Portfolio Optimization

An Application of Convex Optimization

A project as part of the course:
"Advanced Topics in Convex Optimization"

Supplementary Document
Algorithmic Solutions - Programming

Paraskakis Nikolaos, Undergraduate Student

School of Electrical & Computer Engineering
Technical University of Crete

November 4, 2022

Contents

Projected (Sub)Gradient Method

Consider the optimization model

$$\min \{f(\mathbf{w}) : \mathbf{w} \in C\} \tag{1}$$

Suppose that the below assumptions hold:

- $f : \mathbb{E} \to (-\infty, \infty]$ is proper closed convex
- $C \subseteq \mathbb{E}$ is nonempty, closed, convex and compact
- $C \subseteq \text{int}(\text{dom}(f))$
- The optimal set of (1) is nonempty and denoted by W^*. The optimal value of the problem is denoted by f_{opt}.

The above imply the validity of the additional assumption that there exists a constant $L_f > 0$ for which $\|\mathbf{g}\|_2 \leq L_f$ for all $\mathbf{g} \in \partial f(\mathbf{w}), \mathbf{w} \in C$. That L_f is a Lipschitz constant of f over C.

The general form of the algorithm is the following:

$\mathbf{w}_0 \in \mathbb{R}^n, k = 0$
while (1) **do**
  Choose step-size $t_k > 0$
  Compute $f'(\mathbf{w}_k) \in \partial f(\mathbf{w}_k)$
  $\mathbf{w}_{k+1} = P_C(\mathbf{w}_k - t_k f'(\mathbf{w}_k))$
  $k = k + 1$
  if $(convergence)$ **then**
    quit
  end
end

Algorithm 1: Projected subgradient method (General form)

Now, we choose one from below as the step-size rule:

Polyak step-size:

$$t_k = \begin{cases} \frac{f(\mathbf{w}_k) - f_{opt}}{\|f'(\mathbf{w}_k)\|^2} & , f'(\mathbf{w}_k) \neq \mathbf{0} \\ 1 & , f'(\mathbf{w}_k) = \mathbf{0} \end{cases}$$

Dynamic step-size:

$$t_k = \begin{cases} \frac{1}{\|f'(\mathbf{x}_k)\|\sqrt{k+1}} & , f'(\mathbf{w}_k) \neq \mathbf{0} \\ \frac{1}{L_f} & , f'(\mathbf{w}_k) = \mathbf{0} \end{cases}$$

As far as the subgradient is concerned, it is important to note that all the problems that we refer in this presentation have differentiable objective functions, so the term subgradient is abusive and in essence we are looking for objective function's gradient. So, we are using projected gradient method.

Minimum volatility

- $f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{\Sigma}\mathbf{w}$
- $f'(\mathbf{w}) = \nabla f(\mathbf{w}) = \mathbf{\Sigma}\mathbf{w}$
- $L_f = \rho(\mathbf{\Sigma})$
- $C = \{\mathbf{w} \in \mathbb{R}^n : \mathbf{0} \leq \mathbf{w} \leq \mathbf{1}, \mathbf{1}^T\mathbf{w} = 1\}$

Minimum volatility for a given target return

- $f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{\Sigma}\mathbf{w}$
- $f'(\mathbf{w}) = \nabla f(\mathbf{w}) = \mathbf{\Sigma}\mathbf{w}$
- $L_f = \rho(\mathbf{\Sigma})$
- $C = \{\mathbf{w} \in \mathbb{R}^n : \mathbf{0} \leq \mathbf{w} \leq \mathbf{1}, \mathbf{1}^T\mathbf{w} = 1, \boldsymbol{\mu}^T\mathbf{w} = r\}$

Maximum quadratic utility given some risk aversion

- $f(\mathbf{w}) = -\boldsymbol{\mu}^T\mathbf{w} + \gamma\frac{1}{2}\mathbf{w}^T\mathbf{\Sigma}\mathbf{w}$
- $f'(\mathbf{w}) = \nabla f(\mathbf{w}) = -\boldsymbol{\mu} + \gamma\mathbf{\Sigma}\mathbf{w}$
- $L_f = \rho(\mathbf{\Sigma})$
- $C = \{\mathbf{w} \in \mathbb{R}^n : \mathbf{0} \leq \mathbf{w} \leq \mathbf{1}, \mathbf{1}^T\mathbf{w} = 1\}$

Note that $\rho(\mathbf{\Sigma})$ is the spectral radius of a square matrix $\mathbf{\Sigma}$ and is equal to the maximum of the absolute values of its eigenvalues.

As far as the step-size is concerned, we will experiment with the dynamic step-size rule, because it does not need the knowledge of the optimal value and is more applicable in real life.

For convergence we check whether $f(\mathbf{w}_k) \leq c \cdot f_{opt}$. Parameter c takes values near 1, such 1.001 or smaller. This condition can be used, given that we have prior solved problem via CVX, so we have obtained f_{opt}. In another case, we can use for convergence checking the condition $\| \frac{\mathbf{w}_k - \mathbf{w}_{k+1}}{\mathbf{w}_k} \|_2 \approx 0$.

The projection onto set C, namely $P_C(\mathbf{w})$, is being calculated using the FDPG method and it will be explained at a next section.

