Portfolio Optimization in a Big Data Context

Thierry Bazier-Matte

Summer 2015

Abstract

Following [1], we provide a portfolio optimization method based on machine learning methods.

1 Introduction

This document considers a two-asset portfolio, of which one is the risk-free asset, yielding a constant return rate R_f, and the other being a risky asset s, typically a stock, yielding a random return rate r_{st} for each period t. We suppose that each risky asset s can be decribed daily by an *information vector* x_{st} containing potentially useful information, such as technical, fundamental or news-related information. Furthermore, we assume that the allocation of each asset of the portfolio p_{st} can be fully determined using a *decision vector* q. The allocation rule is the folowing: $q^T x_{st}$ is allocated to the risky asset and $1 - q^T x_{st}$ is allocated to the risk-free asset. Over the period t, the portfolio p_{st} consisting of asset s will therefore yield a return rate of:

Maybe considerations about the length of the period should be added? For example, it's not specified what's the period length of R_f.

$$p_{st}(q) = r_{st} q^T x_{st} + (1 - q^T x_{st}) R_f. \tag{1}$$

The question we now wish to ask is how the decision vector q should be chosen. We assume we have access to a training dataset S_n, comprising of s different assets over t periods, such that $n = s \times t$.

What's the difference between having n points with having $s \times t$ points? For example, what if $s \gg t$ or the reverse?

2 Definitions and Bounds

2.1 Definitions Notation

Most of the following notation and defintions follow directly from [2].

Let S_n be a set of n vectors of $\boldsymbol{R}^p \times \boldsymbol{R}$ of the form:

$$S_n = \{(x_1, r_1), \ldots, (x_n, r_n)\}. \tag{2}$$

Each component of S_n is a tuple (x, r), where x is the information vector and r is the observed return rate.

Using S_n, we wish to create a decision vector $q_{S_n} \in \boldsymbol{R}^p$ from which we can make an investment decision when confronted with a random draw $d = (x, r)$.

Loss, cost and regret. We introduce the loss ℓ as being a function mapping from the space of the decision vectors and full observations (including market observations and the return of the asset) to a numerical quantity called the cost. Formally, given q a decision vector and a random draw $d = (x, r)$, the loss will be

$$\ell(q, d) = c(q(x), r) = c(q^T x, r). \tag{3}$$

Supposing an utility U, there are two different cost functions we can use. The first one, and perhaps the most obvious one is defined by

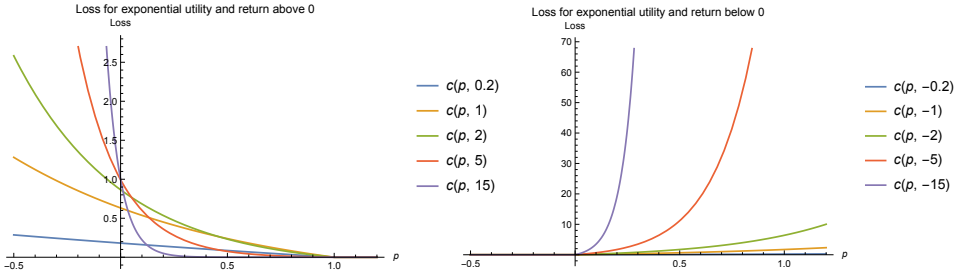
$$c(p, r) = -U(pr + (1 - p)R_f). \tag{4}$$

If the utility is piecewise linear, then the cost is not bounded, and so an infinite return would yield a negatively infinite cost. This means that risk-taking will be encouraged since risky position, ie. $|p| > 1$ can in fact yield a negative cost. On the other hand, with an unbounded utility, the algorithm minimizing the cost over the sample must have $\lambda > 0$, otherwise the problem might be unconstrained. **Unless we add a constant piece if $r < \bar{r}$.**

We can also define a new cost function that is always non-negative, that we shall call the regret ρ of the decision:

$$\rho(p, r) = \begin{cases} \lfloor U(r) - U(pr + (1 - p)R_f) \rfloor & \text{if } r > R_f \\ \lfloor U(R_f) - U(pr + (1 - p)R_f) \rfloor & \text{if } r \leq R_f, \end{cases} \tag{5}$$

where by $\lfloor \cdot \rfloor$ we mean $\max(., 0)$. In the case of exponential utilty, this cost function is actually equivalent to the one previously defined, but is quite different in the case of an unbounded utility function, for example the piece-wise linear one. Such a cost does not encourage risky position, since the cost is at least 0. This means that if $r > 0$, then there's no point having $p > 1$ instead of $p = 1$ since both will yield a zero cost position.



