OCO Universal Portfolio Algorithms summary

1 Introduction

This document summarizes modern online algorithms for the universal portfolio selection problem, with an emphasis on Online Newton Step.

2 Notations

1. K: set of feasible actions
2. $\mathbf{x}_t \in K$: action at time t
3. $l_t(\mathbf{x}) : K \mapsto \mathbb{R}$: loss function at time t
4. $\Pi_K(\mathbf{y}) = \arg\min_{\mathbf{x}\in K} \frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_2^2$: Euclidean projection of $\mathbf{y}$ to set K w.r.t matrix A.
5. $\Pi_K^A(\mathbf{y}) = \arg\min_{\mathbf{x}\in K} (\mathbf{y}-\mathbf{x})^T A(\mathbf{y}-\mathbf{x})$: Euclidean projection of $\mathbf{y}$ to set K
6. $Regret_T(\mathbf{u}) = \sum_{t=1}^T l_t(\mathbf{x}_t) - l_t(\mathbf{u})$: regret with respect to a fixed, arbitrary competitor $\mathbf{u}$
7. L: constant for L-Lipschitz continuity
8. D: diameter of a set, i.e. $D \geq \|\mathbf{y}-\mathbf{x}\| \quad \forall \mathbf{y},\mathbf{x} \in K$
9. $x_t(i)$: ith component of vector $\mathbf{x}$ at timestep t

3 Universal Portfolio theory

In universal portfolio theory, we make minimal assumptions about the market dynamics with a relatively simple model. Define $W_t \in \mathbb{R}$ as the wealth possessed at time t, $x_t \in \Delta_n$ as the distribution of wealth over n assets, where Δ_n is the n-simplex, and r_t as the price relatives, i.e. the ratio of of assets' price between consecutive time steps:

$$r_t(i) = \frac{i\text{th asset's price at time } t+1}{i\text{th asset's price at time } t}$$

Therefore, the possessed wealth at time T is:

$$W_T = W_1 \prod_{t=1}^T r_t^T x_t$$

To express the above as an OCO problem, we take the logarithm and obtain:

$$\log W_T = \log W_1 + \sum_{t=1}^T \log(r_t^T x_t)$$

$$\log \frac{W_T}{W_1} = \sum_{t=1}^T \log(r_t^T x_t)$$

$$-\log \frac{W_T}{W_1} = -\sum_{t=1}^T \log(r_t^T x_t)$$

and the objective now is to minimize $-\sum_{t=1}^{T}\log(r_t^T x_t)$. Thus, the regret of this problem is:

$$Regret_T(u) = \sum_{t=1}^{T} \log(r_t^T u) - \log(r_t^T x_t)$$

3.1 Constant rebalancing portfolio

Just like analyzing any OCO algorithms, we compare the algorithm of interest to a fixed competitor u. This fixed competitor, commonly the constant rebalancing portfolio (CRP) is a rather naive strategy, where one maintain a constant distribution of wealth at each timestep, which gives the name constant and rebalancing. **Portfolio algorithms that achieve sublinear regret w.r.t. CRP is considered universal.** A more rigorous definition and the motivation of using CRP as a benchmark is given in the following sections.

One classic, yet extreme situation that demonstrates CRP's advantage is as follows. Consider two stocks, one doubles on even days and halves on odd days, the other stock does the same but in opposite days, i.e.:

$$r_{1:T}(1) = 2, \frac{1}{2}, 2, ...$$
$$r_{1:T}(2) = \frac{1}{2}, 2, \frac{1}{2} ...$$

By allocating equal weights to both stocks, and rebalance the wealth after each time step, the wealth grows $0.5 * 2 + 0.5 * 0.5 = 1.25$ times per timestep.

4 Benchmark algorithms

Benchmark algorithms are naive algorithms that one compares an algorithm to.