Interior Point Method

Let the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & f_0(\mathbf{w}) \\ \text{s.t.} \quad & f_i(\mathbf{w}) \leq 0, i = 1, 2, \ldots, m \\ & \mathbf{A}\mathbf{w} = \mathbf{b} \end{aligned} \tag{2}$$

where we assume the following:

- $f_i : \mathbf{dom} f_i \subseteq \mathbb{R}^n \to \mathbb{R}, i = 1, 2, \ldots, m$ are convex and twice differentiable functions
- $\mathbf{A} \in \mathbb{R}^{p \times n}$, $rank(\mathbf{A}) = p$ and $\mathbf{b} \in \mathbb{R}^p$

Problem (2) is defined over set $\mathbb{D} := \cap_{i=1}^{m} \mathbf{dom} f_i$ and the feasible set is defined as follows:

$$\mathbb{W} := \{\mathbf{w} \in \mathbb{D} \mid f_i(\mathbf{w}) \leq 0, i = 1, 2, \ldots, m, \ \mathbf{A}\mathbf{w} = \mathbf{b}\}$$

Interior point methods solve problem (2) by solving a sequence of problems with affine equality constraints. These problems are approximations of problem (2) and their solutions are strictly feasible for problem (2). Problem (2) can be expressed also as follows:

$$\begin{aligned} \min_{\mathbf{w}} \quad & f_0(\mathbf{w}) + \sum_{i=1}^{m} I_-(f_i(\mathbf{w})) \\ \text{s.t.} \quad & \mathbf{A}\mathbf{w} = \mathbf{b} \end{aligned} \tag{3}$$

where

$$I_-(u) := \begin{cases} 0 & , \ u \leq 0 \\ \infty & , \ u > 0 \end{cases}$$

Cost function of problem (3) is in general not differentiable. Consequantly, we cannot use gradient descent method or Newton method to solve it.

An approximation of function $I_{-}(u)$ is $\hat{I}_{-}(u)$, which is defined as follows:

$$\hat{I}_{-}(u) := -\frac{1}{t}\log(-u)$$

with $\mathbf{dom}\hat{I}_{-} = -\mathbb{R}_{++}$ and $t > 0$.

Function $\hat{I}_{-}$ is closed and convex and it is extremely useful.

The value of parameter t defines the quality of the approximation.

We define the **logarithmic barrier function** for problem (2) as follows:

$$\phi(\mathbf{w}) := -\sum_{i=1}^{m}\log(-f_i(\mathbf{w}))$$

with $\mathbf{dom}\phi = \{\mathbf{x} \in \cap_{i=1}^{m}\mathbf{dom}f_i \mid f_i(\mathbf{w}) < 0, i = 1, 2, \ldots, m\}$.

Function ϕ is convex and twice differentiable.

Next, we define the following problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & f_0(\mathbf{w}) + \frac{1}{t}\phi(\mathbf{w}) \\ \text{s.t.} \quad & \mathbf{Aw} = \mathbf{b} \end{aligned} \tag{4}$$

with $\mathbf{w} \in \mathbf{dom} f_0 \cap \mathbf{dom}\phi \subset \mathbb{W}$.

Problem (4) is convex and Newton algorithm for affine equality constrained problems can be used. We can, also, use the projected gradient algorithm, but now we will focus on the Newton algorithm.

Problem (4) or the following equivalent problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & f_t(\mathbf{w}) \equiv t f_0(\mathbf{w}) + \phi(\mathbf{w}) \\ \text{s.t.} \quad & \mathbf{Aw} = \mathbf{b} \end{aligned} \tag{5}$$

are approximations of problem (2), with the approximation getting better as parameter t is increasing.

The addition of term $\frac{1}{t}\phi(\mathbf{w})$ at the cost function $f_0(\mathbf{w})$, intuitively, raises a barrier at the boundary of the feasible set $\mathbb{W}$. In that way, the solution of problem (5) is getting trapped in the interior of set $\mathbb{W}$, but allowing to get very close to its boundary for large values of paramter t.

The first and the second derivative of function $\phi(\mathbf{w})$, where ϕ is the logarithmic barrier function, are given by:

$$\nabla\phi(\mathbf{w}) = \sum_{i=1}^{m} \frac{1}{-f_i(\mathbf{w})} \nabla f_i(\mathbf{w})$$

$$\nabla^2\phi(\mathbf{w}) = \sum_{i=1}^{m} \frac{1}{f_i^2(\mathbf{w})} \nabla f_i(\mathbf{w}) \nabla f_i(\mathbf{w})^T + \sum_{i=1}^{m} \frac{1}{-f_i(\mathbf{w})} \nabla^2 f_i(\mathbf{w})$$

By the form of $\nabla^2\phi(\mathbf{w})$ and the fact that $f_i(\mathbf{w}) < 0, i = 1, 2, \ldots, m$ for $\mathbf{w} \in \mathbf{dom}\phi$, we can see easilly that $\phi(\mathbf{w})$ is a convex function (the Hessian is non-negative definite).

Let $\mathbf{w}_*$ the solution of problem (2) and the optimal value $p_* := f_0(\mathbf{w}_*)$.

Let $\mathbf{w}_*(t)$ the solution of problem (5) for given value of parameter t.

It holds that $\mathbf{w}_*(t) - p_* \leq \frac{m}{t}$. It means that:

- For given value of parameter t, point $\mathbf{w}_*(t)$ is at most $\frac{m}{t}$-suboptimal for problem (2).
- For $t \to \infty$, $\mathbf{w}_*(t)$ converges to the optimal point of problem (2).

An approximation for solving problem (2), with tolerance ϵ, is the solution of the following problem:

$$\begin{array}{ll} \min\limits_{\mathbf{w}} & \frac{m}{\epsilon} f_0(\mathbf{w}) + \phi(\mathbf{w}) \\ \text{s.t.} & \mathbf{A}\mathbf{w} = \mathbf{b} \end{array} \tag{6}$$

Generally, problem (6) is difficult to solve for small value of ϵ. Usually, we use the interior point barrier method, which we explain below.

An efficient approach for the solution of problem (6) is the solution of a sequence of problems of form (5) with increasing t. The solution of such a problem for a specific t consists the initial point for the next problem (with greater t).

This algorithm is as shown below and its initial point has to be feasible.