Utility. There are two ways we can model our utility, and both are concave shaped, to represent a risk-averse approach. The first utility is the linear utility of the form

$$U(r) = r + \min(0, \beta r), \tag{6}$$

with $\beta > 0$. The other utility is exponential:

$$U(r) = -\exp(-\mu r), \tag{7}$$

with $\mu > 0$.

Algorithm. We will be concerned with probabilistic confidence bounds on results produced using the following algorithm, using dataset S_n. We shall denote this algorithm as A_{S_n}.

$$q^\star = \underset{q \in \boldsymbol{R}^p}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} c(q^T x_i, r_i) + \lambda \|q\|_2^2. \tag{8}$$

Assumptions. We will assume that information vectors have been pre-processed and lie in a $X_{\max}^2$ radius ball. We also assume that the return rates observed are comprised within $[-\bar{r}, \bar{r}]$. This last assumption will be relaxed.

2.2 Theoretical Optimal Value

Before we go on to derive an algorithm for the in-sample optimal allocation, we first derive some theoretical results for an optimal allocation scalar $p^\star$, provided the distribution of the returns is priorly known.

Formally, we suppose that this distribution is characterized by its CDF F. We then wish to minimize the expected cost $c(p, r)$, where p is a variable and r a random return rate. Since,

$$E[c(p, r)] = \int_{-\infty}^{\infty} c(p, r)\, dF(r), \tag{9}$$

the optimal allocation $p^\star$ is therefore the solution to

$$\frac{d}{dp}\int_{-\infty}^{\infty} c(p,r)\,dF(r) = 0. \tag{10}$$

However, our theoretical assumptions are fundamentally different from this one, since we consider a joint distribution $\boldsymbol{D} = \boldsymbol{X} \times \boldsymbol{R}$, so that returns are dependant of a draw from $\boldsymbol{X}$, whereas it's assumed here that decision can only be made on the basis of the returns.

2.3 Definitions, Theorems and Tools

We now state a certain number of definitions and theorems that will be of great importance in the analysis on the results produced by the algorithm mentioned above. These definitions and theorems are taken verbatim from [2].

Definition. An algorithm A has *uniform stability* β with respect to the loss function ℓ if, for all $S_n \in \boldsymbol{D}^n$ and $i \in \{1,\dots,n\}$, the following holds:

$$\|\ell(A_{S_n},.) - \ell(A_{S_n^{\setminus i}},.)\|_\infty \leq \beta_n, \tag{11}$$

or, equivalently,

$$\sup_{d\in\boldsymbol{D}} |\ell(A_{S_n},d) - \ell(A_{S_n^{\setminus i}},d)| \leq \beta_n. \tag{12}$$

Here, $S_n^{\setminus i}$ means the set S_n with the ith data point removed.

Furthermore, A is *stable* when $\beta_n = O(1/n)$.

Definition. A loss function ℓ is *σ-admissible* if the associated cost function c is convex with respect to its first argument and the following condition holds for any p_1, p_2 and r:

$$|c(p_1,r) - c(p_2,r)| \leq \sigma|p_1 - p_2|. \tag{13}$$

It is easy to see that if $\operatorname{dom} c(\cdot,r)$ is bounded and $c(\cdot,r)$ does not reach infinity, then its loss function is σ-admissible.

Theorem 1. *Let F be a reproducing kernel Hilbert space with kernel κ such that $\forall x \in X$, $\kappa(x,x) \leq \kappa^2 < \infty$. If ℓ is σ-admissible with respect to F, then the learning algorithm defined by*

$$A_S = \operatorname*{arg\,min}_{g\in F} \frac{1}{n}\sum_{i=1}^{n} \ell(g,d_i) + \lambda\|g\|_k^2 \tag{14}$$

has uniform stability α_n with respect to ℓ with

$$\alpha_n \leq \frac{\sigma^2\kappa^2}{2\lambda n}. \tag{15}$$

Definition. The *true risk* with respect to algorithm A and set S_n is defined as

$$R_{\text{true}}(A, S_n) = E_d[\ell(A_{S_n}, d)], \tag{16}$$

which is, in plain words, the expected loss incured when applying the algorithm created from training set S_n in the wild, ie. out of sample.

