# Online Portfolio Optimization

### Problem formulation

At each iteration $t \in [T]$, the decision maker decides the distribution $x_t \in \Delta_n$ of wealth over $n$ assets, where $\Delta_n = \{x \in \mathbb{R}^n_+, \sum_i x_i = 1\}$ is the n-dimensional simplex. The market returns a ratio vector $r_t \in \mathbb{R}^n_+$ such that the $i$th element, i.e. $r_t(i)$ is the price ratio for the i th asset between iterations t and t+1.

The resultant change of wealth between any iteration is therefore:

$$W_{t+1} = W_t \cdot r_t^T x_t$$

and at the $t$th iteration, the total wealth accumulated would be

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

we can therefore formulate an optimization problem that maximizes the objective function:

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

or more conveniently minimize the objective function:

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

this formulation fits the pattern of a OCO problem, where we let

$$f_t(x) = -\log(r_t^T x)$$

and the regret can be formulated as:

$$Regret_T = \max_{x' \in \Delta_n} \sum_{t=1}^{T} f_t(x') - \sum_{t=1}^{T} f_t(x_t)$$

## Background

### Online gradient descent

The update is simply done by:

$$
\begin{aligned}
y_{t+1} &= x_t - \eta_t \nabla f_t(x_t) \\
x_{t+1} &= \Pi_{x\in\Delta_n}(y_{t+1}) = \arg\min_{x\in\Delta_n} \frac{1}{2}||x - y_{t+1}||_2^2
\end{aligned}
$$

The regret is bounded by $\frac{3}{2}GD\sqrt{T}$ if we chooose the step sizes to be $\eta_t = \frac{D}{G\sqrt{t}}$, where $G$ is the parameter for Lipschitz continuity, and $D$ is the diameter of the set $\Delta_n$, i.e. $||x-y||_2 \leq D \quad \forall x, y, \in \Delta_n$, which is $D = \sqrt{2}$ for simplexes.

To find parameter $G$, we note that in our problem the norm of gradient is upper bounded by:

$$
\begin{aligned}
||\nabla f_t||_2 &= \frac{||r_t||_2}{r_t^T x_t} \\
&\leq \frac{||r_t||_2}{\min_i r_t(i)} \quad \text{by } \textstyle\sum_{i=1}^n x_i = 1
\end{aligned}
$$

so $G$ would be chosen such that $\max_t \frac{||r_t||_2}{\min_i r_t(i)} \leq G$

## Follow the leader (Linearized)

we update the iterates by

$$
x_t = \arg\min_{x\in\Delta_n} \sum_i^{t-1} f_i(x) = \arg\min_{x\in\Delta_n} \sum_i^{t-1} -\log(r_t^T x)
$$

we linearize the cost function by

$$
f_i(x) \approx f_i(x_i) + \nabla f_i(x_i)^T (x - x_i)
$$

and update the iterates by

$$
x_t = \arg\min_{x\in\Delta_n} \sum_i^{t-1} \nabla f_i(x_i)^T x = \arg\min_{x\in\Delta_n} \sum_i^{t-1} (\frac{-r_i}{r_i^T x_i})^T x
$$

since $f_i(x_i)$ and $\nabla f_i(x_i)^T x_i$ are constants

now that $x_t$ is a linear function so $x$ attains minimum when $x = e_j$ ($j$th unit vector) such that the $r_j$ is the minimum element of $r = \sum_i^{t-1} (\frac{-r_i}{r_i^T x_i})$ , i.e. behave greedily. This strategy could however could perform miserably when loss functions fluctuate, a regret of $O(T)$ can be incurred in the worst case.

## Exponentiated Gradient Descent

One of the variants based on the regularized FTL meta-algorithm. This time we update the iterates by

$$x_{t+1} = \arg\min_{x \in \Delta_n} \left\{ \eta \sum_{i}^{t} f_i(x) + R(x) \right\}$$

, with an additional regularization function that is assumed to be strongly convex.

In exponentiated GD, $R(x) = \sum_{i=1}^{n} x_i \log x_i$. With this choice of regularization function, the update can be computed with a closed form expression:

$$x_{t+1} = \frac{y_{t+1}}{||y_{t+1}||_1}, \quad y_{t+1}(i) = y_t(i) e^{-\eta \nabla f_t(i)} \text{ for } i = 1, ..., n$$

where $\eta = \sqrt{\frac{\log n}{2TG_\infty^2}}, ||\nabla_t||_\infty \leq G_\infty$ is a constant step size.

In our case we have $y_t(i) e^{-\eta \nabla f_t(i)} = y_t(i) \exp(\eta \frac{r_t(i)}{r_t^T x_t})$

With regularization, it stabilizes gradient descent algorithms and is able to achieve a regret of $\sqrt{2T \log n}$.

## Online Newton step

Strictly speaking the Online Newton step method is not a second order method, as only gradient information is being used. We use a pseudo-hessian $A_t$ during the update of iterates, and $A_t$ itself is computed by a rank-1 update: $A_t = A_{t-1} + \nabla_t \nabla_t^T$.

We then update the iterates:

$$y_{t+1} = x_t - \frac{1}{\gamma} A_t^{-1} \nabla_t$$
$$x_{t+1} = \Pi_K^{A_t}(y_{t+1}) = \arg\min_{x \in \Delta_n} \{||y_{t+1} - x_{t+1}||_{A_t}^2\}$$

Nevertheless, Online Newton step has a logarithmic regret for exp-concave loss functions.