$\mathbf{w} = \mathbf{w}_0 \in \mathbf{dom} f_0 \cap \mathbf{dom}\phi$ with $\mathbf{A}\mathbf{w}_0 = \mathbf{b}$
$t > 0, \mu > 1,$ tolerance $\epsilon > 0, k = 0$
while (1) **do**
  Compute $\mathbf{w}_* (t)$ by minimizing $tf_0 + \phi$ s.t. $\mathbf{A}\mathbf{w} = \mathbf{b}$, starting at $\mathbf{w}$
  $\mathbf{w} := \mathbf{w}_* (t)$
  $k = k + 1$
  if $\left(\frac{m}{t} < \epsilon\right)$ **then**
    quit
  end
  $t = \mu t$
end

Algorithm 2: The Interior Point Barrier Method

Choice of μ

The choice of the parameter μ involves a trade-off in the number of inner and outer iterations required. If μ is small (i.e. near 1) then at each outer iteration t increases by a small factor. As a result the initial point for the Newton process, i.e. the previous iterate $\mathbf{w}$, is a very good starting point, and the number of Newton steps needed to compute the next iterate is small. Thus, for small μ we expect a small number of Newton steps per outer iteration, but of course a large number of outer iterations since each outer iteration reduces the gap by only a small amount. In this case, the iterates (and indeed, the iterates of the inner iterations as well) closely follow the central path. This explains the alternate name path-following method.

On the other hand, if μ is large, we have the opposite situation. After each outer iteration t increases a large amount, so the current iterate is probably not a very good approximation of the next iterate. Thus, we expect many more inner iterations. This aggressive updating of t results in fewer outer iterations, since the duality gap is reduced by the large factor μ at each outer iteration, but more inner iterations. With μ large, the iterates are widely separated on the central path; the inner iterates veer way off the central path.

This trade-off in the choice of μ is confirmed both in practice and in theory. In practice, small values of μ (i.e. near 1) result in many outer iterations, with just a few Newton steps for each outer iteration. For μ in a fairly large range, from around 3 to 100 or so, the two effects nearly cancel, so the total number of Newton steps remains approximately constant. This means that the choice of is not particularly critical; values from around 10 to 20 or so seem to work well. When the parameter μ is chosen to give the best worst-case bound on the total number of Newton steps required, values of μ near 1 are used.

Choice of t_0

Another important issue is the choice of initial value of t. Here the trade-off is simple. If t_0 is chosen too large, the first outer iteration will require too many iterations. If t_0 is chosen too small, the algorithm will require extra outer iterations, and possibly too many inner iterations in the first centering step. Since $\frac{m}{t_0}$ is the duality gap that will result from the first centering step, one reasonable choice is to choose t_0 so that $\frac{m}{t_0}$ is approximately of the same order as $f_0(\mathbf{w}(0)) - p_*$, or μ times this amount. For example, if a dual feasible point λ, v is known, with duality gap $\eta = f_0(\mathbf{w}(0)) - g(\lambda, v)$, then we can take $t_0 = \frac{m}{\eta}$. Thus, in the first outer iteration we simply compute a pair with the same duality gap as the initial primal and dual feasible points.

Now, we will see how to approach problems of form (5). It is a convex minimization problem with affine equality constraints and twice differentiable cost function. To solve that, we can use projected gradient method, as well as Newton method starting from feasible point. Here, we will focus on Newton method.

The choice between the two algorithms is not so critical but there are some thing we must observe. Practical difference is that Newton method assumes you have much more information available, makes much better updates, and thus converges in less iterations.

But number of iterations needed is not all you want to know. The update of Newton method scales poorly with problem size. If $\mathbf{w} \in \mathbb{R}^d$, then to compute $[\nabla^2 f_t(\mathbf{w})]^{-1}$ you need $\mathcal{O}\left(d^3\right)$ operations. On the other hand, cost of update for gradient descent is linear in d.

In many large-scale applications, very often arising in machine learning for example, d is so large (can be a billion) that you are way beyond being able to make even a single Newton update. But in our case, we do portfolio optimization, where the number of assets is in general small, so the use of Newton method is a good choice. The analytic steps of this algorithm are shown below.

Newton Method

$\mathbf{w} = \mathbf{w}_0 \in \mathbf{dom} f_0 \cap \mathbf{dom}\phi$ with $\mathbf{A}\mathbf{w}_0 = \mathbf{b}$
Backtracking parameters $\alpha \in (0, 0.5)$ and $\beta \in (0, 1)$
$t > 0$, tolerance $\epsilon > 0$, $k = 0$
while (1) **do**

$$\begin{bmatrix} \Delta \mathbf{w}_k \\ \mathbf{s}_k \end{bmatrix} = \begin{bmatrix} \nabla^2 f_t(\mathbf{w}_k) & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} -\nabla f_t(\mathbf{w}_k) \\ \mathbf{0} \end{bmatrix}$$

$\lambda^2(\mathbf{w}_k) = \Delta \mathbf{w}_k^T \nabla^2 f_t(\mathbf{w}_k) \Delta \mathbf{w}_k$
$k = k + 1$
if $\left(\frac{\lambda^2(\mathbf{w}_k)}{2} < \epsilon\right)$ **then**
quit
end
$t = 1$
while $(\mathbf{w}_k + t\Delta \mathbf{w}_k \notin \{\mathbf{w} \in \mathbf{dom} f_0 \cap \mathbf{dom}\phi \mid \mathbf{A}\mathbf{w} = \mathbf{b}\})$ **do**
$t = \beta t$
end
while $\left(f_t(\mathbf{w}_k + t\Delta \mathbf{w}_k) > f_t(\mathbf{w}_k) + \alpha t \nabla f_t(\mathbf{w}_k)^T \Delta \mathbf{w}_k\right)$ **do**
$t = \beta t$
end
$\mathbf{w}_{k+1} = \mathbf{w}_k + t\Delta \mathbf{w}_k$
end

Algorithm 3: The Newton Method

Finding a feasible initial point

Now, we will discuss how we can compute a feasible initial point needed for the algorithms described above. Such a point can be computed easily via CVX by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & 1 \\ \text{s.t.} \quad & f_i(\mathbf{w}) \leq 0, i = 1, 2, \ldots, m \\ & \mathbf{A}\mathbf{w} = \mathbf{b} \end{aligned} \tag{7}$$

Next, we will see how we can compute a feasible initial point using the interior point barrier method. An approach of finding such a point for problem (2) is by solving the following feasibility problem:

$$\begin{aligned} \min_{\mathbf{w} \in \mathbb{R}^n, s \in \mathbb{R}} \quad & s \\ \text{s.t.} \quad & f_i(\mathbf{w}) \leq s, i = 1, 2, \ldots, m \\ & \mathbf{A}\mathbf{w} = \mathbf{b} \end{aligned} \tag{8}$$