Definition. The *empirical risk* with respect to algorithm A and set S_n is defined as

$$\hat{R}(A, S_n) = \frac{1}{n} \sum_{i=1}^{n} \ell(A_{S_n}, d_i), \tag{17}$$

which is, in plain words, the average cost incured by our model over all the training set.

Theorem 2. *Let A be an algorithm with uniform stability α_n with respect to a loss function ℓ such that $0 \leq \ell(A_{S_n}, d) \leq M$ for all $d = (x, r) \sim D$ and all sets S_n of size n. Then for any $n \geq 1$ and any $\delta \in (0, 1)$, the following bound holds with probability at least $1 - \delta$ over the random draw of the sample S_n:*

$$|R_{\text{true}}(A, S_n) - \hat{R}(A, S_n)| \leq 2\alpha_n + (4n\alpha_n + M)\sqrt{\frac{\log(2/\delta)}{2n}}. \tag{18}$$

2.4 Remarks and applications to our algorithm

In this section we shall derive bounds on the proposed algorithms concerned with the difference between the in-sample average cost using the decision vector determined by our algorithm with the (theoretical) expected cost over the whole distribution when using the same decision vector.

The analysis will be done in two parts, one for each of our cost functions: first with $c = -U$ (the canonical cost) and second with the cost function as the regret at each decision. Both are quite similar in their results.

2.4.1 Canonical cost.

We first consider the case where

$$c(p, r) = -U(pr + (1 - p)R_f). \tag{19}$$

Remark (σ-admissibility)**.** We first remark that both forms of U yield a convex function of p with r fixed, since U is concave.

Next, the expression $|c(p_1, r) - c(p_2, r)|$ reduces to

$$|U(p_1 r + (1 - p_1)R_f) - U(p_2 r + (1 - p_2)R_f|. \tag{20}$$

Now because $r \in [-\bar{r}, \bar{r}]$, U is Lipschitz continuous on its domain, and so (20) is bounded by

$$\alpha|p_1 r + (1-p_1)R_f - (p_2 r + (1-p_2)R_f)| = \alpha|p_1 - p_2||r - R_f| \tag{21}$$

where

$$\alpha = \sup_{r \in [-\bar{r}, \bar{r}]} |U'(r)|. \tag{22}$$

In the linear case, the derivative is piecewise constant, and is set to 1 on for returns below r_c, so that $\alpha = 1$. In the exponential case, $U'(r) = \exp \mu r$, and $\alpha = \exp \mu \bar{r}$.

The bound (21) must hold for any r. The expression $|r - R_f|$ will reach its largest value at $r = -\bar{r}$, since R_f is assumed to be non-negative, and therefore

$$|c(p_1, r) - c(p_2, r)| \leq \alpha(\bar{r} + R_f)|p_1 - p_2|, \tag{23}$$

which shows that ℓ is σ-admissible with the canonical cost, with $\sigma = \alpha(\bar{r} + R_f)$.

We also remark that the utility is actually invariant to scaling, ie. $\tilde{U} = kU$ where k is positive yields the same relative utility. The σ bound can therefore be as tight as desired, since it can be scaled by an arbitrary positive constant, and the algorithm will yield the same result.

Remark (Algorithmic Stability)**.** Because the loss function is σ-admissible, the conditions for Theorem 1 are respected: if we take the kernel on the space $\boldsymbol{R}^p$ as the regular inner product, than $\kappa(x, x) \leq X_{\max}^2$. Therefore, our algorithm has algorithmic stability with:

$$\alpha_n \leq \frac{k^2(\bar{r} + R_f)^2 X_{\max}^2}{2\lambda n} \tag{24}$$

with linear utility and

$$\alpha_n \leq \frac{k^2 \exp(2\mu\bar{r})(\bar{r} + R_f)^2 X_{\max}^2}{2\lambda n} \tag{25}$$

in the case of exponential utility.

