Université Paris Descartes
UFR de Mathématiques et Informatique
45, rue des Saints-Pères 75270 Paris cedex 06

UNIVERSITÉ PARIS DESCARTES

MASTER 1

Programmation

TP Transport Optimal - Scilab

Introduction

Dans ce TP, on s'intéresse au problème du transport optimal ou du mariage entre deux nuages de points du plan $\mathbb{R}^2$. Soit n un entier. On considère $2n$ points $X_1, \dots, X_n, Y_1, \dots, Y_n$ du plan. Chaque point X_k a des coordonnées que l'on note (x_k^1, x_k^2) dans le plan, et chaque point Y_k a des coordonnées que l'on note (y_k^1, y_k^2). On cherche à résoudre le problème de transport optimal (ou de mariage optimal) entre les points $(X_j)_{1\leq j\leq n}$ et les points $(Y_j)_{1\leq j\leq n}$, c'est-à-dire que l'on cherche la permutation σ de l'ensemble $\{1, \dots, n\}$ qui permet d'atteindre le minimum de la quantité

$$\sum_{i=1}^{n} \|X_i - Y_{\sigma(i)}\|_2^2. \tag{1}$$

Ce problème est très classique en logistique ou en économie : on peut par exemple imaginer que des marchandises sont stockées dans des fabriques situées aux points X_i et que chaque Y_j représente un client. On souhaite alors répartir le transport des marchandises des fabriques vers les clients de manière à ce que le coût total de transport soit le plus petit possible. Dans ce qui suit, on notera

$$X = \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix} \quad \text{et} \quad Y = \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix}$$

les matrices à n lignes et 2 colonnes contenant l'ensemble des points X_k et Y_k. X et Y sont appelés des nuages de points du plan.

Commencez par créer deux nuages de points X^0 et Y^0 sous la forme de matrices $n \times 2$ et par les visualiser, par exemple en tirant les points aléatoirement avec une loi uniforme sur le carré $[0,1] \times [0,1]$.

```
n= 10;
X0 = rand(n,2,'uniform');              // création du premier nuage
Y0 = rand(n,2,'uniform');              // deuxième nuage
plot2d(X0(:,1),X0(:,2),style=-9);      // affichage du premier nuage (ronds)
plot2d(Y0(:,1),Y0(:,2),style=-1);      // affichage du deuxième nuage (croix)
```

Exercice 1 - Transport en dimension 1.

On suppose dans un premier temps que les deux nuages de points X et Y sont dans $\mathbb{R}$ et pas dans $\mathbb{R}^2$. On pourra donc écrire dans cet exercice X et Y comme des vecteurs colonnes de longueur n. Les points de X ne sont pas forcément ordonnés, on note donc $\sigma_X : \{1, \dots, n\} \longrightarrow \{1, \dots, n\}$ la permutation qui ordonne les points de X de manière croissante :

$$X_{\sigma_X(1)} \leq X_{\sigma_X(2)} \leq \cdots \leq X_{\sigma_X(n)}.$$

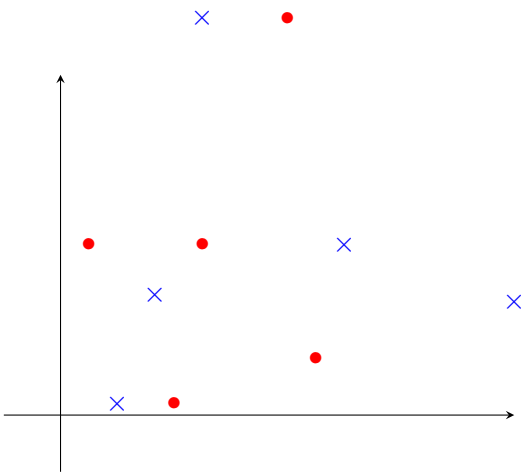

FIGURE 1 – Deux nuages de points de $\mathbb{R}^2$. Le nuage X est représenté par des disques et le nuage Y par des croix. On souhaite marier les points de X et les points de Y de manière à rendre la somme des distances au carré entre points mariés la plus petite possible.