If there exists $\mathbf{w}$ such that $\mathbf{Aw} = \mathbf{b}$, let $\bar{\mathbf{w}}$, then problem (8) is always strictly feasible, because we can let $(\bar{\mathbf{w}}, \bar{s})$ as initial point with:

$$\bar{s} \geq \max\{f_1(\mathbf{w}), f_2(\mathbf{w}), \ldots, f_m(\mathbf{w})\}$$

If $p < n$, then the system $\mathbf{Aw} = \mathbf{b}$ has infinite solutions (remember the assumption that thr rows of matrix $\mathbf{A}$ are linearly independent. A solution is given by $\bar{\mathbf{w}} = \mathbf{A}^{\#}\mathbf{b}$, where $\mathbf{A}^{\#}$ is the pseudo-inverse of matrix $\mathbf{A}$.

Let $\bar{\mathbf{w}}_*$ the optimal point and $\bar{p}_*$ the solution of problem (8). Then, it holds that:

- If $\bar{p}_* > 0$, then problem (2) is infeasible.
- If $\bar{p}_* < 0$, then problem (2) is feasible, and one feasible point for problem (2) is $\bar{\mathbf{w}}_*$.
- If $\bar{p}_* = 0$, then problem (2) is not strictly feasible.
  In practice, we may have $|\bar{p}_*| < \epsilon$ for a very small $\epsilon > 0$. In that case, inequalities $f_i(\mathbf{w}) < -\epsilon, i = 1, 2, \ldots, m$, are not feasible, while inequalities $f_i(\mathbf{w}) < \epsilon, i = 1, 2, \ldots, m$, are feasible.

It is important to note that it is not necessary to solve problem (8).

It is enough to find a $\mathbf{w}$ that gives $s < 0$.

Maximum return for a given target volatility

The following problem is equivalent to the corresponding original, which can be solved with the interior point barrier method explained before:

$$
\begin{aligned}
\min_{\mathbf{w}} \quad & f_0(\mathbf{w}) \equiv -\boldsymbol{\mu}^T \mathbf{w} \\
\text{s.t.} \quad & \mathbf{A}\mathbf{w} = \mathbf{b} \text{ , where } \mathbf{A} \equiv \mathbf{1}^T \text{ and } \mathbf{b} \equiv 1 \\
& f_{1,i}(\mathbf{w}) \equiv -w_i \leq 0 \text{ , } i = 1, 2, \ldots n \\
& f_{2,i}(\mathbf{w}) \equiv w_i - 1 \leq 0 \text{ , } i = 1, 2, \ldots n \\
& f_3(\mathbf{w}) \equiv \mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w} - v \leq 0
\end{aligned}
\tag{9}
$$

The equivalence of the original problem and problem (9) lies in the fact that inequality $f_3(\mathbf{w})$ is going to be satisfied as an equality. This is because as we reducing the variance v (volatility $\sqrt{v}$), the return decreases. But we want to maximize return, so the maximum return comes with exact variance value v.

Phase I

For the cost function $f_0(\mathbf{w}, s)$ we have the following:

- $f_0(\mathbf{w}, s) = s$
- $\nabla f_0(\mathbf{w}, s) = \mathbf{e}_{n+1}$
- $\nabla^2 f_0(\mathbf{w}, s) = \mathbf{0}$

For the constraint function $f_{1,i}(\mathbf{w}, s)$ we have the following:

- $f_{1,i}(\mathbf{w}, s) = -\mathbf{w}_i - s, i = 1, 2, \ldots, n$
- $\nabla f_{1,i}(\mathbf{w}, s) = -\mathbf{e}_i - \mathbf{e}_{n+1}, i = 1, 2, \ldots, n$
- $\nabla^2 f_{1,i}(\mathbf{w}, s) = \mathbf{0}, i = 1, 2, \ldots, n$

For the constraint function $f_{2,i}(\mathbf{w}, s)$ we have the following:

- $f_{2,i}(\mathbf{w}, s) = \mathbf{w}_i - 1 - s, i = 1, 2, \ldots, n$
- $\nabla f_{2,i}(\mathbf{w}, s) = \mathbf{e}_i - \mathbf{e}_{n+1}, i = 1, 2, \ldots, n$
- $\nabla^2 f_{2,i}(\mathbf{w}, s) = \mathbf{0}, i = 1, 2, \ldots, n$

For the constraint function $f_3(\mathbf{w}, s)$ we have the following:

- $f_3(\mathbf{w}, s) = \mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w} - v - s$
- $\nabla f_3(\mathbf{w}, s) = \begin{bmatrix} 2\boldsymbol{\Sigma}\mathbf{w} \\ -1 \end{bmatrix}$
- $\nabla^2 f_3(\mathbf{w}, s) = \begin{bmatrix} 2\boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix}$

For the logarithmic barrier function $\phi(\mathbf{w}, s)$ we have the following:

$$\phi(\mathbf{w}, s) = -\sum_{i=1}^{n} \log(\mathbf{w}_i + s) - \sum_{i=1}^{n} \log(s + 1 - \mathbf{w}_i) - \log\left(s + v - \mathbf{w}^T \mathbf{\Sigma} \mathbf{w}\right)$$

$$\nabla\phi(\mathbf{w}, s) = \sum_{i=1}^{n} \frac{1}{\mathbf{w}_i + s}(-\mathbf{e}_i - \mathbf{e}_{n+1}) + \sum_{i=1}^{n} \frac{1}{s + 1 - \mathbf{w}_i}(\mathbf{e}_i - \mathbf{e}_{n+1}) + \frac{1}{s + v - \mathbf{w}^T \mathbf{\Sigma} \mathbf{w}} \begin{bmatrix} 2\mathbf{\Sigma}\mathbf{w} \\ -1 \end{bmatrix}$$