4.1 Buy and Hold

With Buy and Hold (BAH) strategy, one simply invests in a pool of assets and wait until the investment period T is over. Therefore, the wealth gain is:

$$W_T = W_1 x^T \bigodot_{t=1}^{T} r_t$$

where $\odot$ denotes element-wise product.

4.2 Best stock

The best stock strategy is a special case of BAH, where we allocate all of the wealth to one single stock that is the best in hindsight, that is:

$$x^* = \arg\max_{i=1,\ldots,n} e_i^T \bigodot_{t=1}^{T} r_t$$

where e_i is the ith unit vector.

4.3 Constant rebalancing portfolio

As a benchmark, we are interested in comparing a strategy of interest to the best CRP (BCRP), i.e.

$$x^* = \arg\max_x \prod_{t=1}^{T} r_t^T x$$

It might not be obvious why BCRP is commonly used as a benchmark, we note the following properties of BCRP:

Denote $W_T(x)$ as the wealth at time T by taking strategy x,

First, BCRP exceeds the Best stock strategy, i.e.

$$W_T(x^*) \geq \max_{i=1,..,n} W_T(e_i)$$

Proof: BCRP and Best stock maximizes over the whole simplex and only the vertices respectively.

Second, BCRP exceeds geometric mean (value line), i.e.

$$W_T(x^*) \geq (\prod_{i=1}^{n} W_T(e_i))^{\frac{1}{n}}$$

Proof: $W_T(x^*) \geq \max_i W_T(e_i) \geq W_T(e_i)\ \forall i = 1, ..., n$

Third, BCRP exceeds arithmetic mean, i.e.

$$W_T(x^*) \geq \sum_{i=1}^{n} \alpha_i W_T(e_i)$$

where $\alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i = 1$.

Proof: $W_T(x^*) = \max_{\alpha \in \Delta_n} \sum_{i=1}^{n} \alpha_i W_T(e_i)$

Lastly, $W(x^*)$ by BCRP is clearly independent of the order $x_1, ..., x_T$

5 Follow the winner approaches

This class of algorithms gradually adjust how much they value a particular strategy based on their past performance. One of the classic examples is the Hedge algorithm, where it penalizes a strategy proportional to the loss incurred, or put it another way, a more successful strategy would be weighted more in the long run. It should therefore be clear that online algorithms derived from the meta algorithm Follow-the-leader (FTL) also fall into this class of method, such as the vanilla FTL, FTRL (regularized), Hedge etc..

Below we introduce some seemingly different algorithms, that are actually closely related to FTL.

5.1 Cover's universal portfolio algorithm

The idea of universal portfolio algorithms introduced above was first proposed by Cover [1]. In his paper he also introduced an universal algorithm that has the same growth rate of BCRP asymptotically. Such strategy does not rely on future information, yet it is able to achieve comparable performance with BCRP, which makes the optimal decision in hindsight. As appealing as it seems, it suffers from high computational cost to implement.

Cover's algorithm closely resembles the Hedge algorithm for the expert problem. In the latter case, to quickly recap, we have n experts that are being allocated with corresponding weights $w(i)$ $i = 1, .., n$ based on how much loss they incurred previously. These weights then determine the probability $x(i)$ of selecting ith expert by $x(i) = w(i)/\sum_{j=1}^{n} w(j)$.

In contrast, the feasible set of allocating assets in portfolio selection is the simplex Δ_n. In Cover's algorithm, we have infinitely many 'experts' $x \in \Delta_n$. Up to the tth timestep, following the 'expert' x would allow us to gain wealth of $W_t(x) = \prod_{i=1}^{t} r_t^T x$ times (assume $W_1 = 1$). The algorithm then select the next strategy by:

$$x_{t+1} = \frac{\int_{x \in \Delta_n} x W_t(x) dx}{\int_{x \in \Delta_n} W_t(x) dx}$$

which is simply one that is weighted by the gain of all feasible distributions. At this point the similarity between Cover's and the Hedge algorithm should be obvious. We note that the gain $\prod_{i=1}^{t} r_t^T x$ is equivalent to the loss $\exp(-\sum_{i=1}^{t} \log(r_t^T x))$, and the latter expression is also used for penalizing loss or as a weight of an expert in the Hedge algorithm. Therefore, Cover's algorithm can be considered as a continuous case of the Hedge algorithm, where we have infinitely many 'experts'.

