TP1 – Partie 1 : Analyse en Composantes Principales

Acquis d'apprentissage

- Comprendre l'ACP,
- Se familiariser avec l'application de l'ACP sur plusieurs jeux de données,
- Être capable d'extraire le taux de contribution à la variance de chacune des composantes principales,
- Reconstruire des données à partir de leur projection ACP.

PCA pas à pas (sans utilisation de l'implémentation PCA de scikit-learn)

Les données : Tableau des individus et des variables

n variables

X =

V_1	V_2	V_3	...	V_{n-2}	V_{n-1}	V_n
5	1	1	1	2	1	3
...	...	...	...	...	...	...
6	1	1	1	2	1	3

m observations

Attention

- L'ACP est appliquée uniquement sur les variables **numériques**
- Si pour un jeu de données une variable est **catégorielle** : il faudra soit la transformer en une variable numérique ou la supprimer (ne pas la considérer dans l'ACP).
- Pour un jeu de données, on doit **séparer les variables d'entrée** (caractéristiques/features) de **la variable cible**.

Dans ce qui suit, la variable **data** est un tableau numpy qui correspond au tableau des individus et des variables de dimensions m x n.

1. Écrire une fonction standardize_data(data) qui prend comme paramètres des données **data** et les retournes normalisées en utilisant la méthode **StandardScaler** de Scikit-Learn.

   La classe Standardscaler part du principe que les données sont normalement distribuées. Elle permet de transformer chaque caractéristique afin qu'elle ait une valeur moyenne de 0 et un écart type de 1.

Attention ! MinMaxscaler() normalise les données en mettant les valeurs de chaque caractéristique à l'échelle dans une plage [-1,1].

2. Écrire une fonction covariance(data) qui calcule et retourne la matrice de covariance du tableau des caractéristiques **data**
3. Écrire un script permettant de décomposer la matrice de covariance en valeurs et vecteurs propres.
4. Écrire une fonction compute_ACP permettant d'appliquer un ACP sur les données en entrée **data.**
5. Écrire un script permettant de calculer le pourcentage de variance expliquée pour chaque composante principale.
6. Dans un programme principal compute_ACP(data):
    a. Générez un ensemble de 500 vecteurs dans l'espace 3D, suivant une loi normale (de moyenne nulle et de variance unitaire), ensuite visualisez les points générés (utiliser np.random.randn(500,3)).
    b. Appeller la fonction compute_ACP et l'appliquer aux données générées.

Exercice 2 – PCA avec Scikit-Learn sur le jeu de données « iris »

Dans cet exercice, nous allons utiliser le dataset iris de sklearn.datasets de la bibliothèque Scikit-Learn vu en cours.

1. Lire le dataset iris en utilisant la méthode datasets.load_iris() de sklearn.datasets. Explorer ce dataset.
2. Appliquer l'ACP à ce dataset en utilisant 2 composantes principales. Vérifier la dimension des données après application de l'ACP.
3. Afficher les composantes principales en utilisant principal.components_
4. Afficher les données dans le nouveau repère des composantes principales en utilisant 2 composantes. Afficher chaque classe avec une couleur différente.
5. Calculer et afficher la proportion de variance expliquée (explained_variance_ratio_) pour chaque composante principale.
   NB. La variance expliquée dans l'ACP nous permet de comprendre la quantité d'informations conservées après la réduction de la dimensionnalité. Il s'agit de la part de la variabilité des données d'origine qui est capturée par chaque composante principale.
6. Écrire un code Python permettant de calculer le nombre de dimensions requises pour préserver 95% de la variance du jeu de données :
    a. Utiliser la méthode cumsum() de numpy.
    b. Proposer une autre méthode en jouant sur la valeur du paramètre n_components de la classe PCA de Scikit-learn.
7. Représenter la contribution de la variance en fonction du nombre de dimensions (représenter graphiquement cumsum). Interpréter.