$$\begin{aligned} \nabla^2\phi(\mathbf{w}, s) = & \sum_{i=1}^{n} \frac{1}{(-\mathbf{w}_i - s)^2}(-\mathbf{e}_i - \mathbf{e}_{n+1})(-\mathbf{e}_i - \mathbf{e}_{n+1})^T + \\ & + \sum_{i=1}^{n} \frac{1}{(\mathbf{w}_i - 1 - s)^2}(\mathbf{e}_i - \mathbf{e}_{n+1})(\mathbf{e}_i - \mathbf{e}_{n+1})^T + \\ & + \frac{1}{\left(\mathbf{w}^T \mathbf{\Sigma} \mathbf{w} - v - s\right)^2} \begin{bmatrix} 2\mathbf{\Sigma}\mathbf{w} \\ -1 \end{bmatrix} \begin{bmatrix} 2\mathbf{\Sigma}\mathbf{w} \\ -1 \end{bmatrix} + \frac{1}{s + v - \mathbf{w}^T \mathbf{\Sigma} \mathbf{w}} \begin{bmatrix} 2\mathbf{\Sigma} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \end{aligned}$$

Phase II

For the cost function $f_0(\mathbf{w})$ we have the following:

- $f_0(\mathbf{w}) = -\boldsymbol{\mu}^T \mathbf{w}$
- $\nabla f_0(\mathbf{w}) = -\boldsymbol{\mu}$
- $\nabla^2 f_0(\mathbf{w}) = \mathbf{0}$

For the constraint function $f_{1,i}(\mathbf{w})$ we have the following:

- $f_{1,i}(\mathbf{w}) = -\mathbf{w}_i, i = 1, 2, \ldots, n$
- $\nabla f_{1,i}(\mathbf{w}) = -\mathbf{e}_i, i = 1, 2, \ldots, n$
- $\nabla^2 f_{1,i}(\mathbf{w}) = \mathbf{0}, i = 1, 2, \ldots, n$

For the constraint function $f_{2,i}(\mathbf{w})$ we have the following:

- $f_{2,i}(\mathbf{w}) = \mathbf{w}_i - 1, i = 1, 2, \ldots, n$
- $\nabla f_{2,i}(\mathbf{w}) = \mathbf{e}_i, i = 1, 2, \ldots, n$
- $\nabla^2 f_{2,i}(\mathbf{w}) = \mathbf{0}, i = 1, 2, \ldots, n$

For the constraint function $f_3(\mathbf{w})$ we have the following:

- $f_3(\mathbf{w}) = \mathbf{w}^T \mathbf{\Sigma} \mathbf{w} - v$
- $\nabla f_3(\mathbf{w}) = 2\mathbf{\Sigma}\mathbf{w}$
- $\nabla^2 f_3(\mathbf{w}) = 2\mathbf{\Sigma}$

For the logarithmic barrier function $\phi(\mathbf{w})$ we have the following:

- $\phi(\mathbf{w}) = -\sum_{i=1}^{n} \log(\mathbf{w}_i) - \sum_{i=1}^{n} \log(1 - \mathbf{w}_i) - \log(v - \mathbf{w}^T \mathbf{\Sigma} \mathbf{w})$
- $\nabla\phi(\mathbf{w}) = -\sum_{i=1}^{n} \frac{1}{\mathbf{w}_i}\mathbf{e}_i - \sum_{i=1}^{n} \frac{1}{\mathbf{w}_i - 1}\mathbf{e}_i - \frac{1}{\mathbf{w}^T\mathbf{\Sigma}\mathbf{w} - v} 2\mathbf{\Sigma}\mathbf{w}$
- $\nabla^2\phi(\mathbf{w}) = \sum_{i=1}^{n} \frac{1}{\mathbf{w}_i^2}\mathbf{e}_i\mathbf{e}_i^T + \sum_{i=1}^{n} \frac{1}{(\mathbf{w}_i - 1)^2}\mathbf{e}_i\mathbf{e}_i^T + \frac{1}{\left(\mathbf{w}^T\mathbf{\Sigma}\mathbf{w} - v\right)^2} 4\mathbf{\Sigma}\mathbf{w}\mathbf{w}^T\mathbf{\Sigma} + \frac{1}{v - \mathbf{w}^T\mathbf{\Sigma}\mathbf{w}} 2\mathbf{\Sigma}$

Maximum Sharpe ratio

The original problem of Sharpe ratio maximization is not convex and there is not any analytic algorithm to solve it. Next, we will see how we can transform it to another equivalent problem that is convex with linear inequality constraints and affine equality constraints.

The original problem after some unimportant changes in the notations takes the following form:

$$\begin{aligned} \max_{\mathbf{w}} \quad & SR \equiv \frac{\boldsymbol{\mu}^T \mathbf{w} - R_f}{\sqrt{\mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w}}} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{w} = \mathbf{b} \text{ , where } \mathbf{A} \equiv \mathbf{1}^T \text{ and } \mathbf{b} \equiv 1 \\ & \mathbf{w} \geq \mathbf{0} \\ & \mathbf{C}\mathbf{w} \geq \mathbf{d} \text{ , where } \mathbf{C} \equiv -\mathbf{I}_n \text{ and } \mathbf{d} \equiv -\mathbf{1} \end{aligned} \tag{10}$$

As we have already said, problem (10) is difficult because of the nature of its objective. However, under a reasonable assumption, it can be reduced to a standard convex quadratic program.

The assumption we make is that there exists a vector $\mathbf{w}$ satisfying all the constraints of problem (10) such that $\boldsymbol{\mu}^T\mathbf{w} - R_f > 0$. This assumption is reasonable as it simply says that our universe of assets is able to beat the risk-free rate of return.

Our approach is as follows. Given an asset vector $\mathbf{w}$, we define:

$$f_0(\mathbf{w}) = \frac{\boldsymbol{\mu}^T\mathbf{w} - R_f}{\sqrt{\mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w}}}$$

Since $\mathbf{1}^T\mathbf{w} = 1$, we have:

$$f_0(\mathbf{w}) = \frac{\boldsymbol{\mu}^T\mathbf{w} - R_f}{\sqrt{\mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w}}} = \frac{\boldsymbol{\mu}^T\mathbf{w} - R_f\mathbf{1}^T\mathbf{w}}{\sqrt{\mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w}}} = \frac{\left(\boldsymbol{\mu}^T - R_f\mathbf{1}^T\right)\mathbf{w}}{\sqrt{\mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w}}} = \frac{\hat{\boldsymbol{\mu}}^T\mathbf{w}}{\sqrt{\mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w}}}$$

where we define $\hat{\mu}_i = \mu_i - R_f, i = 1, 2, \ldots, n$.

