Optimization Techniques in Finance

6. Large scale optimization methods and machine learning. I

Andrew Lesniewski

Baruch College
New York

Fall 2019

Outline

Large scale machine learning

- In this and the following sets of notes we describe robust methods for large scale optimization problems suitable for addressing machine learning / statistics problems with huge datasets.
- Key characteristics of these methods are:
  - (i) The solution to a large problem can be broken up into smaller tasks which can be carried out in a decentralized manner on different devices or processors.
  - (ii) The devices / processors coordinate the tasks by passing small amounts of information among each other.

Motivation: dual ascent

- Consider again a convex optimization problem with equality constraints:

$$\min_{x \in \mathbb{R}^n} f(x), \quad \text{subject to } Ax = b, \tag{1}$$

  where $b \in \mathbb{R}^m$, and $A \in \mathrm{Mat}_{m,n}(\mathbb{R})$.
- Its Lagrange function is

$$L(x, \lambda) = f(x) + \lambda^{\mathrm{T}}(Ax - b),$$

  and the dual Lagrange function is

$$\begin{aligned} q(\lambda) &= \inf_{x \in \mathbb{R}^n} \big(f(x) + \lambda^{\mathrm{T}}(Ax - b)\big) \\ &= - \sup_{x \in \mathbb{R}^n} \big(- (A^{\mathrm{T}}\lambda)^{\mathrm{T}} x - f(x)\big) - b^{\mathrm{T}}\lambda \\ &= -f^*(-A^{\mathrm{T}}\lambda) - b^{\mathrm{T}}\lambda, \end{aligned}$$

  where $f^*(\lambda)$ is the conjugate of $f(x)$ defined in Problem 2 of Homework #4.
- The dual problem reads

$$\max_{\lambda \in \mathbb{R}^m} q(\lambda).$$

Motivation: dual ascent

- As we saw in Lecture Notes #4, under the assumption of strong duality, the optimal values of the primal and dual problems are equal.
- The optimal x^* is given by

$$x^* = \arg\min_x L(x, \lambda^*),$$

where λ^* is the solution to the dual problem.

- We can solve the dual problem by the *dual ascent method*.
- Assuming that $q(\lambda)$ is differentiable, we may calculate its gradient in the following way: (i) find $x' = \arg\min_x L(x, \lambda)$, and (ii) set $\nabla q(\lambda) = Ax' - b$.
- This leads to the following iteration scheme:

$$x_{k+1} = \arg\min_x L(x, \lambda_k),$$
$$\lambda_{k+1} = \lambda_k + \alpha_k(Ax_{k+1} - b),$$

where α_k is the step size.

- This method is called dual ascent since it increases the dual function in each iteration, $q(\lambda_{k+1}) > q(\lambda_k)$ (with suitable choices of α_k).

Motivation: Augmented Lagrangian method

- Let us apply the augmented Lagrangian method (discussed in Lecture Notes #2) to problem (1).
- The augmented Lagrange function reads

$$L_\mu(x,\lambda) = f(x) + \lambda^{\mathrm{T}}(Ax-b) + \frac{\mu}{2}\|Ax-b\|^2. \tag{2}$$

- Applying the dual ascent method to the augmented Lagrange function yields the following scheme:

$$\begin{aligned} x_{k+1} &= \arg\min_x L_\mu(x,\lambda_k), \\ \lambda_{k+1} &= \lambda_k + \mu(Ax_{k+1}-b), \end{aligned}$$

which is a special case of the scheme discussed in Lecture Notes #2.

ADMM

- We now formulate the *alternating direction method of multipliers* (ADMM).
- Consider two convex functions $f(x)$, where $x \in \mathbb{R}^n$, and $g(z)$, where $z \in \mathbb{R}^m$, and two matrices $A \in \mathrm{Mat}_{p,n}(\mathbb{R})$ and $B \in \mathrm{Mat}_{p,m}(\mathbb{R})$. The problem is to find

$$\min\ f(x) + g(z) \quad \text{subject to } Ax + Bz = c, \tag{3}$$

  where $c \in \mathbb{R}^p$ is a constant.
- Notice that structure of the objective function: it is a sum of two functions of different sets of variables. The constraint is a linear relation between these variables.
- As in the augmented Lagrangian method we define the augmented Lagrange function:

$$L_\rho(x, z, \lambda) = f(x) + g(z) + \lambda^{\mathrm{T}}(Ax + Bz - c) + \frac{1}{2}\rho\|Ax + Bz - c\|^2, \tag{4}$$

  where $\rho > 0$ is the penalty parameter, and where $\|\cdot\|$ denotes the Euclidean norm.