5.2 Online Newton step (ONS)

We know that first order online algorithms are able to achieve $O(\sqrt{T})$ regret, or $O(\log T)$ for strongly convex losses. As we shall see, the loss function of portfolio selection is not strongly convex, we therefore need a new approach to achieve better regret than $\sqrt{T}$.

In ONS we update the iterates by:

$$x_{t+1} = \Pi_{x \in K}^{A_t}(A_t^{-1} b_t)$$

where A_t, b_t would be defined foramlly in latter sections. The intuitive way to understand ONS is through offline Newton's method. The close-form solution of minimizing the quadratic function $x^T Ax - 2b^T x$ is $x^* = A^{-1}b$, which is also the case in online settings, except the matrices A, b are time-dependent and we solve a slightly different optimization problem at every step. We finally project the solution to the feasible set w.r.t the matrix A_t if it is infeasible.

5.2.1 Exp-Concave functions

We first introduce a class of functions, namely exp-concave functions.

Definition 1. *A convex function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is α-exp-concave over $K \subset \mathbb{R}$ if the function g is concave, where $g : K \mapsto \mathbb{R}$ is defined as*

$$g(x) = e^{-\alpha f(x)}$$

Lemma 1. *Let $f : K \mapsto \mathbb{R}$ be an α-exp-concave function, and denote D, L as the diameter of K and the bound of (sub)gradients of f respectively. The following bound holds for $\gamma \leq \frac{1}{2}\min\{\frac{1}{LD}, \alpha\}$ and for all $x, y \in K$*

$$f(x) \geq f(y) + \nabla f(y)^T (x - y) + \frac{\gamma}{2}(x - y)^T \nabla f(y) \nabla f(y)^T (x - y)$$

Proof. Let $a(x) = x^{2\gamma/\alpha}$ and $b(x) = e^{-\alpha f(x)}$, the composition $h(x) = a \circ b = e^{-2\gamma f(x)}$ is also concave since $a(x)$ is a nondecreasing function for $2\gamma \leq \alpha$ and $b(x)$ is concave.

Then by concavity of $h(x)$, we have

$$h(x) \leq h(y) + \nabla h(y)^T (x - y)$$

that is,

$$\begin{aligned} e^{-2\gamma f(x)} &\leq e^{-2\gamma f(y)} - 2\gamma e^{-2\gamma f(y)} \nabla f(y)^T (x - y) \\ e^{-2\gamma f(x)} &\leq e^{-2\gamma f(y)} (1 - 2\gamma \nabla f(y)^T (x - y)) \\ f(x) &\geq f(y) - \frac{1}{2\gamma} \log(1 - 2\gamma \nabla f(y)^T (x - y)) \end{aligned}$$

By construction we have $\gamma \leq \frac{1}{2}\min\{\frac{1}{LD}, \alpha\}$, which implies $2\gamma LD \leq 1$, and $|2\gamma \nabla f(y)^T (x-y)| \leq 2\gamma LD$. Alltogether we have $|2\gamma \nabla f(y)^T (x - y)| \leq 2\gamma LD \leq 1$. Then, $-\log(1 - z) \geq z + \frac{1}{4}z^2$ holds for $z \geq -1$ by second order taylor approximation of $-\log(1 - z)$ at $z = 0$, the lemma is then obtained by setting $z = 2\gamma \nabla f(y)^T (x - y)$.