Using this fact, we note that for any vector $\mathbf{w}$ with $\mathbf{1}^T\mathbf{w} = 1$ and any scalar $\lambda > 0$, it holds that $f_0(\lambda\mathbf{w}) = f_0(\mathbf{w})$.

Now we can state our optimization problem.

Let $\hat{\mathbf{C}}$ be the matrix whose i, j-entry is $C_{i,j} - d_i$.

The problem we consider is:

$$\begin{aligned} \max_{\mathbf{y}} \quad & \frac{1}{\sqrt{\mathbf{y}^T \boldsymbol{\Sigma} \mathbf{y}}} \\ \text{s.t.} \quad & \hat{\mathbf{A}}\mathbf{y} = \mathbf{b}, \ \text{, where } \hat{\mathbf{A}} \equiv \hat{\boldsymbol{\mu}}^T \text{ and } \mathbf{b} \equiv 1 \\ & \mathbf{y} \geq \mathbf{0} \\ & \hat{\mathbf{C}}\mathbf{y} \geq \mathbf{0} \end{aligned} \tag{11}$$

To see that problems (10) and (11) are indeed equivalent, suppose that $\mathbf{y}_*$ is an optimal solution to (11). Notice that because of the equality constraint $\hat{\mathbf{A}}\mathbf{y} = \mathbf{b}$, $\mathbf{y}_*$ is not identically zero, and so by $\mathbf{y} \geq \mathbf{0}$, it is $\sum_{i=1}^{n} y_{*,i} > 0$. Define the vector:

$$\mathbf{w}_* = \frac{\mathbf{y}_*}{\sum_{i=1}^{n} y_{*,i}}$$

Then, by construction it holds that:

$$\sum_{i=1}^{n} w_{*,i} = 1$$

Further, since $\mathbf{y}$ satisfies $\hat{\mathbf{C}}\mathbf{y} \geq \mathbf{0}$, then for any row i we have that:

$$\sum_{j=1}^{n} (c_{i,j} - d_i)\, y_{*,j} \geq 0$$

or in other words:

$$\sum_{j=1}^{n} c_{i,j} y_{*,j} \geq \left(\sum_{j=1}^{n} y_{*,j} \right) d_i$$

And as a consequence:

$$\sum_{j=1}^{n} c_{i,j} w_{*,j} \geq d_i$$

Therefore, $\mathbf{w}_*$ is feasible for problem (10). Further, as we observed before it holds that:

$$f_0(\mathbf{w}_*) = f_0(\mathbf{y}_*) = \frac{1}{\sqrt{\mathbf{y}_*^T \boldsymbol{\Sigma} \mathbf{y}_*}}$$

since $\hat{\boldsymbol{\mu}}^T \mathbf{y} = 1$.

In summary, the value of problem (10) is at least as large as the value of problem (11). The converse is proved in a similar way. So, indeed, problems (10) and (11) are equivalent.

So we just have to solve problem (11). But this is clearly equivalent to the following problem:

$$\begin{aligned} \min_{\mathbf{y}} \quad & \mathbf{y}^T \boldsymbol{\Sigma} \mathbf{y} \\ \text{s.t.} \quad & \hat{\mathbf{A}} \mathbf{y} = \mathbf{b}, \text{ , where } \hat{\mathbf{A}} \equiv \hat{\boldsymbol{\mu}}^T \text{ and } \mathbf{b} \equiv 1 \\ & \mathbf{y} \geq \mathbf{0} \\ & \hat{\mathbf{C}} \mathbf{y} \geq \mathbf{0} \end{aligned} \tag{12}$$

which is just a standard quadratic program.

In order to solve problem (12), using the interior point barrier method, we write it in the following form:

$$\begin{aligned} \min_{\mathbf{y}} \quad & f_0(\mathbf{y}) \equiv \mathbf{y}^T \boldsymbol{\Sigma} \mathbf{y} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{y} = \mathbf{b} \ , \text{ where } \mathbf{A} \equiv \hat{\boldsymbol{\mu}}^T \text{ and } \mathbf{b} \equiv 1 \\ & f_{1,i}(\mathbf{y}) \equiv -y_i \leq 0 \ , \ i = 1, 2, \ldots n \\ & f_{2,i}(\mathbf{y}) \equiv -\hat{\mathbf{c}}_i^T \mathbf{y} \leq 0 \ , \ i = 1, 2, \ldots n \end{aligned} \tag{13}$$

Then, the solution to the original problem is given by:

$$\mathbf{w} = \frac{\mathbf{y}}{\sum_{i=1}^{n} y_i}$$

Phase I

For the cost function $f_0(\mathbf{y}, s)$ we have the following:

- $f_0(\mathbf{y}, s) = s$
- $\nabla f_0(\mathbf{y}, s) = \mathbf{e}_{n+1}$
- $\nabla^2 f_0(\mathbf{y}, s) = \mathbf{0}$

For the constraint function $f_{1,i}(\mathbf{y}, s)$ we have the following:

- $f_{1,i}(\mathbf{y}, s) = -\mathbf{y}_i - s, i = 1, 2, \ldots, n$
- $\nabla f_{1,i}(\mathbf{y}, s) = -\mathbf{e}_i - \mathbf{e}_{n+1}, i = 1, 2, \ldots, n$
- $\nabla^2 f_{1,i}(\mathbf{y}, s) = \mathbf{0}, i = 1, 2, \ldots, n$

For the constraint function $f_2(\mathbf{y}, s)$ we have the following:

