TELECOM ParisTech **SD204**
2015/2016 **Joseph Salmon, Maxime Sangnier, Florence d'Alché**

TP noté n° 2 : SVD-PCA

Pour ce travail vous devez déposer un **UNIQUE** fichier sous format `ipynb` sous EOLE, dans le dossier correspondant à votre groupe. Pour utiliser `ipython notebook` sur les machines de l'école utiliser la version notée "(Anaconda)" dans l'onglet "Applications/développement/". Le nom du fichier déposé sera **OBLIGATOIREMENT** `Nom_Prenom_TP2_group_k.ipynb` ($k \in \{1, 2, 3, 4\}$ est votre numéro de groupe). Vous devrez charger votre fichier sur Éole (SD204 > TP), dans le dossier correspondant à votre groupe, et ce avant 23h59, le 3/02/2016. La note totale est sur **20** points répartis comme suit :

- qualité des réponses aux questions : **15** pts,
- qualité de rédaction, de présentation et d'orthographe : **2** pts,
- indentation, Style PEP8 (*cf.* par exemple https://github.com/ipython/ipython/wiki/Extensions-Index#pep8), commentaires adaptés : **2** pts,
- absence de bug : **1** pt.

Malus : **5** pts par tranche de 12h de retard (sauf excuses validées par l'administration) ; 2 pts pour non respect des autres consignes de rendu.

Rappel : aucun travail par mail accepté

- Retour sur l'algèbre linéaire -

On rappelle l'*astuce du noyau* : pour tout $X \in \mathbb{R}^{n \times p}$ et $\mathbf{y} \in \mathbb{R}^n$ l'équation suivante est vraie :

$$X^\top (XX^\top + \lambda \, \mathrm{Id}_n)^{-1} \mathbf{y} = (X^\top X + \lambda \, \mathrm{Id}_p)^{-1} X^\top \mathbf{y} \tag{1}$$

1) Vérifier cette propriété numériquement pour $\lambda = 10^{-5}$ sans inverser de matrice (on pourra utiliser la fonction `array_equal` de `numpy`), pour X une matrice dont les entrées sont *i.i.d.* et tirées selon une loi gaussienne d'espérance 0 et de variance 4, et $\mathbf{y}$ est un vecteur dont les entrées sont tirées suivant une loi uniforme sur $[-1, 1]$, pour des tailles :
   (a) $n = 100$ et $p = 2000$
   (b) $n = 2000$ et $p = 100$
2) Pour les même X et $\mathbf{y}$ que ci-dessus, proposez une étude numérique et/ou graphique qui montre pour qu'elle taille de n et de p il est plus intéressant d'utiliser l'une ou l'autre des méthode de calcul dans l'équation (1). Commentez votre choix.

- Spectre de matrice aléatoire -

3) Choisissez deux lois de probabilités (non-gaussienne) de moyenne nulle et de variance 1, et construire dans les deux cas une matrice $X \in \mathbb{R}^{n \times p}$ dont les entrées sont tirées suivant cette loi.
4) Afficher sur un même graphique le spectre (*i.e.,* les valeurs propres) de $X^\top X/n$ pour $n = 1000$, pour $p = 200, 500, 1000, 2000$.

- Méthode de type puissance -

Reprendre une matrice $X \in \mathbb{R}^{p \times n}$ générée à la question 3).

5) Écrire une fonction qui code l'algorithme 1.

Algorithm 1:

Data: X ; le nombre maximal d'itérations : n_{iter} ; v initial
Result: v

$j = 0$
while $j < n_{\text{iter}}$ **do**
  $z \leftarrow Xv$
  $v \leftarrow X^\top z$
  $v \leftarrow v/\|v\|$
  $j \leftarrow j + 1$

6) Appliquer ce dernier avec diverses valeurs d'initialisation pour v, et divers nombres d'itérations max. Commenter sur la convergence, et identifier le vecteur v obtenu.

- PCA -

7) Importer avec `Pandas` la base de données `defra_consumption.csv` disponible ici https://sites.google.com/site/maximesangnier/defra_consumption.csv

8) Centrer et réduire les données avec `sklearn`, on notera $X \in \mathbb{R}^{n \times p}$ une telle matrice (avec n le nombre d'observations, et p le nombre de variables).

9) Appliquer l'ACP de X en affichant un scatter plot après projection sur les deux premiers axes.

10) Faire de même en gardant cette fois trois axes principaux (afficher donc un graphique en 3D).

11) Comparez les graphiques 2D et 3D obtenus ci-dessus avec ceux obtenus de la manière suivante :
   (a) Calculer la matrice $X^\top X$, et la diagonaliser. Projeter sur les vecteurs associés aux 2 valeurs propres les plus grandes (puis aux trois plus grandes).
   (b) Calculer la SVD de X. Projeter sur les vecteurs associés aux 2 valeurs singulières les plus grandes (puis aux trois plus grandes).
   (c) Évaluer les différences de temps de calcul pour ces différentes méthodes.

- Reconnaissance de visages par régression logistique -

L'objectif de cet exercice est de faire de la reconnaissance de visages automatiquement.

12) Charger les données grâce au code suivant :

```
from sklearn.datasets import fetch_lfw_people

# Download the data, if not already on disk and load it as numpy arrays
lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)
# introspect the images arrays to find the shapes (for plotting)
n_samples, h, w = lfw_people.images.shape
```

```
# for machine learning we use the 2 D data directly (as relative pixel
# positions info is ignored by this model)
X = lfw_people.data
n_features = X.shape[1]
# the label to predict is the id of the person
y = lfw_people.target
target_names = lfw_people.target_names
n_classes = target_names.shape[0]
print("Total dataset size:")
print("n_samples: %d" % n_samples)
print("n_features: %d" % n_features)
print("n_classes: %d" % n_classes)
```

13) Que représentent les variables explicatives ici ?

14) Pour éviter de lancer la régression logistique sur un trop grand nombre de variables explicatives, il faut réduire ce nombre avant de l'appliquer (ici, sans pénalisation). Vous comparerez deux approches :

   (a) utiliser l'ACP et le pourcentage d'inertie expliquée pour sélectionner avant régression logistique le nombre d'attributs ;

   (b) utiliser la validation croisée pour sélectionner le nombre de directions principales (vecteurs propres) trouvées par l'ACP qui seront utilisées comme nouvelle base pour apprendre la régression logistique. Vous pourrez utiliser la fonction `pipeline` de `scikit-learn`.

Liens pour aller plus loin :

⋆ ⋆ ⋆ http://www.astro.washington.edu/users/vanderplas/