ADMM

- The ADMM algorithm is an iterative procedure formulated as follows. Find an initial value (x_0, z_0, λ_0) and iterate until stopping criterion is met:

$$\begin{aligned} x_{k+1} &= \arg\min_x L_\rho(x, z_k, \lambda_k) \\ z_{k+1} &= \arg\min_z L_\rho(x_{k+1}, z, \lambda_k) \\ \lambda_{k+1} &= \lambda_k + \rho(Ax_{k+1} + Bz_{k+1} - c) \end{aligned} \tag{5}$$

- The key difference between ADMM and the augmented Lagrange method is that, in the latter, we would find the *joint* minimizer

$$(x_{k+1}, z_{k+1}) = \arg\min_{x,z} L_\rho(x, z, \lambda_k),$$

  while in the former we alternate the search directions between x and z.
- This is somewhat reminiscent of the Gauss-Seidl method from linear algebra.
- We emphasize that in ADMM ρ is a tunable parameter whose value is determined depending on the problem at hand and, unlike in the barrier method, it is not sent to ∞.

Scaled form of ADMM

- ADMM can be formulated in an equivalent (but often more convenient) way, by completing the square in the augmented Lagrangian:

$$\begin{aligned} L_\rho(x, z, \lambda) &= f(x) + g(z) + \lambda^{\mathrm{T}}(Ax + Bz - c) + \frac{1}{2}\rho\|Ax + Bz - c\|^2 \\ &= f(x) + g(z) + \frac{1}{2}\rho\|Ax + Bz - c + \zeta\|^2 - \frac{1}{2}\rho\|\zeta\|^2, \end{aligned}$$

  where $\zeta = \frac{1}{\rho}\lambda$ is the scaled dual variable.

- In terms of x, z, and ζ, the ADMM algorithm reads:

$$\begin{aligned} x_{k+1} &= \arg\min_x \left(f(x) + \frac{1}{2}\rho\|Ax + Bz_k - c + \zeta_k\|^2 \right) \\ z_{k+1} &= \arg\min_z \left(g(z) + \frac{1}{2}\rho\|Ax_{k+1} + Bz - c + \zeta_k\|^2 \right) \\ \zeta_{k+1} &= \zeta_k + Ax_{k+1} + Bz_{k+1} - c \end{aligned} \tag{6}$$

- Notice that ζ_k is the running sum of residuals $Ax_k + Bz_k - c$ from feasibility.

ADMM

- Before moving on, let us state a theorem guaranteeing the convergence of the algorithm.
- If $f(x)$ is a function on $\mathbb{R}^n$, its *epigraph* is defined as the set

$$\text{epi}(f) = \{(x,t) \in \mathbb{R}^n \times \mathbb{R} : \ f(x) < t\}.$$

- A function $f(x,y)$ on $\mathbb{R}^n \times \mathbb{R}^m$ is said to have a *saddle point*, if there exists a point $(\overline{x}, \overline{y})$ such that for all (x,y)

$$f(\overline{x}, y) \leq f(\overline{x}, \overline{y}) \leq f(x, \overline{y}). \tag{7}$$

- In other words, starting from the point $(\overline{x}, \overline{y})$, the function does not increase in the direction of y and does not decrease in the direction of x.

ADMM

- *Theorem*. Assume that:
  - (i) $\mathrm{epi}(f)$ and $\mathrm{epi}(g)$ are nonempty, closed, and convex,
  - (ii) the non-augmented Lagrange function $L_0(x, z, \lambda)$ has a saddle point (with respect to the groups of variables (x, z) and λ).

  Then:
  - (i) the iterates (x_k, z_k) converge to a feasible point, i.e. the residuals go to zero $Ax_k + Bz_k - c \to 0$, as $k \to \infty$,
  - (ii) the objective function converges to its optimal value, i.e. $f(x_k) + g(z_k) \to p^*$, as $k \to \infty$,
  - (iii) the dual variables converge to their optimal point, i.e. $\lambda_k \to \lambda^*$, as $k \to \infty$.
- For the proof, see [1].
- In the following, we assume that the assumptions of the theorem above hold.