As discussed above, $k > 0$ is a scaling term we can add to the utility function, with which we can actually have algorithmic stability as tight as needed.

How can we leverage k?

Remark (Generalization Bound)**.** Using Theorem 2, we can derive an out-of-sample bound on the result produced by our algorithm for each utility forms that holds with probability $1 - \delta$, for $0 < \delta < 1$. The M value stated by Theorem 2 corresponds to the highest loss we can incur on the domain, which happens when $p = 1$ and $r = -\bar{r}$, ie.

$$M = c(1, -\bar{r}) = -U(\bar{r}). \tag{26}$$

In the linear utility case,

$$|R_{\text{true}}(A, S_n) - \hat{R}(A, S_n)| \leq 2\alpha_n + (4n\alpha_n + M)\sqrt{\frac{\log(2/\delta)}{2n}} \tag{27}$$

$$\leq \frac{k^2(\bar{r} + R_f)^2 X_{\max}^2}{\lambda n} + \left(\frac{2k^2(\bar{r} + R_f)^2 X_{\max}^2}{\lambda} + M\right)\sqrt{\frac{\log(2/\delta)}{2n}}, \tag{28}$$

and in the exponential utility case,

$$|R_{\text{true}}(A, S_n) - \hat{R}(A, S_n)| \leq 2\alpha_n + (4n\alpha_n + M)\sqrt{\frac{\log(2/\delta)}{2n}} \tag{29}$$

$$\leq \frac{k^2 \exp(2\mu\bar{r})(\bar{r} + R_f)^2 X_{\max}^2}{\lambda n} + \left(\frac{2k^2 \exp(2\mu\bar{r})(\bar{r} + R_f)^2 X_{\max}^2}{\lambda} + M\right)\sqrt{\frac{\log(2/\delta)}{2n}}. \tag{30}$$

Remark (Absolute Bound)**.** Finally, we derive a measure of performance when we compare $\hat{q} = A_{S_n}$ with the true optimal $q^\star$ over the distribution, and how each perfom when applied on the true risk of the distribution...

2.4.2 Regret cost.

We now turn our attention toward the regret cost, ie.

$$c(p, r) = \lfloor U(\max(r, R_f)) - U(pr + (1 - p)R_f) \rfloor. \tag{31}$$

This type cost is different from the canonical cost in that to does not encourage 'risky' position for which $0 < p < 1$, since there's no gain in over-investing our portfolio in a winning asset. Conversely, there's no gain in shorting a losing asset. Such a cost could therefore be more suitable in a conservative environment, even if the results produced by the algorithm A might very well yield a position $0 < p < 1$.

Remark (σ-admissibility)**.** The σ-admissibility of the regret cost follows closely from the σ-admissibility of the canonical cost derived above. Indeed, we remark that, provided that $c(p_1, r), c(p_2, r) > 0$, then the expression $|c(p_1, r) - c(p_2, r)|$ reduces to

$$|U(p_1 r + (1 - p_1)R_f) - U(p_2 r + (1 - p_2)R_f)|, \tag{32}$$

wich has exactly the same form as (20), and therefore yields the same σ bound. Now for the case where, without loss of generality $c(p_2, r) = 0$. Let $\tilde{c}$ be defined as c, but without $\lfloor \cdot \rfloor$, so that $\lfloor \tilde{c}(p, r) \rfloor = c(p, r)$. In particular $\tilde{c}(p_2, r) < 0$ and so

$$|c(p_1, r) - c(p_2, r)| = |\lfloor \tilde{c}(p_1, r) \rfloor - \lfloor \tilde{c}(p_2, r) \rfloor| \tag{33}$$

$$\leq |\tilde{c}(p_1, r) - \tilde{c}(p_2, r)| \tag{34}$$

$$= |U(p_1 r + (1 - p_1)R_f) - U(p_2 r + (1 - p_2)R_f)|, \tag{35}$$

which has again the the same form as (20).

This means that the loss defined with the regret cost has the same σ-admissibility as the loss defined with the canonical cost, namely

$$\sigma = k(\bar{r} + R_f) \tag{36}$$

in the case of linear utility and

$$\sigma = k(\bar{r} + R_f)\exp(\mu\bar{r}) \tag{37}$$

in the case of exponential utility.

