SY09 Printemps 2020
TP 03 — Représentation euclidienne des données

1 Représentation euclidienne

1.1 Représentation des données

Dans cette partie, on s'intéresse à un jeu de données contenant les notes de $n = 9$ individus pour $p = 5$ matières : mathématiques, sciences « naturelles » (physique-chimie), français, latin, « arts » (dessin-musique).

On souhaite étudier ces données et les représenter de manière à caractériser les individus en fonction de leur niveau scolaire. Plus particulièrement, on cherchera à déterminer une *base de représentation* permettant de visualiser « au mieux » les données.

(1) Charger le jeu de données présent dans le fichier `data/notes.txt`. Visualiser la dispersion des notes. On pourra utiliser la fonction fournie `add_labels` pour ajouter les étiquettes.

Analyse et représentation succinctes

(2) Faire une brève analyse des données ; en particulier :

1. comment sont réparties les notes, dans chaque matière ?
2. Peut-on rapprocher certaines matières les unes des autres ?

(3) On cherche à identifier des groupes d'élèves en fonction des résultats scolaires.

1. Représenter les élèves en fonction de leurs résultats dans les matières scientifiques ; quels sont les groupes d'individus qui semblent se détacher ?
2. Faire de même avec les matières littéraires, puis avec les arts.

(4) Représenter les élèves par deux informations : leur moyenne dans les matières scientifiques, et leur moyenne dans les matières littéraires. Interpréter les résultats obtenus.

Projection et qualité de représentation

(5) On considère la matrice suivante :

$$A_1 = \begin{pmatrix} 1/2 & 0 & 1/2 & 0 & 0 \\ 1/2 & 0 & -1/2 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & -1/2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

1. Définit-elle une base ?
2. Comment exprimer les notes du tableau X dans la nouvelle base A_1, et comment revenir dans la base canonique de $\mathbb{R}^5$?

3. Comment peut-on interpréter les coordonnées des individus exprimées dans cette nouvelle base ? Que permet de visualiser la représentation selon les composantes (X^1, X^2), ou (X^1, X^3), ou encore (X^2, X^4) ?

(6) On considère à présent la matrice B_1 suivante :

$$B_1 = \begin{pmatrix} \sqrt{2}/2 & 0 & \sqrt{2}/2 & 0 & 0 \\ \sqrt{2}/2 & 0 & -\sqrt{2}/2 & 0 & 0 \\ 0 & \sqrt{2}/2 & 0 & \sqrt{2}/2 & 0 \\ 0 & \sqrt{2}/2 & 0 & -\sqrt{2}/2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

1. Définit-elle une base ?
2. Comment peut-on interpréter les coordonnées des individus exprimés dans cette nouvelle base ? Que permet de visualiser la représentation selon les composantes (X^1, X^2), ou (X^1, X^3), ou encore (X^2, X^4) ?

(7) On considère à présent la matrice B_2 suivante :

$$B_2 = \begin{pmatrix} \sqrt{2}/2 & 0 & \sqrt{2}/2 & 0 & 0 \\ 0 & \sqrt{2}/2 & 0 & \sqrt{2}/2 & 0 \\ \sqrt{2}/2 & 0 & -\sqrt{2}/2 & 0 & 0 \\ 0 & \sqrt{2}/2 & 0 & -\sqrt{2}/2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

1. Définit-elle une base ?
2. Comment peut-on interpréter les coordonnées des individus exprimés dans cette nouvelle base ? Que permet de visualiser la représentation selon les composantes (X^1, X^2), ou (X^1, X^3), ou encore (X^2, X^4) ?

1.2 Choix d'une représentation

On définit la qualité de la représentation selon un axe comme étant la quantité d'inertie expliquée par cet axe.

(8) Quelle est la quantité d'inertie du nuage de points expliquée par chacun des axes, si l'on considère la base canonique de $\mathbb{R}^5$? Quelle est la quantité d'inertie totale du nuage de points ? On pourra utiliser `np.cov` avec les bons arguments.

(9) Quelle est la quantité d'inertie expliquée par les axes si l'on considère la base A_1 ? Pour inverser une matrice, on pourra utiliser le sous-module `linalg` avec l'instruction

```
import numpy.linalg as linalg
```

(10) Qu'en est-il pour la base B_1, pour la base B_2 ?

