and-hold, so last time we went through a lot of work to prove something trivial. But this is not so bad, because stocks typically increase in value exponentially quickly. This bound says that you can get almost the same exponential rate as the best stock.

What can we do that is more interesting? We might want to find an algorithm that does almost as well as the best switching sequence of experts. Such an algorithm is simple based on our earlier analysis: at every time period, remove a fixed fraction of your money by selling stocks, and redistribute it among the stocks.

But instead of analyzing this in terms of a switching sequence of experts, we are going to compete against a different strategy: *constant rebalanced portfolios* (CRP).

2 Constant Rebalanced Portfolios

The strategy of a constant rebalanced portfolio is as follows: every day at the beginning of the day, sell off everything, and buy back stocks in proportions that remain fixed. In a *uniform rebalanced portfolio*, for example, on every day you would sell everything and invest $\frac{1}{N}$ of your money in each stock.

Such a portfolio is defined by the quantity b_i, the fraction of money to invest on each day in stock i. We know that

$$b_i \geq 0 \quad \text{and} \quad \sum_i b_i = 1 \quad \text{and} \quad \forall t, i : w_t(i) = b_i.$$

Why is this useful? Here is a toy example with two stocks.

- Stock 1 has a constant value = 1. (i.e. you put money under your mattress)
- Stock 2 doubles and halves in value on alternating days.

day	**stock 1**	**stock 2**
1	1	1
2	1	$\frac{1}{2}$
3	1	1
4	1	$\frac{1}{2}$
$\vdots$	$\vdots$	$\vdots$

(a) prices

day	**stock 1**	**stock 2**
1	1	$\frac{1}{2}$
2	1	2
3	1	$\frac{1}{2}$
4	1	2
$\vdots$	$\vdots$	$\vdots$

(b) price relatives

If you buy and hold, you will never make any money. What happens with a uniform constant rebalanced portfolio?

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

So

$$S_{t+2} = S_t \cdot \frac{3}{4} \cdot \frac{3}{2} = \frac{9}{8} S_t,$$

which means that every two days the amount of wealth increases by a factor of $\frac{9}{8}$. The wealth grows exponentially fast even though the underlying stocks are doing nothing. This is just a contrived example, but the underlying idea is to sell one stock and buy another when prices change—in short, the algorithm encourages you to sell high and buy low.

We would like to look at algorithms that do almost as well as the best constant rebalanced portfolio in hindsight. We will use online learning to prove bounds on their performance.

3 Universal Portfolios

We will look at the *universal portfolio* algorithm, which is due to Tom Cover. The "universal" in the name is in analogy to universal compression, as Cover comes from the information theory community.

We will think of investing in strategies, rather than stocks, analogous perhaps to dividing your money between different mutal funds (or hedge funds, if you're rich enough) and letting them manage your money for you.

On day 1, we will divide our money evenly among *all* constant rebalanced portfolios, and let each constant rebalanced portfolio invest based on its strategy. In effect, we will be using "buy and hold" with an infinitely large family of strategies. There is just one problem: the number of portfolios is not even countably infinite—there is a whole continuum of them!

On day t,

$$\text{wealth invested with CRP } \mathbf{b} = \prod_{s=1}^{t-1} (\mathbf{b} \cdot \mathbf{p_s}) d\mu(\mathbf{b}).$$

The $d\mu(\mathbf{b})$ represents an infinitesimal fraction of the wealth, which means that we'll have to take integrals:

$$\text{total wealth of alg.} = \int \prod_{s=1}^{t-1} (\mathbf{b} \cdot \mathbf{p_s}) d\mu(\mathbf{b}).$$

The integral is over the space of all probability distributions with the standard uniform Lebesgue measure.

In order to figure out how much money to put in each stock, we count the amount of money in each stock in the total. Since each CRP $\mathbf{b}$ places b_i of its wealth in stock i, we can compute

$$\begin{aligned} w_t(i) &= \text{fraction of wealth to invest in stock } i \\ &= \frac{\int b_i \prod_{s=1}^{t-1} (\mathbf{b} \cdot \mathbf{p_s}) d\mu(\mathbf{b})}{\int \prod_{s=1}^{t-1} (\mathbf{b} \cdot \mathbf{p_s}) d\mu(\mathbf{b})}. \end{aligned}$$

This formula tells us that we can implement the algorithm by computing this quantity over all vectors $\mathbf{b}$. This results in some rather hairy integrals, but it can be written down mathematically and computed numerically for small portfolios of 10 stocks or so, or you can approximate it using random sampling for larger numbers of stocks.

Now we would like to prove something about this algorithm. We would apply the analysis of Bayes' algorithm if we could, but that applies only to a finite number of experts, and we have an infinite number of them. So we will prove the following theorem:

Theorem.

$$(\textit{wealth of U.P. alg.}) \geq \frac{1}{(T+1)^{N-1}}(\textit{wealth of best CRP in hindsight}).$$

The $\frac{1}{(T+1)^{N-1}}$ is the penalty we pay for using online learning. This seems rather weak, but we expect the best CRP to grow exponentially fast, on the order of e^{cT}. The theorem says that the exponent of growth of the algorithm gets close to the exponent of the best CRP, something like

$$(\text{exponent of algorithm}) \geq c - O\left(\frac{N \ln T}{T}\right).$$

The theorem we will actually prove has a different constant.

