

Optimisation de portefeuilles

February 6, 2023

1 Formalisation du problème

On dispose d'un portefeuille (x_1, \dots, x_n) . L'agent souhaite optimiser son portefeuille en choisissant les actifs x_1, \dots, x_n qui lui permettront de minimiser son exposition au risque, tout en respectant des contraintes de type linéaire.

On suppose que l'agent a à sa disposition un vecteur d'estimation des retours r , une matrice de variance-covariance Σ . On choisit pour ce problème un paramètre d'aversion au risque $\gamma = \frac{1}{2}$. Une première contrainte sur le portefeuille est que $\sum x_i = 1$, i.e. $1^T x = 1$. On peut également supposer avoir des contraintes comme par exemple $x_i \geq \alpha$ avec $\alpha \in [0, 1]$ (on détient au moins α % de notre portefeuille sous la forme de l'actif i , pour différents i) ou encore des contraintes d'égalité, comme $x_j = \beta$ pour certains j . Sous ces hypothèses, et à condition que Σ soit une matrice définie-positive, on se retrouve à résoudre le programme :

$$\min_x \quad \gamma x^T \Sigma x - r^T x$$

sous les contraintes :

$$1^T x = 1$$

$$x_i \geq \alpha, \text{ pour certains } i \in [1, n]$$

$$x_j = \beta, \text{ pour certains } j \in [1, n]$$

Il s'agit d'un programme d'optimisation convexe (notamment, d'optimisation quadratique), pour lequel il existe des solveurs très optimisés. Ce problème est seulement un cas particulier de

2 Résolution en C++

Nous avons fait le choix d'utiliser la librairie QuadProg, écrite par Luca di Gaspero et disponible sous licence MIT. Nous avons aussi utilisé la librairie Array; ces deux librairies nous ont permis de ne pas tout réécrire de zéro, notamment tout ce qui concerne les opérations matricielles et des algorithmes, notamment celui de décomposition de Cholesky, qui est utilisé pour résoudre ce programme d'optimisation.

Pour récupérer les données fournies par l'utilisateur, nous avons codé une classe CSV, qui permet également de mettre ces données dans un format utilisable par le fichier main.cpp

L'utilisateur doit fournir, dans le répertoire /data, la matrice de variance-covariance, sous le nom matrix.csv, le fichier des retours, sous le nom returns.csv, ainsi que les matrices et les vecteurs des contraintes d'égalité et d'inégalité. Pour le bon fonctionnement du programme, il faut que les vecteurs contiennent une deuxième ligne vide.

3 Exemple d'utilisation

Notons r le vecteur des retours, Σ la matrice de variance-covariance, CE la matrice des contraintes d'égalités, (CI) la matrice des contraintes d'inégalités, $ce0$ le vecteur des contraintes d'égalités et $ci0$ le vecteur des contraintes d'inégalités. Pour le bon fonctionnement du programme, il faut définir les contraintes de la façon suivante :

$$(CE)^T x + ce0 = 0$$

$$(CI)^T x + ci0 \geq 0$$

Dans le répertoire /data, il existe déjà des fichiers permettant de tester, dans un cas particulier, le bon fonctionnement du programme d'optimisation.

Nous avons pris voulu optimiser un portefeuille de la forme (x_1, x_2) , avec pour vecteur des retours $r^T = (0.3, 0, 6)$, $\Sigma = \begin{pmatrix} 0.1 & 0.2 \\ 0.2 & 0.8 \end{pmatrix}$, qui est bien définie positive.

Pour une solution facile à vérifier à la main, nous avons voulu tester les contraintes $x_1 + x_2 = 1$, $2x_1 = 1$, d'où $= \begin{pmatrix} 1.0 & 2.0 \\ 1.0 & 0.0 \end{pmatrix}$, et $(ce0)^T = (-1.0, -1.0)$

Nous avons enfin pris $= \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$, ainsi que $(ci0)^T = (0.0, 0.0)$, ce qui force le programme à générer des solutions (si elles existent), telles que $x_1 \geq 0$ et $x_2 \geq 0$

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