Remark (Algorithmic Stability)**.** Because this regret cost has the same σ-admissibility as the canonical cost, then the algorithmic stability also remains the same for both algorithm, ie.

$$\alpha_n \leq \frac{k^2(\bar{r} + R_f)^2 X_{\max}^2}{2\lambda n} \tag{38}$$

with linear utility and

$$\alpha_n \leq \frac{k^2 \exp(2\mu\bar{r})(\bar{r} + R_f)^2 X_{\max}^2}{2\lambda n} \tag{39}$$

in the case of exponential utility.

Remark (Generalization Bound)**.** Likewise, the only difference here with the canonical cost is the value of the M variable of Theorem 2, defined as the highest cost on the domain. In the case here, this value is again reached for an investment $p = 1$ on a losing asset at $r = -\bar{r}$, so that,

$$M = \lfloor U(R_f) - U(\bar{r}) \rfloor. \tag{40}$$

We are left with the same expression for the generalization bounds, in the linear utility case,

$$|R_{\text{true}}(A, S_n) - \hat{R}(A, S_n)| \leq 2\alpha_n + (4n\alpha_n + M)\sqrt{\frac{\log(2/\delta)}{2n}} \tag{41}$$

$$\leq \frac{k^2(\bar{r} + R_f)^2 X_{\max}^2}{\lambda n} + \left(\frac{2k^2(\bar{r} + R_f)^2 X_{\max}^2}{\lambda} + M\right)\sqrt{\frac{\log(2/\delta)}{2n}}, \tag{42}$$

and in the exponential utility case,

$$|R_{\text{true}}(A, S_n) - \hat{R}(A, S_n)| \leq 2\alpha_n + (4n\alpha_n + M)\sqrt{\frac{\log(2/\delta)}{2n}} \tag{43}$$

$$\leq \frac{k^2 \exp(2\mu\bar{r})(\bar{r} + R_f)^2 X_{\max}^2}{\lambda n} + \left(\frac{2k^2 \exp(2\mu\bar{r})(\bar{r} + R_f)^2 X_{\max}^2}{\lambda} + M\right)\sqrt{\frac{\log(2/\delta)}{2n}}. \tag{44}$$

2.5 True Optimal Bound

One bound that is still of interest is the absolute difference between the in-sample cost using in-sample decision $\hat{R}(A, S_n)$ and the expected cost over the unknown distribution D (ie. the true risk) using the unknown optimal decision $q^\star$ defined by

$$q^\star = \underset{\{q : X \subset \boldsymbol{R}^p \to \boldsymbol{R}\}}{\arg\min} \; R_{\text{true}}(q). \tag{45}$$

The true optimal bound is therefore expressed by

$$|R_{\text{true}}(q^\star) - \hat{R}(A, S_n)|. \tag{46}$$

We can then apply the triangle inequality to bound this expression by the generalization bound plus $|R_{\text{true}}(q^\star) - R_{\text{true}}(\hat{q})|$.

However, whereas [1] could bound this expression using a theorem from quantile regression theory, I'm here at loss (no pun intended...)

Things to consider. Here are some points that should be considered.

- We have assumed so far we were dealing with datasets taken iid. But in a financial setting, this is not necessarily the case, as influence of features might be changing with time. (HMM?)
- In the newsvendor article, the generalization bounds actually hold with probability $1 - \delta - n\gamma$, if we assume $\bar{D}$ (the highest considered demand) holds with probability $1 - \gamma$. It is only mentioned in the non-regularized case. A reference would be needed, as the $n\gamma$ terms grows up too quickly to be of any use (unless we're being very restrictive on $X_{\max}$). [Reference needed !]
- We are also considering discrete quantities, updated daily. Our model should encompass quantities that might change during the day.
- A section on how data should be normalized must also be added. This is essential to go from theory to pratical applications.

3 Multi-asset Portfolio

We now consider a multi-asset portfolio, made of m risky assets and a risk-free asset. Each of theses risky assets have information vector x_{st}, where $s \in \{1, \dots, m\}$ and $t \in \{1, \dots, n\}$. Therefore, we suppose having access to a sample dataset $S_n = \{X_1, \dots, X_n\}$ where $X_i \in \boldsymbol{R}^{m \times p}$.