Proximal operators

- In special, but practically important cases, the x and z searches can be conducted more efficiently.
- Notice that we can express the x-update of in the equations defining ADMM (unscaled or scaled) as

$$x' = \arg\min_x \left(f(x) + \tfrac{1}{2}\,\rho\|Ax - v\|^2\right), \tag{8}$$

  where v is a constant vector ($v = -Bz + c$ in the unscaled vector or $v = -Bz + c + \zeta$ in the scaled version).
- An analogous formula holds for the z search.
- This justifies the following abstract definition: the *proximal operator* $\mathrm{prox}_{f,\rho}$ is defined by

$$\mathrm{prox}_{f,\rho}(v) = \arg\min_x \left(f(x) + \tfrac{1}{2}\,\rho\|x - v\|^2\right). \tag{9}$$

- For a detailed survey of proximal operators see [2].

Indicator function

- *Example 1*. Consider the case of the *indicator function* $f(x) = \chi_C(x)$ of a set $C \subset \mathbb{R}^n$, defined as follows:

$$\chi_C(x) = \begin{cases} 0, & \text{if } x \in C, \\ \infty, & \text{otherwise.} \end{cases} \tag{10}$$

- Consider the case of $A = I$.
- Clearly, if $v \in C$, then $x' = v$. Otherwise, x' is the element of C with the shortest Euclidean distance to v.
- This defines a mapping $\Pi_C : \mathbb{R}^n \to C$, called the *projection* onto the set C.
- In other words,

$$\begin{aligned} x' &= \text{ projection of } v \text{ onto the set } C \\ &= \Pi_C(v). \end{aligned} \tag{11}$$

Quadratic objective function

- *Example 2*. Consider first the unconstrained quadratic objective function:

$$f(x) = \frac{1}{2} x^{\mathrm{T}} P x + q^{\mathrm{T}} x + r, \tag{12}$$

with positive semidefinite P.

- In this, we can solve the optimization problem (8) explicitly:

$$x' = (PA^{\mathrm{T}} A + \rho)^{-1} (\rho A^{\mathrm{T}} v - q), \tag{13}$$

assuming that $P + \rho A^{\mathrm{T}} A$ is invertible.

- For matrix inversion, one should use an appropriate direct method of numerical linear algebra, that suits best the structure of the matrix $P + \rho A^{\mathrm{T}} A$.
- In cases when P^{-1} is known explicitly, we can use the matrix inversion lemma, which we have already discussed in these notes:

$$(P + \rho A^{\mathrm{T}} A)^{-1} = P^{-1} - \rho P^{-1} A^{\mathrm{T}} (I + \rho A P^{-1} A^{\mathrm{T}})^{-1} A P^{-1}, \tag{14}$$

assuming that all the inverses exist.

Separable function

- *Example 3*. If x can be decomposed into subvectors $x = (x_1, \ldots, x_p)$ such that (i) the matrix A is block diagonal with respect to this decomposition, and (ii) $f(x)$ is separable, i.e. $f(x) = f_1(x_1) + \ldots + f_p(x_p)$, then the augmented Lagrange function is also separable.
- In this case, the x minimizations for each of the subvectors x_i can be carried out in parallel.
- In particular, if the decomposition extends all the way to individual components of $x = (x_1, \ldots, x_n)$, $f(x) = f_1(x_1) + \ldots + f_n(x_n)$, the x minimizations can be carried out through n independent scalar minimizations.
- One such example is the L_1 norm.

Absolute value

- *Example 4.* Consider the non-smooth function $f(x) = \alpha|x|$, $x \in \mathbb{R}$, where $\alpha > 0$:

$$x' = \arg\min_x \left(\alpha|x| + \tfrac{1}{2}\,\rho(x - v)^2\right).$$

- Even though $|x|$ is not differentiable, the minimizer is easy to find and is given by

$$x' = S_{\alpha/\rho}(v), \tag{15}$$

where the *soft thresholding operator* S_κ is given by

$$S_\kappa(v) = \begin{cases} v - \kappa, & \text{if } v > \kappa, \\ 0, & \text{if } |v| \le \kappa, \\ v + \kappa, & \text{if } v < -\kappa. \end{cases} \tag{16}$$

- Note that this can be written as

$$S_\kappa(v) = (1 - \kappa/|v|)^+ v, \tag{17}$$

showing that S_κ has the shrinking effect on v.