It might not be immediately obvious how exp-concave functions are related to online portfolio selection. First of all, one can easily check that the loss function in this case, i.e.$-\log(r_t^T x_t)$ is 1-exp-concave. Second, to give more intuition about this class of function, it is a subset of convex functions, but a superset of strongly convex functions:

$$\text{Strongly convex functions} \subset \text{Exp-concave functions} \subset \text{Convex functions}$$

The concavity of $g(x)$ implies $\nabla^2 g(x) \preceq 0$, that is $\alpha e^{-\alpha f(x)}[\alpha \nabla f(x) \nabla f(x)^T - \nabla^2 f(x)] \preceq 0$, thus,

$$\nabla^2 f(x) \succeq \alpha \nabla f(x) \nabla f(x)^T$$

We take a closer look at the term $\nabla f(x) \nabla f(x)^T$. This is a rank-1 PSD matrix which does not fulfill strong convexity, since in the worst case it is possible that $x^T(\nabla f(x)\nabla f(x)^T - \alpha I)x < 0$. Nevertheless, $\nabla f(x)\nabla f(x)^T$ is large in the direction of gradients, i.e. $x^T \nabla f(x) \nabla f(x)^T x$ is large when $x = \nabla f(x)$. This implies there is significant convexity in a particular direction but not everywhere, and therefore it is a weaker condition than strong convexity. This motivates the use of algorithms specially designed for this class of functions, namely ONS, instead of first order algorithms that has $O(\log T)$ regret only for strongly convex functions.

5.2.2 ONS from FTL

It might not be obvious but ONS actually can be derived from the FTL algorithm. Recall in FTL, we udpate the iterates by simply:

$$x_{t+1} = \arg\min_{x \in K} \sum_{i=1}^{t} l_i(x)$$

By approximating the loss function with Lemma 1, we obtain the Follow-the 'approximate' leader (FTAL) instead, i.e.

$$x_{t+1} = \arg\min_{x \in K} \sum_{i=1}^{t} l_i(y_i) + \nabla l_i(y_i)^T (x - y_i) + \frac{\gamma}{2}(x - y_i)^T \nabla l_i(y_i) \nabla l_i(y_i)^T (x - y_i)$$

Removing the constants and multiplying the expression with $2/\gamma$, we have

$$\begin{aligned} &\arg\min_{x \in K} \sum_{i=1}^{t} x^T \nabla l_i(y_i) \nabla l_i(y_i)^T x - 2(\nabla l_i(y_i) \nabla l_i(y_i)^T y_i - \frac{1}{\gamma} \nabla l_i(y_i))x \\ &= \arg\min_{x \in K} x^T A_t x - 2 b_t^T x \end{aligned}$$

where we let $A_t = \sum_{i=1}^{t} \nabla l_i(y_i) \nabla l_i(y_i)^T$ and $b_t = \sum_{i=1}^{t} \nabla l_i(y_i) \nabla l_i(y_i)^T y_i - \frac{1}{\gamma} \nabla l_i(y_i)$. The above optimization problem can alternatively be expressed as:

$$\arg\min_{x \in K} (x - A_t^{-1} b_t)^T A_t (x - A_t^{-1} b_t) - b_t^T (A_t^{-1} b_t)$$

which is exactly the update of ONS: $x_{t+1} = \Pi_{x \in K}^{A_t}(A_t^{-1} b_t)$.

5.2.3 Regret bound of ONS

To show the regret bound of ONS, we first need a few lemmas.

Lemma 2. *The regret bound of FTL is upper bounded by the summation of differences in consecutive actions:*

$$Regret_T(u) = \sum_{t=1}^{T} l_t(x_t) - l_t(u) \leq \sum_{t=1}^{T} l_t(x_t) - l_t(x_{t+1})$$

Proof by induction. Since $l_t(x_t)$ is present in both sides, it suffices to show that

$$\sum_{t=1}^{T} l_t(u) \geq \sum_{t=1}^{T} l_t(x_{t+1})$$

For $T = 1$, the inequality holds trivially since $x_2 = \arg\min l_1(x)$ and u is an arbitrary action, so $l_1(u) \geq l_1(x_2)$.