- $f_{2,i}(\mathbf{y}, s) = -\hat{\mathbf{c}}_i{}^T \mathbf{y} - s, i = 1, 2, \ldots, n$
- $\nabla f_{2,i}(\mathbf{y}, s) = \begin{bmatrix} \hat{\mathbf{c}}_i \\ 0 \end{bmatrix} - \mathbf{e}_{n+1}, i = 1, 2, \ldots, n$
- $\nabla^2 f_{2,i}(\mathbf{y}, s) = \mathbf{0}, i = 1, 2, \ldots, n$

For the logarithmic barrier function $\phi(\mathbf{y}, s)$ we have the following:

$$\phi(\mathbf{y}, s) = -\sum_{i=1}^{n} \log(\mathbf{y}_i + s) - \sum_{i=1}^{n} \log\left(\hat{\mathbf{c}}_i{}^T \mathbf{y} + s\right)$$

$$\nabla\phi(\mathbf{y}, s) = \sum_{i=1}^{n} \frac{1}{\mathbf{y}_i + s}(-\mathbf{e}_i - \mathbf{e}_{n+1}) + \sum_{i=1}^{n} \frac{1}{\hat{\mathbf{c}}_i{}^T \mathbf{y} + s}\left(-\begin{bmatrix}\hat{\mathbf{c}}_i \\ 0\end{bmatrix} - \mathbf{e}_{n+1}\right)$$

$$\begin{aligned}\nabla^2\phi(\mathbf{y}, s) = &\sum_{i=1}^{n} \frac{1}{(-\mathbf{y}_i - s)^2}(-\mathbf{e}_i - \mathbf{e}_{n+1})(-\mathbf{e}_i - \mathbf{e}_{n+1})^T + \\ &+ \sum_{i=1}^{n} \frac{1}{\left(-\hat{\mathbf{c}}_i{}^T \mathbf{y} - s\right)^2}\left(-\begin{bmatrix}\hat{\mathbf{c}}_i \\ 0\end{bmatrix} - \mathbf{e}_{n+1}\right)\left(-\begin{bmatrix}\hat{\mathbf{c}}_i \\ 0\end{bmatrix} - \mathbf{e}_{n+1}\right)^T\end{aligned}$$

Phase II

For the cost function $f_0(\mathbf{y})$ we have the following:

- $f_0(\mathbf{y}) = \mathbf{y}^T \boldsymbol{\Sigma} \mathbf{y}$
- $\nabla f_0(\mathbf{y}) = 2\boldsymbol{\Sigma}\mathbf{y}$
- $\nabla^2 f_0(\mathbf{y}) = 2\boldsymbol{\Sigma}$

For the constraint function $f_{1,i}(\mathbf{y})$ we have the following:

- $f_{1,i}(\mathbf{y}) = -\mathbf{y}_i, i = 1, 2, \ldots, n$
- $\nabla f_{1,i}(\mathbf{y}) = -\mathbf{e}_i, i = 1, 2, \ldots, n$
- $\nabla^2 f_{1,i}(\mathbf{y}) = \mathbf{0}, i = 1, 2, \ldots, n$

For the constraint function $f_{2,i}(\mathbf{y})$ we have the following:

- $f_{2,i}(\mathbf{y}) = -\hat{\mathbf{c}}_i{}^T\mathbf{y}, i = 1, 2, \ldots, n$
- $\nabla f_{2,i}(\mathbf{y}) = -\hat{\mathbf{c}}_i, i = 1, 2, \ldots, n$
- $\nabla^2 f_{2,i}(\mathbf{y}) = \mathbf{0}, i = 1, 2, \ldots, n$

For the logarithmic barrier function $\phi(\mathbf{y})$ we have the following:

- $\phi(\mathbf{y}) = -\sum_{i=1}^{n} \log(\mathbf{y}_i) - \sum_{i=1}^{n} \log\left(\hat{\mathbf{c}}_i{}^T\mathbf{y}\right)$
- $\nabla\phi(\mathbf{y}) = -\sum_{i=1}^{n} \frac{1}{\mathbf{y}_i}\mathbf{e}_i - \sum_{i=1}^{n} \frac{1}{\hat{\mathbf{c}}_i{}^T\mathbf{y}}\hat{\mathbf{c}}_i$
- $\nabla^2\phi(\mathbf{y}) = \sum_{i=1}^{n} \frac{1}{\mathbf{y}_i^2}\mathbf{e}_i\mathbf{e}_i^T + \sum_{i=1}^{n} \frac{1}{\left(\hat{\mathbf{c}}_i{}^T\mathbf{y}\right)^2}\hat{\mathbf{c}}_i\hat{\mathbf{c}}_i{}^T$

Fast Dual Proximal Gradient (FDPG) Method

Consider the following optimization problem:

$$f_{opt} = \min_{\mathbf{w} \in \mathbb{E}} f(\mathbf{w}) + g(\mathcal{A}(\mathbf{w})) \tag{14}$$

where the following assumptions are made:

- $f : \mathbb{E} \to (-\infty, +\infty]$ is proper, closed and σ-strongly convex.
- $g : \mathbb{V} \to (-\infty, +\infty]$ is proper, closed and convex.
- $\mathcal{A} : \mathbb{E} \to \mathbb{V}$ is a linear transformation.
- There exists $\hat{\mathbf{w}} \in \text{ri}(\text{dom}(f))$ and $\hat{\mathbf{z}} \in \text{ri}(\text{dom}(g))$ such that $\mathcal{A}(\hat{\mathbf{w}}) = \hat{\mathbf{z}}$.

This problem has a unique optimal solution, which we denote by $\mathbf{w}_*$.