Theorem.

$$(\textit{wealth of U.P. alg.}) \geq \frac{1}{e(T+1)^{N-1}}(\textit{wealth of best CRP in hindsight}).$$

The proof is due to Blum and Kalai, and is much simpler than Cover's original proof.

Proof. We'll let $\mathbf{b}^*$ denote the best CRP in hindsight. We gave this strategy some money in the initial allocation, but not enough to give the desired result. So we will look at all CRPs in some neighborhood around $\mathbf{b}^*$. The proof has two parts:

1. Show that all CRPs in a neighborhood of $\mathbf{b}^*$ make almost as much as $\mathbf{b}^*$.
2. Show that the neighborhood got a decent fraction of the initial wealth.

The amount of money that the U.P. algorithm makes is at least (the fraction of money in that neighborhood) $\times$ (the minimum amount of money made by any CRP in the neighborhood).

Let $\Delta = \{\text{all CRPs}\}$. We'll define the neighborhood around $\mathbf{b}^*$ to be

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

The neighborhood is defined by a mixture of $\mathbf{b}^*$ with all portfolios. We will think of α as small, and will define it later. So each strategy in the neighbhorhood puts most of its money in $\mathbf{b}^*$ and a small amount in $\mathbf{z}$.

Part 1: Say $\mathbf{b} = (1-\alpha)\mathbf{b}^* + \alpha\mathbf{z}$. On each time step this strategy gives $(1-\alpha)$ of its wealth to $\mathbf{b}^*$ and α to $\mathbf{z}$. So on each time step,

$$(\mathbf{b}\text{'s increase in wealth}) \geq (1-\alpha)(\mathbf{b}^*\text{'s increase in wealth}).$$

After T time steps,

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

Since α is fairly small, the amount of money that $\mathbf{b}$ gets is close to the amount that $\mathbf{b}^*$ gets.

Part 2: We will compute the volume of the neighborhood. The amount of money in the neighborhood is proportional to its volume.

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

Shifting $\mathbf{b}^*$ to the origin does not change the volume of its neighborhood, so

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

Each dimension of the simplex is scaled by α, so the total size of the scaled simplex goes up by $\alpha^{(\text{dim. of simplex})}$. The dimension of Δ is $N-1$.

Putting the two parts together, we have

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

This expression will be maximized when $\alpha = \frac{1}{T+1}$, and gives the bound

$$(\text{total wealth of U.P.}) \geq \frac{1}{e(T+1)^{N-1}}(\text{wealth of } \mathbf{b}^*).$$

□

Given this algorithm, then, why are we all not rich? People have tried this approach on historical stock market data, and it seems to do pretty well at making money. Cover's algorithm started a whole line of research. It turns out that Cover's algorithm is optimal for theoretical reasons, but in experiments, it is outperformed by the "exponentiated gradient" (EG) rule and even by the uniform constant rebalanced portfolio. Former Princeton students Elad Hazan, Amit Agarwal, and Satyen Kale have done followup work on a method that outperforms the uniform CRP.

In this case, it seems that theory is too pessimistic. The theoretical bounds compare well against worst-case predictions, but in reality the stock market is not so adversarial.

4 Game Theory

Game theory is a beautiful area and is related to machine learning. Our discussion over the next lecture and a half will pull together many of the topics we have seen this semester into a common framework.

Our first example of a *two-person zero-sum game* is rock-paper-scissors. In this game, the two players simultaneously choose a strategy, and the outcome of the game is determined by the rules "rock beats scissors", "paper beats rock", and "scissors beat paper". These rules are summarized in the table below:

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

One player chooses a row, and the other player chooses a column. The corresponding entry in the table shows how much the column player loses, or the row player wins.

This would seem to be a very restrictive and narrow model of a game, but in fact it's possible to put nearly any game into this framework. For example, in chess, we could imagine that each player writes down their entire strategy by specifying a move for every possible board position. Then we could just simulate an entire game by applying the rules instead of actually playing. In principle, we could list all such strategies for every player in an exceedingly large table like the above, where each entry specifies whether a given strategy beats another strategy. Thus each entry $M(i, j) \in [0, 1]$ would give the outcome if the row player plays strategy i and the column player plays strategy j.

Such a deterministic strategy is called a "pure" strategy.

In reality, we want to use randomness in order to play a game effectively, otherwise we would end up like Bart Simpson, who plays rock-paper-scissors by always choosing rock. In a "mixed" strategy, each player will choose a distribution over the strategies available. The row player will choose a distribution P over the rows, and the column player will choose a distribution Q over columns. The expected outcome of such a game will be

$$\sum_{i,j} P(i)M(i,j)Q(j) = P^T MQ = M(P,Q).$$

Next time, we will look at the fundamentals of game theory and connections to on-line learning and boosting.