Inductive hypothesis: We assume the inequality holds for $T = 2, ..., k-1$, we shall prove that it also holds for $T = k$:

$$\begin{aligned}\sum_{t=1}^{k-1} l_t(u) &\geq \sum_{t=1}^{k-1} l_t(x_{t+1}) \quad \text{by I.H.}\\ \sum_{t=1}^{k-1} l_t(u) + l_k(x_{k+1}) &\geq \sum_{t=1}^{k-1} l_t(x_{t+1}) + l_k(x_{k+1})\\ &= \sum_{t=1}^{k} l_t(x_{t+1})\end{aligned}$$

Since u is an arbitrary action, setting $u = x_{k+1}$ completes the proof.

Then define the R.H.S of inequality in Lemma 1 as $\tilde{f}_t(x)$, we have $f_t(x) \geq \tilde{f}_t(x)\ \forall x \in K$, and particularly $f_t(x_t) = \tilde{f}_t(x_t)$.

Next, we upper bound the regret of l with $\tilde{l}$:

Lemma 3. *Denote $\tilde{l}_t$ as the approximation of f_t in Follow-The-Approximate-Leader algorithm, the following inequality holds:*

$$\sum_{t=1}^{T} \tilde{l}_t(x_t) - \tilde{l}_t(u) \geq \sum_{t=1}^{T} l_t(x_t) - l_t(u)$$

Proof. The above inequality immediately follows from the fact that $l_t(x) \geq \tilde{l}_t(x)$ and $l_t(x_t) = \tilde{l}_t(x_t)$.

Combining both Lemma 2 and 3, we have:

$$l_t(x_t) - l_t(u) \leq \tilde{l}_t(x_t) - \tilde{l}_t(u) \leq \tilde{l}_t(x_t) - \tilde{l}_t(x_{t+1})$$

So it suffices to bound $\sum_{t=1}^{T} \tilde{l}_t(x_t) - \tilde{l}_t(x_{t+1})$.

Lemma 4. *Let A be a positive definite matrix and x be a vector such that $A - xx^T \succ 0$, then*

$$x^T A^{-1} x \leq \log\left[\frac{|A|}{|A - xx^T|}\right]$$

Proof. Let $B = A - xx^\top$. For any positive definite matrix C, let $\lambda_1(C), \lambda_2(C), \ldots, \lambda_n(C)$ be its (positive) eigenvalues.

$$\begin{aligned} x^\top A^{-1} x \quad &= \operatorname{Tr}\left(A^{-1} x x^\top\right)\\ &= \operatorname{Tr}\left(A^{-1}(A-B)\right)\\ &= \operatorname{Tr}\left(A^{-1/2}(A-B)A^{-1/2}\right)\\ &= \operatorname{Tr}\left(I - A^{-1/2}BA^{-1/2}\right)\\ &= \textstyle\sum_{i=1}^{n}\left[1 - \lambda_i\left(A^{-1/2}BA^{-1/2}\right)\right] \qquad \because \operatorname{Tr}(C) = \sum_{i=1}^{n}\lambda_i(C)\\ &\leq -\textstyle\sum_{i=1}^{n}\log\left[\lambda_i\left(A^{-1/2}BA^{-1/2}\right)\right] \qquad \because 1 - x \leq -\log(x)\\ &= -\log\left[\textstyle\prod_{i=1}^{n}\lambda_i\left(A^{-1/2}BA^{-1/2}\right)\right]\\ &= -\log\left|A^{-1/2}BA^{-1/2}\right| = \log\left[\frac{|A|}{|B|}\right] \qquad \because \textstyle\prod_{i=1}^{n}\lambda_i(C) = |C| \end{aligned}$$