Convex optimization revisited

- Consider a constrained convex optimization problem:

$$\min_x f(x) \quad \text{subject to } x \in C, \tag{18}$$

  where $C \subset \mathbb{R}^n$ is a convex set.
- This can be written in the ADMM form as follows:

$$\min_{x,z} f(x) + g(z) \quad \text{subject to } x - z = 0, \tag{19}$$

  where $g(z) = \chi_C(z)$ is the indicator function of C, as defined in (10).
- The augmented Lagrange function (using the scaled dual variable) reads

$$L_\rho(x, z, \zeta) = f(x) + g(z) + \frac{1}{2}\rho\|x - z + \zeta\|^2. \tag{20}$$

Convex optimization revisited

- As a consequence, the ADMM algorith reads

$$\begin{aligned} x_{k+1} &= \arg\min_x \left(f(x) + \frac{1}{2}\rho \|x - z_k + \zeta_k\|^2 \right) \\ z_{k+1} &= \Pi_C\left(x_{k+1} + \zeta_k\right) \\ \zeta_{k+1} &= \zeta_k + x_{k+1} - z_{k+1} \end{aligned} \tag{21}$$

- Note that the x update is given by the proximal operator $\mathrm{prox}_{f,\rho}(z_k - \zeta_k)$.

LASSO regression

- Recall that the problem of estimating a linear regression model with LASSO regularization is the following optimization problem:

$$\min_x \ \frac{1}{2}\|Ax - b\|_2^2 + \alpha\|x\|_1, \tag{22}$$

where A is a matrix built out of data, and $\alpha > 0$ is the parameter quantifying the strength of the regularization.

- This can be formulated in ADMM form as follows:

$$\min_x \ \frac{1}{2}\|Ax - b\|_2^2 + \alpha\|z\|_1 \quad \text{subject to } x - z = 0. \tag{23}$$

LASSO regression

- The calculations done in the examples above show that the ADMM algorithm for this problem can be written as

$$\begin{aligned} x_{k+1} &= (A^{\mathrm{T}}A + \rho I)^{-1}(A^{\mathrm{T}}b + \rho z_k - \lambda_k) \\ z_{k+1} &= S_{\alpha/\rho}(x_{k+1} + \lambda_k/\rho) \\ \lambda_{k+1} &= \lambda_k + \rho(x_{k+1} - z_{k+1}) \end{aligned} \tag{24}$$

- Note that the matrix $A^{\mathrm{T}}A + \rho I$ is invertible as along as $\rho > 0$.

Consensus optimization

- As a motivation for the next topic, we consider the task of estimating the parameters θ of a statistical model (say, a time series model) based on p different large datasets $\mathcal{X}_1, \ldots, \mathcal{X}_p$.
- The log likelihood function of the dataset $\mathcal{X}_i$ is $f_i(\theta) = -\log \mathcal{L}(\theta|\mathcal{X}_i)$, and the aggregate log likelihood function is

$$f(\theta) = \sum_{i=1}^{p} f_i(\theta).$$

- The MLE estimate θ^* of θ is the minimizer of $f(\theta)$. For performance reasons, we would like to carry out the optimization in a distributed manner.
- This leads to the problem of *consensus optimization*.
- In the following, we revert to denote the parameters by x rather than θ.

Consensus optimization

- We consider the problem in which the objective functions has p terms (for example, $f_i(x)$ is the objective function of i-th block of training data):

$$\min \sum_{i=1}^{p} f_i(x). \tag{25}$$

- We could add a regularization term $r(x)$ to the objective function, such as Tikhonov regularization or LASSO penalty.
- Splitting the arguments, we can write this problem in ADMM form:

$$\min \sum_{i=1}^{p} f_i(x_i) \quad \text{subject to } x_i - z = 0, \tag{26}$$

where x_i are referred to as local variables, and z is the global variable.