(11) On cherche à représenter le nuage de points dans un plan, au prix d'une perte d'information. Quels axes choisirait-on, parmi ceux définis par la base canonique, par B_1, ou par B_2 ? Pourquoi ? Interpréter.

(12) On considère à présent la matrice B_3 suivante :

$$B_3 = \begin{pmatrix} 1/2 & 1/2 & 1/2 & 1/2 & 0 \\ 1/2 & 1/2 & -1/2 & -1/2 & 0 \\ 1/2 & -1/2 & -1/2 & 1/2 & 0 \\ 1/2 & -1/2 & 1/2 & -1/2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

La matrice B_3 est elle aussi orthogonale. Est-elle meilleure que B_1 en terme d'inertie expliquée ?

2 Inertie cumulée enveloppante

L'idée de cette partie est de caractériser une bonne base orthogonale en terme d'inertie cumulée. Pour ce faire, on va avoir besoin de générer des bases orthogonales aléatoirement.

On admettra que le code suivant fournit une matrice orthogonale générée aléatoirement.

```
import numpy.linalg as linalg
U, _ = linalg.qr(np.random.randn(5, 5))
```

(13) Créer une fonction qui renvoie les inerties cumulées associées au changement de base d'une matrice orthogonale choisie aléatoirement.

(14) Afin de visualiser plusieurs courbes d'inertie cumulée pour différentes bases choisies aléatoirement, on veut former le jeu de données comportant les trois descripteurs suivants :
- l'identifiant de la base considérée,
- l'axe jusqu'auquel effectuer le cumul des inerties,
- l'inertie cumulée correspondante.

(15) Visualiser à l'aide de la fonction `sns.pointplot` les courbes pour chaque base considérée avec en abscisse le nombre d'axes et en ordonnée l'inertie cumulée. Quelle devrait être la propriété d'une base optimale en terme d'inertie expliquée ? Semble-t-il y en avoir une parmi les bases générées aléatoirement ?

(16) Quel est la particularité de la base orthogonale définie par l'instruction suivante :

```
Bx = array([[0.515, -0.567, 0.051, -0.289, -0.573],
            [0.507, -0.372, 0.014, 0.553, 0.546],
            [0.492, 0.65, -0.108, 0.394, -0.41],
            [0.485, 0.323, -0.023, -0.674, 0.453],
            [0.031, 0.113, 0.992, 0.034, -0.013]])
```

(17) Que donne la matrice orthogonale suivante

$$B_y = B_x \begin{pmatrix} A & 0 & 0 \\ 0 & A & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

avec A une matrice orthogonale de taille 2 quelconque. Pourquoi ?

3 Inertie par rapport à un axe

On se propose de vérifier expérimentalement les formules de Huygens. Pour cela, on utilise le code suivant

```
n, p = 100, 2
X = np.random.randn(n, p)

feat_metric = np.eye(p)
sample_metric = 1 / n * np.eye(n)
```

(18) Compléter la fonction `inertia_factory` et définir deux fonctions `inertia_point`, `inertia_axe` qui calculent des inerties par rapport à un point ou à une droite.

(19) Vérifier les fonctions avec le code suivant

```
X0 = np.array([[1, 0], [0, 1]])
n0, p0 = X0.shape
feat_metric0 = np.eye(p0)
sample_metric0 = 1 / n0 * np.eye(n0)
feat_inner0, inertia_point0, inertia_axe0 = inertia_factory(feat_metric0,
↪  sample_metric0)

import math
p0 = np.zeros(2)
v = np.array([0, 1])
p1 = np.array([-1, 0])
assert(math.isclose(inertia_point0(X0), .5))
assert(math.isclose(inertia_point0(X0, p0), 1.))
assert(math.isclose(inertia_axe0(X0, v), .5))
assert(math.isclose(inertia_axe0(X0, v, p1), 2.5))
```

(20) Calculer l'inertie de `X` par rapport à un point quelconque.

(21) Vérifier la première formule de Huygens en recalculant l'inertie d'une autre manière.

(22) Calculer l'inertie de `X` par rapport à un axe quelconque.

(23) Vérifier la formule de Huygens de l'inertie par rapport à une droite.