De même, on note $\sigma_Y : \{1, \dots, n\} \longrightarrow \{1, \dots, n\}$ la permutation qui ordonne les points de Y de manière croissante.
Dans ce cas particulier (nuages dans $\mathbb{R}$), la permutation σ qui permet de rendre la quantité (1) la plus petite possible est celle qui envoie le k^{ieme} plus petit point de X sur le k^{ieme} plus petit point de Y. Cette permutation peut s'écrire

$$\sigma = \sigma_Y \circ \sigma_X^{-1}.$$

1. Expliquez pourquoi le minimum de la quantité (1) est alors égal à

$$\sum_{i=1}^{n} \|X_{\sigma_X(i)} - Y_{\sigma_Y(i)}\|_2^2. \tag{2}$$

2. Ecrire une fonction `transport1D`(X, Y) qui prend en entrée les vecteurs X et Y et renvoie les vecteurs de permutations

$$s_X = \begin{pmatrix} \sigma_X(1) \\ \vdots \\ \sigma_X(n) \end{pmatrix} \quad \text{et} \quad s_Y = \begin{pmatrix} \sigma_Y(1) \\ \vdots \\ \sigma_Y(n) \end{pmatrix}$$

et la quantité (2). On pourra utiliser la fonction **gsort** de Scilab.

Exercice 2 - Transport par projections en deux dimensions.
On souhaite maintenant résoudre le problème pour deux nuages de points X et Y dans le plan. Dans ce qui suit, on rappelle que chaque nuage de points Z de taille n pourra s'écrire sous la forme d'une matrice à n lignes et 2 colonnes :

$$Z = \begin{pmatrix} Z_1 \\ \vdots \\ Z_k \\ \vdots \\ Z_n \end{pmatrix} = \begin{pmatrix} z_1^1 & z_1^2 \\ \vdots & \vdots \\ z_k^1 & z_k^2 \\ \vdots & \vdots \\ z_n^1 & z_n^2 \end{pmatrix},$$

où (z_k^1, z_k^2) sont les coordonnées du point Z_k dans le plan.

Remarque : Trouver la permutation σ qui rend la quantité (1) *la plus petite possible est un problème difficile en pratique, on propose donc de le résoudre de manière approchée. Pour cela, on propose un algorithme itératif qui va progressivement déplacer les points de X vers les points de Y en résolvant plusieurs problèmes unidimensionnels successifs.*

L'algorithme approché que l'on propose d'utiliser est itératif et consiste à répéter M fois des étapes de projection et de déplacement des points du nuage X. On détaille ces étapes dans ce qui suit.

Projection aléatoire. A chaque étape, on tire aléatoirement une direction θ dans $]-\pi/2, \pi/2]$ avec une loi uniforme et on note u_θ le vecteur $u_\theta = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}$. On projette ensuite les points de X et les points de Y orthogonalement sur la droite du plan passant par 0 et de vecteur directeur u_θ (voir la Figure 2). On obtient ainsi deux nouveaux nuages de points U et V, qui peuvent être vus comme des nuages de points sur $\mathbb{R}$.

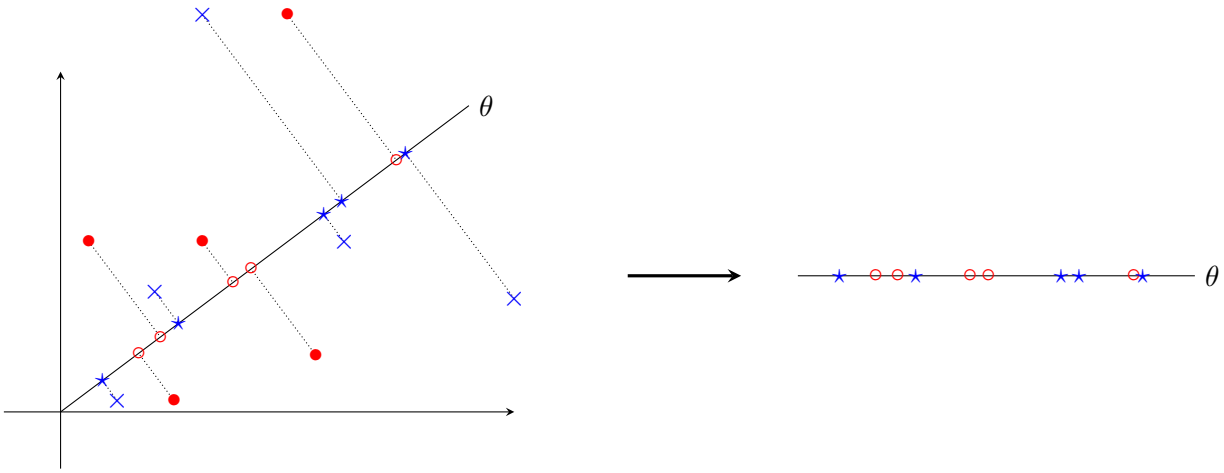

FIGURE 2 – Projections des nuages X et Y selon la direction θ. Après projection, on obtient un problème de mariage entre deux nouveaux nuages de points U et V en une dimension.