There are two ways to approach the multi-asset scenario.

3.1 First approach

On one hand, we can define a single decision vector q such that the return on a single holdinq period would be defined by

$$r^T X q + (1 - 1^T X q) R_f, \tag{47}$$

where $r \in \boldsymbol{R}^m$ is the return vector of the m stocks and 1^T is a contant vector of ones in $\boldsymbol{R}^m$ whose role is simply to sum over the components of Xq, ie. sum over the risky allocations of the portfolio. However, this method has the disadvantage that the decision vector is applied for each stock in the same way, and so all their information vectors are ultimately 'averaged' over. Therefore, if some asset is more sensitive to an information component than other assets, this knowledge might be lost if using this method.

Nevertheless, we can again define two different cost functions, which our optimal decision vector would minimize when applied to our sample dataset S_n. Again, the cost can be simply defined as

$$c(p, r) = -U(r^T p + (1 - 1^T p) R_f). \tag{48}$$

As discussed previously, this definition might have the disadvantage of providing an unconstrained optimization problem when determining the optimal decision vector.

We are not limited to a single cost definition, and in fact, the multi-asset scenario provides many legitimate cost definitions. Here the most 'aggressive' one:

$$c(p, r) = \begin{cases} \lfloor U(\max r) - U(r^T p + (1 - 1^T p) R_f) \rfloor & \text{if } \max r > R_f \\ \lfloor U(R_f) - U(r^T p + (1 - 1^T p) R_f) \rfloor & \text{if } \max r \leq R_f. \end{cases} \tag{49}$$

This means that we expect (or conversely, that we suffer a lesser loss) when the total return of our portfolio is near the best return achieved by our considered stocks. Instead of $\max r$, we also could have chosen the average returns $\bar{r}$ during the period as the target that we wish to achieve (provided that the average is superior to the risk-free return). We could in fact chose any function mapping from a vector of returns r to a scalar as our global return objective.

In any case, the algorithm determining the decision vector q would once be defined as previously, ie. as (8).

3.2 Second approach

Under this second approach, instead of averaging over all information vector at the risk of losing asset-specific information, we now use a decision matrix Q of size $p \times m$, whose m columns are actually trained independently one to each other so that each asset reacts differently from the rest.

The allocation of each vector would then be $\text{diag}(XQ)$, ie. the diagonal elements of the $m \times m$ matrix XQ. The one-period return is therefore

$$r^T \text{diag}(XQ) + (1 - 1^T \text{diag}(XQ))R_f. \tag{50}$$

This method still has a disadvantage though, in that it won't consider the eventual influence of information x_j of asset j to asset j, ie. each stock is trained independantly one to each other.

3.3 Third Approach

Finally, we introduce a third approach with the objective of using all available data of the market, available through $X \in \boldsymbol{R}^{m \times sp}$. However, we must be clear about what information X carries with it; its information is only bounded to different assets, and therefore does not include 'global' information with it.

Following a simple idea introduced by [4], we can add an unknown linear transformation W to X so that q can benefit from the whole information matrix, and not only from the average of its columns. The total return using a transformation W and decision vector q would therefore be expressed as

$$r^T XWq + (1 - 1^T XWq)R_f. \tag{51}$$

3.4 Fourth Approach

We now introduce a more general framework that actually encompasses all other methods we've seen so far. In addition, we also add regularization terms so that the following rules are respected :

- Given an asset i, we insist that the allocation scalar for i relies more strongly on feature vector x_i than on feature vectors from other assets in the market.
- The market should be segmented in different industries, and our algorithm should partition the m stocks in k groups of similar nature. This would also impact how cross-influence of feature affect very weakly assets of other groups.
- Each asset i has its own decision vectors q_{ij}, where q_{ij} is the influence of asset j on asset i.

