L'analyse en composantes principales

1 ACP : Exemple simple

Nous voulons étudier un tableau de notes.

1.1 Pré-étude des données

- Charger les données qui sont dans le tableau notes "notes.csv". Pour cela on utilise la commande :

```
X = np.transpose(pd.read_csv("notes.csv", A completer ...))
# On transpose car les individus doivent etre en ligne pour utiliser
    les fonctions de Sklearn.
```

  Il s'agit d'un tableau de notes obtenues par 9 élèves dans les matières suivantes : mathématiques, sciences, français, latin et dessin.

  Nous allons tout d'abord étudier graphiquement et statistiquement ces données notées par X. On va d'abord créer les vecteur des noms des individus et des noms des variables :

```
nomi = list(X.index) # noms de variables
nomv = list(X.columns) # noms des individus
```

Pour chaque graphique demandé, extraire de l'information sur les données et confirmer ces constatations visuelles par des mesures statistiques (par exemple écart-type) :

- Représenter graphiquement la matière "français". Pour cela, on peut utiliser la commande :

  `matplotlib.pyplot.hist`

- Représenter graphiquement la matière "latin".

- Représenter dans un plan le nuage de points caractérisés par les deux variables "mathématique" et "sciences". Si vous souhaitez ajouter les noms des individus vous pouvez utiliser les commandes suivantes :

```
for i in range(0,n):
        plt.scatter(Data[i,0],Data[i,1])
        plt.text(Data[i,0],Data[i,1],nomi[i])
```

- Représenter dans un plan le nuage de points caractérisés par les deux variables "mathématique" et "dessin".

1.2 Calcul de l'ACP

Soit X les données de dimension p rangées dans un tableau de taille $n \times p$. On peut effectuer l'ACP en Python en utilisant, par exemple, la commande PCA de `scikit-learn` :

```
from sklearn.decomposition import PCA

acp = PCA(n_components=p)
cc = acp.fit_transform(X) ## cc contient les projections
```

Quelques options et attributs de la classe `PCA` :

- Options:
  - `n_components` est le nombre de composantes à garder, si le paramètre n'est pas défini, toutes les composantes seront calculées; avec l'option n components='mle' le nombre de composantes à garder est calculé automatiquement
  - `whiten` est le flag de l'ACP sur la matrice de corrélation: whiten='TRUE'
- Attribus de la structure stockée dans acp
  - `components` correspond aux axes principales (vecteurs propres de la matrice de covariance)
  - `explained_variance` la quantité de variance (d'inertie) expliquée par chaque axe et donc la valeur propre associée à chaque axe;

Si vous souhaitez projeter un nouveau nuage de points dans les axes ainsi définis par l'ACP :

```
        data2proj = acp.transform(data2)components
```

1.3 Représentation

Les différentes étapes qui suivent doivent vous permettre de comprendre la structuration de l'ensemble d'apprentissage en classes. Ces classes vont correspondre à autant de "modes de fonctionnement". Pour chaque étape, analyser les différents graphiques et mesures afin de conclure sur l'organisation des données.

- Afficher l'évolution de l'inertie expliquée (variance) cumulée selon le nombre d'axes (`scree{plot}`). Choisir le nombre d'axes à conserver.
- Représenter les individus dans les plans $E_1 \cup E_2$ et $E_1 \cup E_3$ (représentation 1). Ici E_i correspond à la projection sur l'axe i.
- Calculer les corrélations entre les variables initiales ζ_k, $k = 1, \ldots, p$ et les composantes principales Y_j, $j = 1, \ldots, p$:

$$\mathrm{CORR}(\zeta_k, Y_j) = acp.components[j,k] \times \sqrt{\frac{\lambda_j}{\sigma_k^2}}$$

  où $\sigma_k^2 = \mathrm{Var}(\zeta_k)$ et λ_j est la variance (valeur propre) correspondant à la composante Y_j. Attention, la matrice dite de rotation `acp.components` est rangée par ligne (chaque ligne correspond au vecteur propre).
- étudier les valeurs de corrélation entre les matières initiales et les trois premiers axes factoriels. Commenter.
- Tracer la représentation simultanée des individus et des variables (`biplot`) dans les plans $E_1 \cup E_2$ et $E_1 \cup E_3$ (représentation 2). Pour cela, nous vous proposons la fonction `my_biplot` :

```
from affichage_acp import my_biplot

my_biplot(score=cc[:,0:2], coeff=np.transpose(acp.components_[0:2,:]),
    coeff_labels=nomv,... score_labels=nomi,nomx = "PC1",nomy = "PC2")
```

- Analyser les résultats.

2 Données réelles

Nous vous proposons de travailler sur des données réelles dans le cadre d'une application classique de l'ACP : l'analyse d'image multispectrale.

Une image multispectrale est une image dans laquelle les données correspondant à des mesures prises à différentes longueurs d'onde sur le spectre électromagnétique. Les longueurs d'onde peuvent correspondent à l'intervalle du spectre visible mais aussi de l'infra-rouge. L'imagerie multispectrale permet l'extraction d'information complémentaire à l'oeil humain qui se limite à la capture d'information avec ces trois récepteurs. A l'origine, l'imagerie multispectrale était utilisée pour l'observation de la terre que ce soit pour des applications militaires ou civil : en effet, la mesure multispectrale permet de différentier les différents composants d'un scène (végétaux, eaux, terre, bâtiment ...).

