Lecture 14

Online search problem

- An online player is searching for the maximum (or minimum) price in a sequence of prices.
- At each time period i, the player obtains a price quotation p_i. Then the player must decide whether to accept p_i or continue sampling more prices.
- The game ends when the player accepts some price p_j, and the return is p_j.

One-way trading problem

- The online player is a trader whose goal is to trade some initial wealth D_0 in a currency (dollars) to some other asset or currency (won).
- The price is the exchange rate.

- We simplify the problem and ignore transaction fees.

Theorem 1 *(1) Let alg_1 be any randomized one-way trading algorithm. Then there exists a deterministic one-way trading algorithm alg_2 such that for any price sequence σ, $alg_2(\sigma) = E[alg_1(\sigma)]$. (2) Let alg_2 be any deterministic one-way trading algorithm. Then there exists a randomized one-way trading algorithm alg_1 such that $E[alg_1(\sigma)] = alg_2(\sigma)$ for all σ.*

Proof. (2) If alg_2 trades a fraction s_i of its initial wealth at the ith period, alg_1 trades its entire wealth at the ith period with probability s_i. □

Randomization cannot improve the competitive performance in one-way trading.

Competitive Search Algorithms

- Assume that prices are drawn from some real interval $[m, M]$ for $0 < m \leq M$.
- Set $\varphi = M/m$ and call φ the global fluctuation ratio.
- Assume that time is discrete and the horizon is finite. The duration (the number of periods) is denoted by n.
- We distinguish between known and unknown duration (the player is informed immediately before the last period).

Reservation-Price-Policy

Suppose m and M are known to the player.

The optimal deterministic search algorithm is RPP. Call p^* the reservation price.

RPP: Accept the first price $\geq p^* = \sqrt{Mm}$.

Theorem 2 *RPP is $\sqrt{\varphi}$-competitive.*

Proof. By a balancing argument, p^* should be chosen to equate the return ratio (offline to online) in the following two events:

- the posteriori maximum price p_{max} is $\geq p^*$ (the ratio will be M/p^*)
- $p_{max} < p^*$ (the ratio is p_{max}/m). It follows that $M/p^* = p^*/m$.

□

RPP is optimal for infinite and finite durations and when duration is known or unknown.

A dramatic improvement is obtained by using randomization (so, randomization is advantageous for online search).

For simplicity, assume that $\varphi = 2^k$ for some integer k. For $0 \leq i \leq k-1$, RPP_i denotes the deterministic RPP with reservation price $m2^i$.

EXPO: Choose RPP_i with probability $1/k$.

Theorem 3 *EXPO is $c(\varphi)\log\varphi$-competitive, where $c(\varphi) \to 1$ when $\varphi \to \infty$.*

The bound of the above theorem holds even if the player does not know the values of m and M and knows only φ. Here, however, RPP_i has reservation price $p_1 2^i$.

Threat-Based Policy

Let c be any competitive ratio that can be attained by some one-way trading algorithm. First, assume that c is known to the trader.

The threat-based algorithm consists of two rules.

1. Consider converting dollars to won only when the current rate is the highest seen so far.
2. When converting dollars, convert just enough to ensure that a competitive ratio c would be obtained if an adversary dropped the exchange rate to the minimum possible rate and kept it there throughout the game (threat).

How to follow Rule 2.

We describe the optimal threat-based algorithm THREAT.

- Assume a known duration n and known m and M.
- Assume that the optimal competitive ratio c^* is known.

By Rule 1, we can assume that the exchange rate sequence consists of an initial segment (of length $k < n$) of successive maxima. In order to realize a threat, the adversary chooses k and $p_{k+1} = \cdots = p_n = m$.

For $0 \le i \le n$, let D_i and W_i be the number of remaining dollars and the number of accumulated won, respectively, immediately after the ith period. Initially, $D_0 = 1$ and $W_0 = 0$.

Let $s_i = D_{i-1} - D_i$ be the number of dollars traded at the ith period, $1 \le i \le n$. Hence, $W_i = \sum_{j=1}^{i} s_j p_j$.

According to Rule 2, s_i must be chosen such that for any $1 \le i \le n$,

$$\frac{p_i}{W_i + mD_i} \le c^*, \tag{1}$$

where $W_i + mD_i$ is the return of THREAT if an adversary drops the exchange rate to m and p_i is the return of OPT in such a case.