- The augmented Lagrange function reads

$$L_\rho(x, z, \lambda) = \sum_{i=1}^{p} \left(f_i(x_i) + \lambda_i^{\mathrm{T}}(x_i - z) + \frac{1}{2}\rho \|x_i - z\|^2 \right). \tag{27}$$

Consensus optimization

- The ADMM iteration takes the form

$$
\begin{aligned}
x_{i,k+1} &= \arg\min_{x_i} \left(f_i(x_i) + \lambda_{i,k}^{\mathrm{T}}(x_i - z_k) + \frac{1}{2}\rho \|x_i - z_k\|^2 \right) \\
z_{k+1} &= \frac{1}{p} \sum_{i=1}^{p} \left(x_{i,k+1} + \lambda_{i,k}/\rho \right) \\
\lambda_{i,k+1} &= \lambda_{i,k} + \rho(x_{i,k+1} - z_{k+1})
\end{aligned}
\tag{28}
$$

- First order condition requires that $\nabla_z L_\rho(x, z, \lambda) = 0$, or

$$
\sum_{i=1}^{p} \lambda_i = 0. \tag{29}
$$

Consensus optimization

- The algorithm can thus be written in the following simpler form:

$$\begin{aligned} x_{i,k+1} &= \arg\min_{x_i} \left(f_i(x_i) + \lambda_{i,k}^{\mathrm{T}}(x_i - \overline{x}_k) + \frac{1}{2}\rho \|x_i - \overline{x}_k\|^2 \right) \\ \lambda_{i,k+1} &= \lambda_{i,k} + \rho(x_{i,k+1} - \overline{x}_{k+1}), \end{aligned} \tag{30}$$

  where

$$\overline{x}_k = \frac{1}{p}\sum_{i=1}^{p} x_{i,k}. \tag{31}$$

- This can be summarized as the following fully parallelizable procedure:
  - (i) collect $x_{i,k}$ from the processors and calculate the average $\overline{x}_k$,
  - (ii) send $\overline{x}_k$ to the processors,
  - (iii) update $\lambda_{i,k}$ in each of the processors,
  - (iv) update $x_{i,k}$ in each of the processors.

Consensus optimization

- Going back to the motivating example, we could interpret this procedure statistically as follows.
- $x_{i,k+1}$ is the MAP estimate of x_i under the prior probability distribution $N(\overline{x}_k + \lambda_{i,k}/\rho,\ \rho I)$.
- The mean of the prior distribution is the previous iteration's consensus adjusted by processor's i own view (the shadow price of disagreement with consensus).

Exchange problem

- We consider the following *exchange problem*:

$$\min \sum_{i=1}^{p} f_i(x_i) \quad \text{subject to} \quad \sum_{i=1}^{p} x_i = 0, \tag{32}$$

  where $x_i \in \mathbb{R}^n$, for each $i = 1, \ldots, p$.
- The exchange problem can be interpreted as p agents (traders) exchanging n commodities: $(x_i)_j > 0$ means that agent i purchases commodity j, while $(x_i)_j < 0$ means that agent i sells commodity j.
- The cost for agent i is $f_i(x_i)$ and the objective is to minimize the total cost.
- We formulate the problem in ADMM form:

$$\min \sum_{i=1}^{p} f_i(x_i) + \chi_C(z) \quad \text{subject to } x_i = z_i,\ i = 1, \ldots, p, \tag{33}$$

  where $C = \left\{ z \in \mathbb{R}^n : \sum_{i=1}^{p} z_i = 0 \right\}$.

Exchange problem

- The augmented Lagrange function reads

$$L_\rho(x, z, \lambda) = \sum_{i=1}^{p} f_i(x_i) + \chi_C(z) + \lambda^{\mathrm{T}}(x - z) + \frac{1}{2}\rho \sum_{i=1}^{p} \|x_i - z_i\|^2, \tag{34}$$

  where λ is an $n \times p$ matrix of Lagrange multipliers.
- The dual Lagrange function is

$$\begin{aligned} q(\lambda) &= \sum_{i=1}^{p} \inf_{x_i} \left(f(x_i) + \lambda_i^{\mathrm{T}} x_i\right) + \inf_{z} \left(\chi_C(z) - \lambda^{\mathrm{T}} z\right) \\ &= -\sum_{i=1}^{p} \sup_{x_i} \left(-\lambda_i^{\mathrm{T}} x_i - f(x_i)\right) + \inf_{z} \left(\chi_C(z) - \lambda^{\mathrm{T}} z\right), \end{aligned}$$

  i.e.

$$q(\lambda) = \begin{cases} -\sum_{i=1}^{p} f_i^*(-\lambda_i), & \text{if } \lambda_1 = \ldots = \lambda_p, \\ -\infty, & \text{otherwise.} \end{cases} \tag{35}$$

- This means that the exchange problem is the dual of the consensus problem!