The Fast Dual Proximal Gradient (FDPG) Method, that solves problem (14) and achieves a $\mathcal{O}\left(\frac{1}{k^2}\right)$ rate of convergence of the primal sequence, takes the following form:

$\mathbf{s}_0 = \mathbf{y}_0 \in \mathbb{V}, t_0 = 1, k = 0, L \geq L_F = \frac{\|\mathcal{A}\|^2}{\sigma}$

while (1) **do**

$\mathbf{u}_k = \underset{\mathbf{u}}{\operatorname{argmax}} \left\{ \langle \mathbf{u}, \mathcal{A}^T(\mathbf{s}_k) \rangle - f(\mathbf{u}) \right\}$

$\mathbf{y}_{k+1} = \mathbf{s}_k - \frac{1}{L}\mathcal{A}(\mathbf{u}_k) + \frac{1}{L}\text{prox}_{Lg}(\mathcal{A}(\mathbf{u}_k) - L\mathbf{z}_k)$

$t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$

$\mathbf{s}_{k+1} = \mathbf{y}_{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(\mathbf{y}_{k+1} - \mathbf{y}_k)$

$k = k + 1$

if (*convergence*) **then**

quit

end

end

Algorithm 4: The Fast Dual Proximal Gradient (FDPG) Method

Here, the primal sequence is given by $\mathbf{w}_k = \underset{\mathbf{w} \in \mathbb{E}}{\operatorname{argmax}} \left\{ \langle \mathbf{w}, \mathcal{A}^T(\mathbf{y}_k) \rangle - f(\mathbf{w}) \right\}$.

Let $\mathbf{d}, \mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^n$, with $\mathbf{v}_1 \leq \mathbf{v}_2$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$. We want to compute the projection of $\mathbf{d}$ onto set $C = \{\mathbf{w} \in \mathbb{R}^n | \mathbf{A}\mathbf{w} = \mathbf{b}, \mathbf{v}_1 \leq \mathbf{w} \leq \mathbf{v}_2\}$.

In essence, we have to compute an orthogonal projection onto the intersection of closed convex sets. Given p closed and convex sets $C_1, \ldots, C_p \subset \mathbb{E}$ and a point $\mathbf{d} \in \mathbb{E}$, the orthogonal projection of $\mathbf{d}$ onto the intersection is the optimal solution of the problem

$$\min_{\mathbf{w} \in \mathbb{R}^n} \left\{ \frac{1}{2} \|\mathbf{w} - \mathbf{d}\|^2 : \mathbf{w} \in \cap_{i=1}^p C_i \right\} \tag{15}$$

We will assume that the intersection $\cap_{i=1}^p C_i$ is nonempty and that projecting onto each set C_i is an easy task. Problem (15) fits model (14) with $\mathbb{V} = \mathbb{E}^p$, $\mathcal{A}(\mathbf{z}) = (\mathbf{z}, \mathbf{z}, \ldots, \mathbf{z})$ (this is p times) for any $\mathbf{z} \in \mathbb{E}$, $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w} - \mathbf{d}\|^2$ and $g(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_p) = \sum_{i=1}^p \delta_{C_i}(\mathbf{w}_i)$.

We have that:

- $\text{argmax}_{\mathbf{w}}\{\langle \mathbf{v}, \mathbf{w}\rangle - f(\mathbf{w})\} = \mathbf{v} + \mathbf{d}$ for any $\mathbf{v} \in \mathbb{E}$
- $\|\mathcal{A}\| = p$
- $\sigma = 1$
- $\mathcal{A}^T(\mathbf{y}) = \sum_{i=1}^{p} y_i$ for any $\mathbf{y} \in \mathbb{E}^p$
- $\text{prox}_{Lg}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_p) = (P_{C_1}(\mathbf{v}_1), P_{C_2}(\mathbf{v}_2), \ldots, P_{C_p}(\mathbf{v}_p))$ for any $\mathbf{v} \in \mathbb{E}^p$.

Using these facts, the FDPG method for solving problem (15) can be explicitly written as follows:

$\mathbf{s}_0 = \mathbf{y}_0 \in \mathbb{R}^m, t_0 = 1, k = 0, L \geq \|\mathbf{A}\|_{2,2}^2$

while (1) **do**

$\mathbf{u}_k = \sum_{i=1}^{p} \mathbf{s}_{i,k} + \mathbf{d}$

$\mathbf{y}_{i,k+1} = \mathbf{s}_{i,k} - \frac{1}{L}\mathbf{u}_k + \frac{1}{L}P_{C_i}(\mathbf{u}_k - L\mathbf{s}_{i,l})$ for $i = 1, 2, \ldots, p$

$t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$

$\mathbf{s}_{k+1} = \mathbf{y}_{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(\mathbf{y}_{k+1} - \mathbf{y}_k)$

$k = k + 1$

if $(convergence)$ **then**

quit

end

end

Remember that the primal sequence is given by $\mathbf{w}_k = \mathbf{A}^T\mathbf{y}_k + \mathbf{d}$.

For convergence we check whether $\|\frac{\mathbf{w}_k - \mathbf{w}_{k+1}}{\mathbf{w}_k}\|_2 \approx 0$.

In our problems, we adjust the corresponding constraints to an appropriate form of $\mathbf{Aw} = \mathbf{b}$ along with box constraints, in order to compute the projections needed.

We have that $C = C_1 \cap C_2$, where:
$C_1 = \{\mathbf{w} \in \mathbb{R}^n : \mathbf{Aw} = \mathbf{b}\}$ and $C_2 = \{\mathbf{w} \in \mathbb{R}^n : \mathbf{v}_1 \leq \mathbf{w} \leq \mathbf{v}_2\}$.

The projections onto set C_1 is given by:
$P_{C_1}(\mathbf{w}) = \mathbf{w} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}(\mathbf{Aw} - \mathbf{b})$.

In particular, we will have either $\begin{bmatrix}\mathbf{1}^T\end{bmatrix} \cdot \mathbf{w} = \begin{bmatrix}0\end{bmatrix}$ or $\begin{bmatrix}\mathbf{1}^T \\ \boldsymbol{\mu}^T\end{bmatrix} \cdot \mathbf{w} = \begin{bmatrix}0 \\ r\end{bmatrix}$

The projection onto set C_2 is given by:
$P_{C_2}(\mathbf{w}) = P_{\text{Box}[\mathbf{v}_1,\mathbf{v}_2]}(\mathbf{w})_i = \min\{\max\{w_i, v_{1,i}\}, v_{2,i}\}$ for $i = 1, \ldots, n$.

With this algorithm we compute the projections needed for the other algorithms mentioned before.