Lemma 5. *Let A_0 be a positive definite matrix, and for $t \geq 1$, let $A_t = \sum_{\tau=1}^{t} v_t v_t^\top$ for some vectors $v_1, v_2, \ldots, v_t$. Then the following inequality holds:*

$$\sum_{t=1}^{T} v_t^\top \left(A_t + A_0\right)^{-1} v_t \leq \log\left[\frac{|A_T + A_0|}{|A_0|}\right]$$

Proof. By Lemma 4, we have

$$\begin{aligned}
\sum_{t=1}^{T} v_t^\top (A_t + A_0)^{-1} v_t &\leq \sum_{t=1}^{T} \log\left[\frac{|A_t + A_0|}{\left|A_t + A_0 - v_t v_t^\top\right|}\right] \\
&= \log\left[\frac{|A_1 + A_0|}{|A_0|}\right] + \sum_{t=2}^{T} \log\left[\frac{|A_t + A_0|}{|A_{t-1} + A_0|}\right] && \because A_1 = v_1 v_1^T \\
&= \log\left[\frac{|A_t + A_0|}{|A_0|}\right] && \text{by telescopic sum}
\end{aligned}$$

We now proceed to bound $\sum_{t=1}^{T} \tilde{l}_t(x_t) - \tilde{l}_t(x_{t+1})$. For simplicity, denote

1. $L_t(x) = \sum_{i=1}^{t-1} \tilde{l}_i(x)$
2. $\nabla_t = \nabla l_t(x_t) = \nabla \tilde{l}_t(x_t)$
3. $\Delta \nabla L_t(x_t) = \nabla L_{t+1}(x_{t+1}) - \nabla L_t(x_t)$

By convexity we have

$$\tilde{l}_t(x_t) - \tilde{l}_t(x_{t+1}) \leq -\nabla \tilde{l}_t(x_t)^T (x_{t+1} - x_t) = -\nabla_t^T \Delta x_t \tag{1}$$

, and we will formulate the bound for the R.H.S..

The gradient $\nabla L_{t+1}(x)$ can be written as

$$\nabla L_{t+1}(x) = \sum_{i=1}^{t} \nabla l_i(x_i)^T + \gamma \nabla l_i(x_i) \nabla l_i(x_i)^T (x - x_i)$$

Thus,

$$\begin{aligned}
\nabla L_{t+1}(x_{t+1}) - \nabla L_{t+1}(x_t) &= \gamma \sum_{i=1}^{t} \nabla l_i(x_i) \nabla l_i(x_i)^T (x_{t+1} - x_t) \\
&= \gamma A_t \Delta x_t
\end{aligned} \tag{2}$$

where we denote $A_t = \sum_{i=1}^{t} \nabla l_i(x_i) \nabla l_i(x_i)^T$. We note that the L.H.S of (2) is equivalent to $\Delta \nabla L_t(x_t) - \nabla_t$, so adding $\epsilon\gamma\Delta x_t$ to both sides we obtain:

$$\Delta \nabla L_t(x_t) - \nabla_t + \epsilon\gamma\Delta x_t = \gamma(A_t + \epsilon I)\Delta x_t \tag{3}$$

Premultiplying both sides with $\frac{-1}{\gamma} \nabla_t^T (A_t + \epsilon I)^{-1}$, we then have:

$$\begin{aligned}
-\nabla_t^T \Delta x_t &= \frac{-1}{\gamma} \nabla_t^T (A_t + \epsilon I)^{-1} \Big[\Delta \nabla L_t(x_t) - \nabla_t + \epsilon\gamma\Delta x_t\Big] \\
&= \frac{-1}{\gamma} \nabla_t^T (A_t + \epsilon I)^{-1} \Big[\Delta \nabla L_t(x_t) + \epsilon\gamma\Delta x_t\Big] + \frac{-1}{\gamma} \nabla_t^T (A_t + \epsilon I)^{-1} \nabla_t
\end{aligned}$$

which is the expression from (1) that we are trying to formulate an upper bound. We then bound the R.H.S. of the above separately.