La difficulté de ce type d'imagerie est que en chaque position (pixel), l'information est caractérisée par un grand nombre de valeurs (dans notre exemple 200). Cela rend difficile alors la manipulation, l'interprétation et bien sûr la visualisation. L'ACP appliquée sur l'image devient alors un outil central pour traiter ces images.

Nous proposons de travailler sur un jeu de données d'exemple Indians Pines[1].

Cette scène a été recueillie par le capteur satellitaire AVIRIS au-dessus du site d'essai d'Indian Pines dans le nord-ouest de l'Indiana et se compose de 145×145 pixels et de 224 bandes de réflectance spectrale dans la gamme de longueurs d'onde 0,4-2,5 10^{-6} mètres. Cette scène est un sous-ensemble d'une scène plus grande. La scène `Indian Pines` contient deux tiers d'agriculture et un tiers de forêt ou autre végétation naturelle pérenne. On y trouve deux grandes autoroutes à deux voies, une voie ferrée, ainsi que des habitations à faible densité, d'autres structures bâties et des routes plus petites. La vérité terrain disponible est classée en seize classes qui ne sont pas toutes mutuellement exclusives. Nous avons également réduit le nombre de bandes à 200 en supprimant les bandes couvrant la région d'absorption de l'eau : [104-108], [150-163], 220. Les données d'Indian Pines sont disponibles sur le site MultiSpec de l'université de Pursue[2].

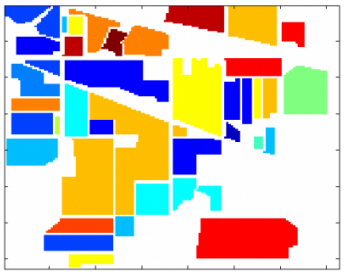
[1] Baumgardner, M. F., Biehl, L. L., Landgrebe, D. A. (2015). 220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3. Purdue University Research Repository. doi:10.4231/R7RX991C

[2] `https://engineering.purdue.edu/~biehl/MultiSpec/hyperspectral.html`

	Class	Samples
1	Alfalfa	46
2	Corn-notill	1428
3	Corn-mintill	830
4	Corn	237
5	Grass-pasture	483
6	Grass-trees	730
7	Grass-pasture-mowed	28
8	Hay-windrowed	478
9	Oats	20
10	Soybean-notill	972
11	Soybean-mintill	2455
12	Soybean-clean	593
13	Wheat	205
14	Woods	1265
15	Buildings-Grass-Trees-Drives	386
16	Stone-Steel-Towers	93

Classes de vérité du sol pour la scène d'Indian Pines et leur nombre d'échantillons respectifs.

Extrait d'une bande du jeu de données `Indian Pines`



Vérité terrain du jeu de données `Indian Pines`

- Charger les données avec les instructions :

```
from scipy.io import loadmat

imgtmp = loadmat("Indian_pines_corrected.mat")
img = np.float32(imgtmp['indian_pines_corrected'])
maptmp = loadmat("Indian_pines_gt.mat")
map = (maptmp['indian_pines_gt'])
```

La variable `img` est un tableau de taille $145 \times 145 \times 200$ de réels correspondant aux mesures pour différentes longueurs d'ondes. La variable `map` est un tableau de taille $145 \times 145 \times 1$ de `uchar8` correspondant aux labels que chaque pixels selon le tableau présenté ci-dessus (c'est la vérité terrain).

Vous pouvez afficher une composante de l'image :

```
res = img[:,:,18]
plt.imshow((res-np.min(res))/(np.max(res)-np.min(res)))
# On normalise avant affichage car ce sont des reels
plt.imshow(map) # Plus besoin de normaliser
```

- Appliquer une ACP sur la tableau correspondant à l'image multispectrale (n'oubliez pas de mettre l'image sous la forme d'un tableau de 145×145 lignes (individus) pour 200 colonnes (mesures) grâce à l'instruction `reshape`).
- Evaluer combien d'axes sont nécessaires pour conserver une grande partie de l'information. Que pouvez-vous conclure sur l'information initiale.
- Visualiser sous forme d'une image en niveaux de gris la projection sur le principal axe factoriel. Visualiser sous la forme d'une image couleur la projection sur les principaux axes factoriels. (n'oubliez pas de mettre le tableau de projection 145×145 lignes (individus) pour 3 colonnes (mesures) sous la forme d'une image de taille $145 \times 145 \times 3$ grâce à l'instruction `reshape`)
- Comparer avec la réalité terrain et conclure.

  Il peut être important d'avoir le lien entre ces composantes et les longueurs d'ondes initiales afin de déterminer la signature spectrale de l'information ainsi extraite. Cela indique les longueurs

d'ondes mise en valeurs suivant l'axe factoriel retenu. Nous pouvons calculer les corrélations entre les variables initiales (longueur d'ondes) et les composantes principales retenues, ou d'une manière plus simple le vecteur directeur de l'axe factoriel.

- Afficher avec l'instruction `plt.plot(pca.components_[k,:])` pour l'axe `k` la signature spectrale de l'axe.Commenter.