Since $W_i = W_{i-1} + s_i p_i$ and $D_i = D_{i-1} - s_i$ in (1) and we take the minimum amount for s_i, we get,

$$s_i = \frac{p_i - c^*(W_{i-1} + mD_{i-1})}{c^*(p_i - m)}. \tag{2}$$

From (2) at $i = 1$,

$$s_1 = \frac{1}{c^*}\frac{p_1 - mc^*}{p_1 - m}.$$

From (2) with $W_{i-1} + mD_{i-1} = p_{i-1}/c^*$ (from (1)),

$$s_i = \frac{1}{c^*}\frac{p_i - p_{i-1}}{p_i - m}$$

for $i > 1$.

Theorem 4 *Let $c_n^*(m, M)$ denote the optimal competitive ratio for the n-day game. Then $c_n^*(m, M)$ is the unique root of the equation*

$$c = n\left(1 - \left(\frac{m(c-1)}{M-m}\right)^{1/n}\right).$$

Trading with unknown duration: $c_\infty^*(m, M)$ is the unique root of the equation

$$c = \ln \frac{M-m}{m(c-1)}.$$

One-way trading in which the trader knows only the global fluctuation ratio $\varphi = M/m$ but not m and M. For duration n,

$$c_n^*(\varphi) = \varphi\left(1 - (\varphi - 1)\left(\frac{\varphi - 1}{\varphi^{n/(n-1)} - 1}\right)^n\right).$$

For unknown duration,

$$c_\infty^*(\varphi) = \varphi - \frac{\varphi - 1}{\varphi^{1/(\varphi-1)}}.$$

Table: Competitive ratios for search and one-way trading algorithms (unknown duration)

	value of φ					
algorithm	1.5	2	4	8	16	32
RPP (m, M known)	1.22	1.41	2	2.82	4	5.65
EXPO (only φ known)	1.5	2	2.66	3.42	4.26	5.16
THREAT (only φ known)	1.27	1.50	2.11	2.80	3.53	4.28
THREAT (m, M known)	1.15	1.28	1.60	1.97	2.38	2.83

Portfolio Selection

A market of s securities (assets) - stocks, bonds, foreign currencies, or commodities.

Let $P_i = (p_{i1}, \ldots, p_{is})$ denote a vector of prices, where p_{ij} is the number of units of the jth security that can be bought for one dollar at the start of the ith period. The local currency (cash) may or may not be one of the s securities. The price of case is constantly equal to 1.

The change in security prices during the ith period is represented as a column vector $X_i = (x_{i1}, \ldots, x_{is})$, where

$$x_{ij} = \frac{p_{ij}}{p_{(i+1)j}}.$$

The quantity x_{ij} is called the relative price of security j for the ith period. Thus, an investment of d dollars in the jth security at the start of the ith period yields dx_{ij} dollars by the end of the ith

period. Any sequence of price vectors or relative price vectors is called a *market sequence*.

An investment in the market, or *portfolio*, is specified as the proportion of dollar wealth currently invested in each of the s securities, i.e., $B = (b_1, \ldots, b_s)$, where $b_i \geq 0$ and $\sum_i b_i = 1$.

Consider a portfolio $B_1 = (b_{11}, \ldots, b_{1s})$ invested at the start of the 1st period. By the start of the 2nd period, this portfolio yields

$$B_1^t \cdot X_1 = \sum_{j=1}^{s} b_{1j} x_{1j}$$

dollars per initial dollar invested. At this stage the investment can be cashed and adjusted by reinvesting the entire current wealth in some other proportion B_2, etc.

Assume an initial wealth of \$1. Then the compounded return of a sequence of portfolios $B = (B_1, \ldots, B_n)$ with respect to a market

sequence of relative prices $X = (X_1, \ldots, X_n)$ is

$$R(B, X) = \prod_{i=1}^{n} B_i^t \cdot X_i.$$

A *portfolio selection algorithm* is any sequence of portfolios specifying how to reinvest the current wealth from period to period. (Transaction fees are ignored in this model.)

Trading strategies

- Buy-and-hold: Typically mutual fund managers select and buy some portfolio and then hold it for a relatively long time.
- Market timing: Some financial agents use aggressive strategies that buy and sell securities very frequently.

Constant Rebalancing

Constant rebalancing: a fixed portfolio $B = (b_1, \ldots, b_s)$ is used for each trading period.

This constant rebalancing algorithm CR_B is a market timing strategy.

