projet c++

January 25, 2023

Pour mieux "visualiser" ce qu'il nous faut dans le projet C++ on peut essayer de traiter un cas particulier de notre probleme d'optimisation de portefeuille. On dispose d'un portefeuile $(x_1, \ldots, x_n)$. On peut considerer que l'agent veut juste optimiser son portefeuille à l'instant 0, et donc on peut se passer de x_{n+1} (à la difference du papier qu'on a lu).

L'agent a aussi a sa disposition une vecteur d'estimation des retours r, une matrice de variance-covariance (mesure du risque) Σ et et un paramètre d'aversion au risque γ. Une premiere contrainte sur le portefeuille est que $\sum x_i = 1$, i.e. $1^T x = 1$, qui est un cas particulier de notre portfolio constraint. On peut egalement supposer avoir des contraintes comme par exemple $x_i \geq \alpha$ avec $\alpha \in [0, 1]$ (on detient au moins α % de notre portefeuille sous la forme de l'actif i, pour differents i) ou encore des contraintes d'égalité, comme $x_j = \beta$ pour certains j.

Le programme à résoudre est alors : $\min_x \ \gamma x^T \Sigma x - r^T x$
s.c. $1^T x = 1$
$x_i \geq \alpha$,
$x_j = \beta$

J'ai essayé d'utiliser la librairie quadprogpp pour résoudre un tel pb d'optimisation.

Quadprogpp peut resoudre ce genre de problemes, mais uniquement dans le cas particulier ou Σ est definie positive.

Au niveau cette librairie C++, n est la dimension de vecteur portefeuille, m le nombre de contraintes d'égalité et p le nombre de contraintes d'inégalité.

Avec $\gamma = 0.5$, j'ai pris $n = 2$, $r = (-0.3, -0.6)^T$, et la matrice ((0.1, 0.2), (0.2, 0.8)) qui est bien symetrique definie positive.

Il faut definir les contraintes de la façon suivante :

$(CE)^T x + ce_0 = 0$ où CE est la matrice des contraintes d'égalités

$(CI)^T x + ci_0 >= 0$ où CI est la matrice des contraintes d'inégalités

Dans le code que j'ai mis, j'ai testé les contraintes $x_1 + x_2 = 1$, $x_1 = 0.5$ et pas de contraintes d'inegalites (CI est la matrice nulle)

Pour cette configuration j'ai donc pris CE = ((1.0, 2.0), (1.0, 0.0)) (la premiere colonne donne bien la condition $x_1 + x_2 = 1$, la deuxieme colonne donne la contrainte $2x_1 + 0x_0 = 1$. On prend donc $ce_0 = (-1, -1)$

En sortie ce programme donne bien l'allocation qu'on s'attendait par le calcul : $x_1 = x_2 = 0.5$