Exercice 3 – Appliquer une PCA sur le jeu de données leaf

On considère le jeu de données leaf (télécharger le fichier leaf.csv ci-joint).
Il s'agit d'un ensemble de 340 feuilles issues de 40 espèces d'arbres (env. 8 feuilles par espèce) et illustrées dans la figure ci-dessous :

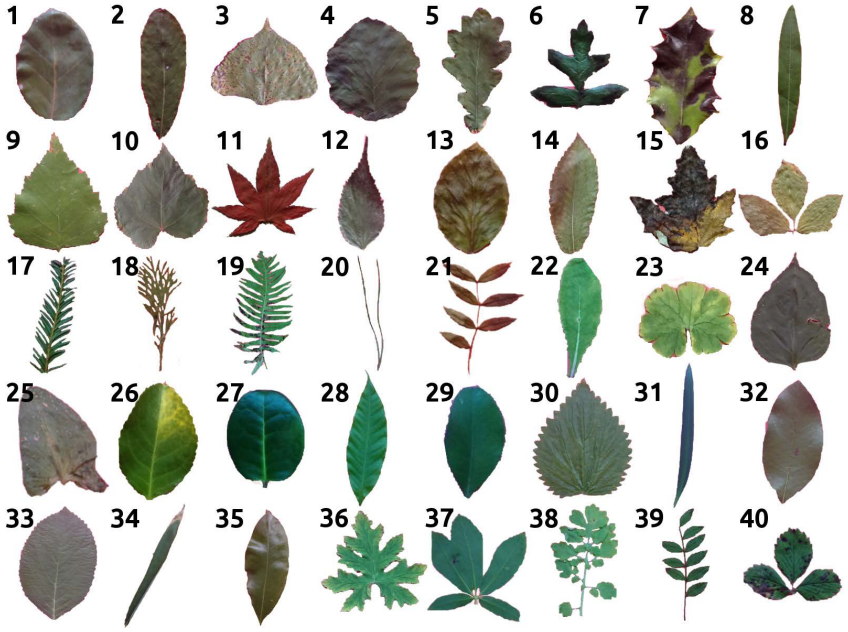

Figure 1. 26 Feuilles de 40 espèces d'arbres

Ces observations sont décrites par plusieurs variables (16 attributs) https://archive.ics.uci.edu/ml/datasets/Leaf.

1. Appliquez l'ACP aux données et affichez la décroissance des valeurs propres.
2. Affichez les projections des données sur les 3 premiers axes principaux en utilisant l'étiquette de classe pour donner une couleur à chaque classe.

Exercice 4.

Dans cet exercice nous allons utiliser le jeu de données MINST qui est composé de petites images de chiffres écrits à la main. L'étiquette associée à chaque image est le chiffre représenté. Nous allons utiliser l'ensemble de chiffres prétraité et intégré dans datasets de la bibliothèque Scikit-Learn (load_digits).

1. Lire et visualiser les données chiffres (digits) du dataset
2. Les données obtenues correspondent à des images 8x8. Donner la dimension des données ? Utiliser PCA pour visualizer ces données. Tracer les deux composantes principales.
3. Visualiser la reconstruction d'un chiffre en utilisant successivement : 1, 2, 3, 10, 20, 30, 50, 64 composantes principales. Interpréter.

TP1 Partie 2 – Descripteurs de Fourier

Acquis d'apprentissage

- Implémenter les descripteurs de Fourier d'une forme contour,
- Vérifier les propriétés d'un descripteur de formes.

On considère la base de formes MPEG-7 qui comprend 70 types d'objets ayant chacun 20 formes différentes (1400 formes). Chaque forme contour a été reparamétré par l'abscisse curviligne. Le fichier « MPEG-7_arc_length parametrization.csv » comprend la liste des 1400 formes contours reparamétrés.

1. Implémenter les descripteurs de Fourier vus en cours (penser à utiliser la fonction Discrete Fourier Transform (numpy.fft) de numpy).
2. Vérifier les propriétés de ces descripteurs
3. Évaluer la performance d'un système de recherche se basant sur ces descripteurs en traçant la courbe Précision en fonction du Rappel (vous pouvez utiliser la fonction sklearn : https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_curve.html).

Voir aussi ce lien pour mieux comprendre les notions de précision/rappel en classification : https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html