Exchange problem

- The ADMM algorithm for the exchange problem reads:

$$\begin{aligned} x_{i,k+1} &= \arg\min_{x_i} \left(f_i(x_i) + \lambda_k^{\mathrm{T}} x_i + \frac{1}{2}\rho \|x_i - (x_{i,k} - \overline{x}_k)\|^2 \right) \\ \lambda_{k+1} &= \lambda_k + \rho \overline{x}_{k+1}, \end{aligned} \tag{36}$$

- The scaled version of ADMM is:

$$\begin{aligned} x_{i,k+1} &= \arg\min_{x_i} \left(f_i(x_i) + \frac{1}{2}\rho \|x_i - (x_{i,k} - \overline{x}_k - \zeta_k)\|^2 \right) \\ \zeta_{k+1} &= \zeta_k + \overline{x}_{k+1}, \end{aligned} \tag{37}$$

- Note that the x-update, for $i = 1, \ldots, p$, can be carried out in parallel. The ζ-update requires collecting the $x_{i,k+1}$ from the processors, calculating the average, and sending $\zeta_{k+1} + \overline{x}_{k+1}$ back to the processors.

Exchange problem

- This algorithm can be interpreted in economic terms as follows.
- As discussed earlier, the dual variables λ_i have the interpretation of (shadow) price vectors for agent i.
- The dual variable λ_k converges to an optimal set of clearing prices for the exchange. The proximal term in the x-update is a penalty for x_{k+1} deviating from x_k, projected onto the feasible set.
- The algorithm can be viewed as price adjustment process in general equilibrium: the market increases or decreases the price of each commodity, depending on whether there is an excess demand or excess supply of the commodity.
- Each agent adjusts his consumption x_i to minimize his individual cost $f_i(x_i)$ adjusted by the cost $\lambda_i^T x_i$. The central collector works toward equilibrium by adjusting the prices λ up or down depending on whether the commodity is oversupplied or undersupplied.
- As λ_k converges to an optimal price vector λ, the effect of the proximal regularization term vanishes. The proximal regularization term can be interpreted as each agent's commitment to help clear the market.

ADMM

- More details on ADMM can be found in the review paper [1] and references cited therein.

Marginal risk contributions

- Consider a long only portfolio of n assets, and let C denote the covariance matrix of their returns.
- The asset allocation is given by a vector $w \in \mathbb{R}^n$, with $w_i > 0$, for each $i = 1, \ldots, n$, and $\sum_{i=1}^n w_i = 1$.
- The standard deviation of the portfolio returns is given by $\sigma(w) = \sqrt{w^{\mathrm{T}} C w}$. As discussed earlier, this standard deviation is often used as a risk metric.
- *Risk parity* allocation attempts to allocate the assets in such a way that each of them contributes the same to the portfolio standard deviation $\sigma(w)$.

Homogeneous functions and Euler's identity

- A function $f(x)$, $x \in \mathbb{R}^n$, is called homogeneous of degree η if for $\alpha > 0$,

$$f(\alpha x) = \alpha^{\eta} f(x). \tag{38}$$

- For example, the standard deviation of portfolio returns is homogeneous of degree 1, as $\sqrt{(\alpha w)^{\mathrm{T}} C(\alpha w)} = \alpha\sqrt{w^{\mathrm{T}} C w}$.
- Taking the derivative of (38) with respect to α and setting $\alpha = 1$ we obtain the relation

$$\nabla f(x)^{\mathrm{T}} x = \eta f(x). \tag{39}$$

  This relation is called Euler's identity.
- In the case of the standard deviation function, (39) reads

$$\nabla \sigma(w)^{\mathrm{T}} w = \sigma(w). \tag{40}$$

Marginal risk contributions

- We can rewrite this identity as

$$\sum_{i=1}^{n} MRC(w)_i = 1, \tag{41}$$

where

$$MRC(w) = \frac{1}{\sigma(w)} \begin{pmatrix} w_1 \frac{\partial \sigma(w)}{\partial w_1} \\ \vdots \\ w_n \frac{\partial \sigma(w)}{\partial w_n} \end{pmatrix}$$
$$= \frac{1}{w^{\mathrm{T}} C w} \begin{pmatrix} w_1 (Cw)_1 \\ \vdots \\ w_n (Cw)_n \end{pmatrix}$$

is the vector of *marginal risk contributions*.