To bound $\frac{-1}{\gamma} \nabla_t^T (A_t + \epsilon I)^{-1} \Big[\Delta \nabla L_t(x_t) + \epsilon\gamma\Delta x_t\Big]$, we first note that by optimality conditions, we have:

$$\nabla L_t(x_t)^T (x - x_t) \geq 0 \tag{4}$$

for any $x \in K$. Applying (4) to time t and $t+1$, we have:

$$\begin{aligned}\nabla L_{t+1}(x_{t+1})^T(x_t - x_{t+1}) + \nabla L_t(x_t)^T(x_{t+1} - x_t) \geq 0 \\ \Delta \nabla F_t(x_t) \Delta x_t \leq 0\end{aligned}$$

We then add $\epsilon\gamma\|\Delta x_t\|^2$ to both sides:

$$\begin{aligned}\epsilon\gamma\|\Delta x_t\|^2 &\geq [\Delta\nabla F_t(x_t) + \epsilon\gamma\Delta x_t]^T \Delta x_t \\ &= \frac{1}{\gamma}[\Delta\nabla F_t(x_t) + \epsilon\gamma\Delta x_t]^T (A_t + \epsilon I)^{-1}[\Delta\nabla L_t(x_t) - \nabla_t + \epsilon\gamma\Delta x_t] && \text{by (3)} \\ &= \frac{1}{\gamma}[\Delta\nabla F_t(x_t) + \epsilon\gamma\Delta x_t]^T (A_t + \epsilon I)^{-1}[\Delta\nabla L_t(x_t) + \epsilon\gamma\Delta x_t] \\ &\quad - \frac{1}{\gamma}[\Delta\nabla F_t(x_t) + \epsilon\gamma\Delta x_t]^T (A_t + \epsilon I)^{-1}\nabla_t \\ &\geq -\frac{1}{\gamma}[\Delta\nabla F_t(x_t) + \epsilon\gamma\Delta x_t]^T (A_t + \epsilon I)^{-1}\nabla_t && \because A_t + \epsilon I \succeq 0\end{aligned}$$

Therefore the first term is bounded by $\epsilon\gamma\|\Delta x_t\|^2 \leq \epsilon\gamma D^2$ since D is the diameter of set K.

As for the second term $\frac{-1}{\gamma}\nabla_t^T(A_t+\epsilon I)^{-1}\nabla_t$, from Lemma 4 and setting $A_0 = \epsilon I, v_t = \nabla_t, \epsilon = 1/\gamma^2 D^2 T$,

$$\begin{aligned}\frac{-1}{\gamma}\nabla_t^T(A_t + \epsilon I)^{-1}\nabla_t &\leq \frac{1}{\gamma}\log\left[\frac{|A_T + \epsilon I|}{|\epsilon I|}\right] \\ &\leq \frac{1}{\gamma}\log(\frac{L^2T + \epsilon}{\epsilon})^n \\ &= \frac{1}{\gamma}n\log(L^2T^2\gamma^2D^2 + 1) \\ &\leq \frac{2}{\gamma}n\log T\end{aligned}$$

where the second inequality follows from the bound of gradients, i.e. $\|\nabla_t\| \leq L$ and $A_t = \sum_{t=1}^T \nabla_t\nabla_t^T$, such that $|A_T + \epsilon I| \leq (L^2T + \epsilon)^n$, and the last inequality is due to the construction of $\gamma LD \leq \frac{1}{2}$ earlier in Lemma 1.

Alltogether, we have the regret bound:

$$\sum_{t=1}^T \tilde{l}_t(x_t) - \tilde{l}_t(x_{t+1}) \leq \frac{2}{\gamma}n\log T + \epsilon\gamma D^2 T = \frac{2}{\gamma}n\log T + \frac{1}{\gamma} = O(\log T)$$

5.3 Online meta optimization

A simple yet powerful extension [2] to the above algorithms is to ensemble them, i.e. follow the winner of the winners.