With this in mind we can now define a general allocation rule given feature vectors $\{x_i\}$ for each asset in the market. As before, we consider a general inner product $x_i^T q_j$ as a mapping from the feature space of i to the allocation scalar space. Therefore, the total influence of the m assets on i can be expressed as

$$x_i^T q_{ii} + \sum_{j \neq i} x_j^T q_{ij}. \tag{52}$$

We can reduce this expression by introducing a decision matrix $Q_i \in \boldsymbol{R}^{p \times m}$ defined by

$$Q_i = \begin{pmatrix} [Q_i]_1 & \cdots & [Q_i]_m \end{pmatrix}. \tag{53}$$

By using the same feature matrix X, the above expression is therefore equivalent to

$$\operatorname{tr}(XQ_i), \tag{54}$$

so that the total return will be given by

$$\sum_{i=1}^{m} r_i \operatorname{tr}(XQ_i) + \left(1 - \sum_{i=1}^{m} \operatorname{tr}(XQ_i)\right) R_f. \tag{55}$$

Now, as before the Q_i matrices are unknown at first, but can be learned using an optimization problem, where the objective is the cost $c(p, r)$. Let us, just for a moment, neglect the regularization term. Than, the optimal matrices are $Q_i^\star$ are the solution to the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & c(p, r) && (56) \\ \text{s. t.} \quad & p_i = \operatorname{tr}(XQ_i). && (57) \end{aligned}$$

The cost function c is defined as above, ie. using a cost for vectorized returns and allocation vector.

Now, there are some fundamental questions concerning this optimization problem which we'll report for later. Indeed, before we start analyzing the theoretical guanrantees we can get out of this problem, we would first like to add regularization terms to the objective, with the hope of being compliants with the rules we've proposed above.

The first rule we proposed is certainly the most easy to implement. What it says is basically that in the allocation scalar for asset i, the contribution from feature i should be larger than contribution coming from feature of other assets.

Mathematically, this would translate in this 'soft' constraint:

$$x_i^T [Q_i]_i \gg \sum_{j \neq i} x_j^T [Q_i]_j. \tag{58}$$

In an optimization context, we can add the following expression to the objective:

$$+ \operatorname{tr}(XQ_i)^2 - \lambda(x_i^T[Q_i]_i)^2, \tag{59}$$

with $\lambda \geq 0$. In plain words, this means that we expect the whole allocation scalar close to the contribution from $x_i^T[Q_i]_i$, with a regularization parameter λ to avoid all other terms $x_j^T[Q_i]_j$ falling to zero.

Now, let's tackle the second constraint, where we expect the cross-contribution of a feature vector to only affect a close number of assets. For example, we'd expect technology assets to affect each other, but with little impact on energy assets. This evidently reminds of $L1$ regularization, where the activation is non-smooth, and either you're part of a group, or you're not.

We can first start by defining the number of groups, or segments, we exepct the market to have. We shall note this number of groups by k.

Let us then state the problem formally. Let $\Xi \in \boldsymbol{R}^{m \times m}$ be the allocation matrix, such that

$$[\Xi]_{ij} = x_j^T[Q_i]_j. \tag{60}$$

Then, the allocation vector p for each of the m assets corresponds to the sum of each of the rows of Ξ. Therefore, Ξ is invariant to reordering of individual rows. In the process of optimization, we want to have to turn Ξ in a k-block matrix, with each block representing a certain sector. However, this *block* requirement should not be too hard, as in certain event, we could have assets with overlapping sectors. Anyway, $L1$ sounds like a good idea, since this kind of regularization often allows for features to be either 'in' or 'out'. In regularization, there's always two opposing forces, and its role is to find balance between those two forces.

So what are the two forces at play here? On one side, the optimization problem (56)

Food for thought

Add something perhaps on how he can cramp into the model some paarameters so that only a handful assets are being chosen (perhaps L1 reg. or something like that). Also maybe we could add another complexity where we would like to keep stocks we already have, perhaps using a utility telling how much we want to discourage the new daily assets. That sort of things.

Something else that was so far overlooked is the fact that the sum of all allocation components need not sum up to 1. Again, a regularization with a plateau shape form could perhaps translate what we need.

References

[1] Cynthia Rudin and Gah-Yi Vahn. *The Big Data Newsvendor: Pratical Insights from Machine Learning*, Operations Research, 2015.

[2] Olivier Bousquet and André Elisseeff. *Stability and Generalization*, Journal of Machine Learning Research, 2002.

[3] Rockafellar, R. T. *Convex Analysis*, Princeton University Press, 1970.

[4] Ming Fai Wong et al. *Stock Market Prediction from WSJ: Text Mining via Sparse Matrix Factorization.* ICDM 2014.