Risk budgeting

- Given a vector $b \in \mathbb{R}^n$ with $b_i > 0$, for $i = 1, \ldots, n$, and $\sum_{i=1}^n b_i = 1$, the problem of *risk budgeting* consists in finding a weight vector w so that

$$MRC(w) = b. \tag{42}$$

- In other words, each asset is allocated according to the way it contributes to the overall risk (as measured by the standard deviation) of the portfolio.
- Risk budgeting with all components of b equal, i.e. $b_i = 1/n$, corresponds to risk parity.
- Equation (42) can be stated explicitly as

$$Cw = \sigma(w)^2 \, b \circ w^{-1}, \tag{43}$$

where $\circ$ denotes the pointwise product of vectors, and w^{-1} is the vector of inverses of the components of w.

Risk budgeting

- For convenience, we rescale the variable w by the factor $\sigma(w)$, i.e. we define $x = w/\sigma(w)$.
- In terms of this variable, (43) takes a simpler form:

$$Cx = b \circ x^{-1}. \tag{44}$$

- One can show that this equation has a unique solution with positive components.
- It is important to keep in mind that, as a result of the rescaling, the solution to (44) is not the allocation vector, but it is only proportional to the allocation vector.
- After solving (44), we have to normalize the solution, so that the sum of its components is 1.

Risk budgeting

- An efficient way of solving (44) numerically, due to F. Spinu [5], relies on the observation that this equation arises as the first order condition for the following convex optimization problem:

$$\min f(x) = \frac{1}{2} x^{\mathrm{T}} C x - b^{\mathrm{T}} \log(x), \tag{45}$$

where $\log(x)$ is the vector of logarithms of the component of the vector x.

- Indeed,

$$\nabla f(x) = Cx - b \circ x^{-1}, \tag{46}$$

and

$$\nabla^2 f(x) = C + \mathrm{diag}(b \circ x^{-2}), \tag{47}$$

where $\mathrm{diag}(v)$ denotes a diagonal matrix with the vector v on its diagonal.

Risk budgeting

- This allows us to formulate a Newton iteration scheme for solving equation (44).
- In fact, Spinu's detailed analysis, using Nesterov's concept of a self-concordant function, leads to the following refined version of Newton's method.
- In this version, the size of a Netwon step is optimized for performance.
- Set $\lambda_* = 0.95\,\frac{3-\sqrt{2}}{2}$, choose the stopping threshold ε and maximum number of iterations *max_iter*.

Risk budgeting

- Choose x_0, $\lambda_0 = 1$, $k = 0$, and iterate:

while $\lambda_k > \varepsilon$ and $k < max\_iter$

$$u_k = Cx_k - b \circ x_k^{-1}$$

$$H_k = C + \mathrm{diag}(b \circ x_k^{-2})$$

$$h_k = -H_k^{-1} u_k$$

$$\delta_k = \| h_k \circ x_k^{-1} \|_\infty$$

$$\lambda_k = \sqrt{-u_k^{\mathrm{T}} h_k}$$

if $\lambda_k > \lambda_*$

$$x_{k+1} = x_k + \frac{1}{1+\delta_k} h_k$$

else

$$x_{k+1} = x_k + h_k$$

- Once optimal x^* is found, set $w = x^* / \sum_{i=1}^n x_i^*$.

Risk budgeting

- While Spinu's algorithm has excellent performance characteristics, it is limited to portfolios with moderate numbers of positions and without additional constraints.
- Flexible optimization algorithms based on the ADMM methodology for portfolios with constraints on positions are discussed extensively in the literature, see in particular the recent papers [3] and [4].

References


[1] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J.: Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers, *Foundations and Trends in Machine Learning*, **3**, 1 – 122 (2010).


[2] Parikh, N., and Boyd, S.: Proximal Algorithms, *Foundations and Trends in Optimization*, **1**, 123 – 231 (2013).


[3] Perrin, S., and Roncalli, T.: Machine Learning Optimization Algorithms & Portfolio Allocation, preprint (2019).


[4] Richard, J.-C., and Roncalli, T.: Constrained Risk Budgeting Portfolios Theory, Algorithms, Applications & Puzzles, preprint (2019).


[5] Spinu, F.: An Algorithm for Computing Risk Parity Weights, preprint (2013).