The main idea of meta optimization is to combine k 'base' algorithms which output decision variables $x_{t,1}, ..., x_{t,k}$ at time t. We then form a convex hull of the decision variables and obtain the actual decision $\sum_{i=1}^k w_i x_i$, $\sum_{i=1}^k w_i = 1$ from the convex combination of base algorithms. Weights allocated to each base algorithm are also decided base on the loss incurred by the algorithm, similar to weight update in Hedge.

Since the convex hull of decision variables is a superset of the independent variables, it should be obvious that the performance of the ensemble method is at least as good as the base methods. These base algorithms could be any online algorithms, but at least one of the base methods should be universal in order for the ensemble method to be universal. Empirical results have shown substantial improvement in portfolio selection by the ensemble method comprised of Cover's universal portfolio, Hedge and ONS versus the base algorithms individually.

6 Pattern-matching approaches (PM)

This type of algorithms utilize the past iterations that follows a similar trend and aid the decision making for the current step.

Previously discussed approaches assume the sequential events are i.i.d., whereas PM methods assume the next price relative r_{t+1} follows a conditional distribution on past observations $r_1, ..., r_t$, which is a reasonable assumption in stock markets. Therefore, the motivation of PM methods is to maximize the conditional expected gain. This is done by two steps, first we obtain similar sequences (samples), then make decision making based on these observations.

6.1 Sample selection

Denote the window of price relatives of interest as w, and r_{t-w+1}^t as the price relatives from time $t-w+1$ to t, i.e. $\{r_{t-w+1}, ..., r_t\}$. We are interested in finding a window of price relatives with similar trends, so intuitively speaking, we 'slide' the window to find a sequence that satisfies some similarity measure. Formally speaking, denote $C(w, r_{t-w+1}^t)$ as the set that contains indexes that are after similar trends, and S as the similar set:

$$C(w, r_{t-w+1}^t) = \{w < i < t+1 : r_{i-w+1}^{i-1} \in S\}$$

A several ways to quantify similarity were proposed. One way is to measure through Euclidean distance, that is:

$$S = \{r : \|r - r_{t-w+1}^t\| \leq \frac{c}{l}\}$$

where c and l are parameters to control the number of similar samples. Another way is by K-nearest neighbours (KNN):

$$S = \{r : r \text{ is among KNN of } r_{t-w+1}^t\}$$

Lastly, it can also be done by measuring correlation of the windows of price relatives:

$$S = \{x : \frac{Cov(r, r_{t-w+1}^t)}{\sigma(r)\sigma(r_{t-w+1}^t)} \geq \rho\}$$

where ρ is some defined correlation parameter.

6.2 Portfolio selection

Now that the similar set C is obtained, we select the portfolio for the next time step. One straightforward way is to minimize the expected loss of similar timesteps, i.e.

$$x_{t+1} = \arg\min \sum_{i \in C} p_i(-\log r_i^T x)$$

where p_i is the probability for the similar price relative x_i, one could use uniform or non-uniform distributions. In the former case, it reduces to minimizing the sum of similar losses.

To reduce computational complexity of the above expression, one could also use the semi-log-optimal strategy, which simply performs second order taylor approximates of the function $f(x) = \log x$ w.r.t. $x = 1$:

$$f(x) \approx x - 1 - \frac{1}{2}(x-1)^2$$

and resulting a similar portfolio selection update:

$$x_{t+1} = \arg\min \sum_{i \in C} -p_i f(r_i^T x)$$

References

[1] Thomas M Cover. Universal portfolios. *Mathematical finance*, 1(1):1–29, 1991.

[2] Puja Das and Arindam Banerjee. Meta optimization and its application to portfolio selection. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1163–1171, 2011.

[3] Francesco Orabona. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*, 2019.

[4] Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

[5] Bin Li and Steven CH Hoi. Online portfolio selection: A survey. *ACM Computing Surveys (CSUR)*, 46(3):1–36, 2014.

[6] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69:169–192, 2007.