Optimal offline constant rebalancing algorithm CR-OPT: CR-OPT $= CR_{B^*}$, where for a market sequence $X = (X_1, \ldots, X_n)$ the fixed portfolio B^* used by CR-OPT is

$$B^* = \arg\max_B \prod_{1 \le i \le n} B \cdot X_i.$$

Consider a market consisting of cash and one stock. Suppose that the relative prices of the stock follow the sequence: $\frac{1}{2}$, 2, $\frac{1}{2}$, 2, ...

The optimal offline BAH strategy, which buys the stock after the 1st period and sells after any even period, will double its investment.

Calculate the return of CR_B with $B = (\frac{1}{2}, \frac{1}{2})$.

- After 1st period, price $(1, \frac{1}{2})$. value $(\frac{1}{2}, \frac{1}{4})$ - total $\frac{3}{4}$
- After rebalancing, value $(\frac{3}{8}, \frac{3}{8})$.
- After 2nd period, price $(1, 2)$. value $(\frac{3}{8}, \frac{6}{8})$ - total $\frac{9}{8}$

After n periods, CR_B exponentially increases its initial investment by a factor of $(\frac{9}{8})^{n/2}$.

Statistical Adversaries

Two-way trading

fixed-fluctuation model:

- All price relatives x_i are in $\{\alpha, \alpha^{-1}\}$, where $\alpha > 1$.
- Add the restriction that each feasible sequence of price relatives is of length n and the number of downward (i.e., profitable) α^{-1} fluctuations is exactly k, $0 \leq k \leq n$. Hence the number of upward α fluctuations is $n - k$.
- Call the adversary that produces such feasible sequences the (α, n, k) adversary.
- The optimal offline return for each feasible market sequence is exactly $\phi = \alpha^k$.

There is an optimal online algorithm S^{**} against the (α, n, k) adversary.

- Whenever the market is stable, S^{**} performs remarkably well.
- Even if the market exhibits a slight unfavorable trend, S^{**} still yields exponential returns.

So, how to trade?

Risk Management

To manage the risk associated with decision making under uncertainty means to control the inherent tradeoff between risk and reward (e.g., risk-free 10% gain vs. 100% gain with risks)

al-Binali's framework: extend pure competitive analysis by introducing two ingredients: risk and forecast.

- For an online algorithm alg, $C(alg)$ denotes the competitive ratio of alg.
- Let $c^* = \inf_{alg} C(alg)$ be the optimal competitive ratio for this problem.
- Define the risk $R(alg)$ to be $C(alg)/c^*$.
- Define a forecast as any subset F of the allowable input sequences. For example, the set of market sequences allowable for the (α, n, k) adversary is a forecast.

The online player specifies a risk tolerance T. This means that the player is willing to use only algorithms from the set $\mathcal{T} = \{alg : R(alg) \leq T\}$.

Fix a forecast F. An optimal algorithm in this framework is an algorithm from $\mathcal{T}$ that minimizes the competitive ratio, restricted to input sequences from F.

$$C_F(alg) = \sup_{\sigma \in F} \frac{OPT(\sigma)}{alg(\sigma)}.$$

An optimal T-tolerant algorithm alg^* with respect to a forecast F satisfies

$$C_F(alg^*) = \inf_{alg} \{C_F(alg) : alg \in \mathcal{T}\}.$$

- By posing reasonable forecasts the player can boost performance significantly as long as input sequences conform to the forecast.
- Regardless of forecast, the player always keeps the risk within

the desired tolerance.

One-way trading with known duration n and known m and M.

Fix any $\Delta \in [0, M - m]$ and consider the forecast F: the exchange rate will increase to at least $m + \Delta$ within the trading period:

$$F = \{(p_1, \ldots, p_n) : \exists i \text{ such that } p_i \geq m + \Delta\}.$$

For a tolerance T, the optimal T-tolerant algorithm is the following two-stage threat-based algorithm $\textsc{Threat}_T$.

- In the first conservative stage, $\textsc{Threat}_T$ operates under the threat that the prediction is incorrect and attempts to achieve a competitive ratio of Tc^*.
- If the forecast comes true (i.e., the exchange rate swings above $m + \Delta$), $\textsc{Threat}_T$ first computes the optimal restricted ratio $\inf_{alg} C_F(alg)$ and now trades, using this restricted ratio, under the threat that the exchange rate drops to m.

Table: Restricted competitive ratio c_F^*

T	j	c_F^*
1.01	10	1.140
1.01	20	1.143
1.05	10	1.114
1.05	20	1.115
1.1	10	1.0979
1.1	20	1.0975

$\Delta = 0.35$, $m = 1$, $M = 1.5$, $n = 30$, $p_j \geq m + \Delta$.