Déplacement des points de X. Une fois les vecteurs U et V calculés, on détermine les vecteurs de permutations s_X et s_Y qui ordonnent les projections U et V, par exemple en leur appliquant la fonction `transport1D` de l'exercice 1.
Ensuite, on fait évoluer le nuage de points X dans la direction de u_θ, proportionnellement au déplacement entre U et V. Plus précisément, on met à jour les positions de X par la formule

$$\forall 1 \leq i \leq n, \quad X_{s_X(i)} = X_{s_X(i)} + \varepsilon \left(V_{s_Y(i)} - U_{s_X(i)}\right) u_\theta,$$

où $\varepsilon > 0$ est le pas de l'algorithme et est fixé par l'utilisateur.

Algorithme de Transport optimal par projections. On répète ces étapes de projection et de déplacement M fois, en tirant à chaque fois une nouvelle direction aléatoire θ pour la projection. Si le pas de l'algorithme ε est bien choisi et si le nombre d'itérations M est suffisamment grand, les points du nuage X doivent coïncider à la fin de l'algorithme avec ceux du nuage Y.

Au final, le pseudo-code de l'algorithme complet de transport approché peut s'écrire de la manière suivante :

Pseudo-code

Entrée : nuages X^0 et Y^0, pas ε, nombre d'itérations M.
Sortie : nuage X déplacé et coût total de transport C.
Initialisez $X = X^0$ et $Y = Y^0$.
for $k = 1$ **to** M **do**
  Tirer θ aléatoirement dans $]-\pi/2, \pi/2]$ (loi uniforme).
  $U =$ `projection`(X, θ).
  $V =$ `projection`(Y, θ).
  $[s_X, s_Y] =$ `transport1D`(U, V)
  $\forall 1 \leq i \leq n$, $X_{s_X(i)} = X_{s_X(i)} + \varepsilon \left(V_{s_Y(i)} - U_{s_X(i)}\right) u_\theta$
Coût total $C = \sum_{i=1}^{n} \|X^0_{s_X(i)} - Y^0_{s_Y(i)}\|_2^2$.

Fin du pseudo-code

1. Ecrire une fonction $[U] =$ `projection`(X, θ) qui prend en entrée une direction θ et un nuage de points X du plan, et qui calcule le vecteur projeté $U = X.u_\theta$ (le . est une multiplication de matrices, X étant codé sous la forme d'une matrice de taille $n \times 2$ et le vecteur u_θ sous la forme d'un vecteur 2×1).
2. Ecrire une fonction $[X, C] =$ `transport`$(X^0, Y^0, M, \varepsilon)$ qui fait évoluer le nuage de points X^0 vers le nuage de points Y^0 et calcule le coût de transport entre eux grâce à l'algorithme précédent.
3. Afin d'accélérer l'algorithme, on décide de stopper les itérations de la boucle FOR lorsque le maximum sur k des distances $\|X_{s_X(k)} - Y_{s_Y(k)}\|_2$ est inférieur à un seuil δ fixé par l'utilisateur. Modifier la fonction Scilab précédente pour prendre en compte ce raffinement et commentez.
4. Tester cet algorithme sur des nuages de points X^0 et Y^0 crées à votre convenance (par exemple ceux crées au début). Prendre $\varepsilon = 10^{-2}$ et $M = 10^5$. Afficher le résultat du mariage trouvé en utilisant la commande `xpoly`. Par exemple, pour afficher un segment entre le point X_i et le point Z_i, on peut écrire

   ```
   xpoly([X(i,1),Z(i,1)],[X(i,2),Z(i,2)],'lines');
   ```

5. Lancez l'algorithme plusieurs fois sur les mêmes données et observer les résultats obtenus. Les directions θ de projection étant choisies aléatoirement au cours des étapes, le résultat de l'algorithme (à la fois le coût de transport C et la manière dont les points sont mariés) peut changer d'une expérience à l'autre. L'algorithme est-il stable en pratique ? Commenter la stabilité lorsque l'on change le nombre de points n des nuages, les valeurs du pas ε et le nombre d'itérations.