---

## TP : SVD/PCA

---

For this lab, you have to upload a **single** ipynb file. Please use the following script to format your filename (bad name will lead to a 1 point penalty) :

```
# Change here using YOUR own first and last names
fn1 = "joseph"
ln1 = "salmon"
filename = "_".join(map(lambda s: s.strip().lower(),
                        ["SD204_lab2", ln1, fn1])) + ".ipynb"
```

You have to upload it on EOLE (site pédagogique / TP) before Wednesday 17/01/2018, 23h59 in the folder corresponding to your group. Out of 20 points, 5 are specifically dedicated to :

- Presentation quality : writing, clarity, no typos, visual efforts for graphs, titles, legend, colorblindness, etc. (2 points).
- Coding quality : indentation, PEP8 Style, readability, adapted comments, brevity (2 points)
- No bug on the grader's machine (1 point)

**Note :** you can use `https://github.com/agramfort/check_notebook` to check your notebook is fine, and also use `https://github.com/kenkoooo/jupyter-autopep8` to enforce `pep8` style.

**Beware** : labs submitted late, by email or uploaded in a wrong group folder will be graded 0/20.

---

### Exercice 1. (Linear algebra)

We remind the *kernel trick* : for any $X \in \mathbb{R}^{n\times p}$ and $\mathbf{y} \in \mathbb{R}^n$ the following equation holds true :

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

1) Check this property numerically for $\lambda = 10^{-5}$ without inverting any matrix[1], for a matrix $X$ whose entries are generated randomly (*i.i.d.* ) according to a Gaussian distribution with mean zero and variance 5, and for a vector $\mathbf{y}$ with coordinates generated randomly (*i.i.d.* ) according to a uniform distribution over $[-1, 1]$,
   (a) check it for $n = 100$ and $p = 2000$,
   (b) check it for $n = 2000$ and $p = 100$.
2) For similar scenarios to (a) and (b), propose a numerical/graphical study to compare (according to $n$ and $p$) when it is more time efficient to use one of the two methods provided by (1) to predict with a ridge estimate. Synthesize your experiment.

### Exercice 2. (Random matrix spectrum)

3) Choose three non-Gaussian probability distributions, with mean 0 and variance 2, and write a <u>function</u> that to $n$ and $p$ create a matrix $X \in \mathbb{R}^{n\times p}$ with entries generated (*i.i.d.* ) according to each distribution.
4) Display on one single graph the singular value of $X$ for $n = 1000$, and $p = 200, 500, 1000, 2000$ for the three distribution chosen.
5) Display on one single graph the spectrum (*i.e.,* the set of eigen values) of $X^\top X/n$ for $n = 1000$, and $p = 200, 500, 1000, 2000$.

### Exercice 3. (Power method)

We considers matrices $X \in \mathbb{R}^{n\times p}$ generate as in Exercise 1, question 1a.

1. you could use for instance a testing tool from `numpy` such as `allclose`

6) Write a function coding the following Algorithm 1.

**Algorithme 1 :** Power method

**Input** : Matrix $X$ ; maximal number of iterations : $n_{\text{iter}}$, $u \in \mathbb{R}^n, v \in \mathbb{R}^p$ : initial vectors
**for** $j = 0, \ldots, n_{\text{iter}}$ **do**
  $u \leftarrow Xv$ // $u$ update
  $v \leftarrow X^\top u$ // $v$ update
  $v \leftarrow v/\|v\|$ ; $u \leftarrow u/\|u\|$ // Normalization
**Output :** $u, v$

7) Illustrate visually (with a graph) a case where the algorithm converges. Is this true that the output $u, v$ from the algorithm converge to the singular vector associated to the larges singular values of $X$ ?

8) Provide two sets of initialization vectors leading to different limits for this algorithm ; explain how they are related.

9) Provide a way to approximate the largest singular value of $X$ using the power method.

10) Build upon the power method to provide an algorithm that can approximate the second largest singular value of $X$ (without using an SVD function). On could use twice the power method.

### Exercice 4. (PCA)

11) Import with `Pandas` the dataset `defra_consumption.csv` available here : http://josephsalmon.eu/enseignement/TELECOM/SD204/defra_consumption.csv

12) Center and standard the dataset using the `preprocessing` module from `sklearn`, we write $X \in \mathbb{R}^{n\times p}$ the associated matrix ($n$ is the number of observations, and $p$ is the number of variables).

13) Display a *scatter plot* of the $n$ points projected on the space generated by the first principal axes.

14) Repeat the previous question for the space generated by three axes.

15) Compare the previous 2D and 3D graphs with the one obtained as follows :

   (a) Compute $X^\top X$, and diagonalize it. Project the $n$ points encoded by $X$ over the span of the eigen vectors associated to the two (respectively three) largest eigen values.

   (b) Compute the SVD of $X$. Project the $n$ points encoded by $X \in \mathbb{R}^{n\times p}$ over the span of the left singular vectors associated to the two (respectively three) largest singular values [2].

   (c) Evaluate the difference in timing for the two methods.

### Exercice 5. (Logistic regression for face classification)

Charge the dataset using the following script :

```
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
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)
```

2. you could use for instance, the `TruncatedSVD` from `sklearn` for this task

16) Describe precisely what are the features in this dataset. How many features are available ?

17) To avoid performing a logistic regression over too many features, you need to reduce the dimension. Compare the performance of the two following methods (in term of accuracy / number of mistakes produced) :

    (a) perform a PCA step with an explained variance (or inertia) percentage of 95% to transform the original dataset, before performing a logistic regression step ;

    (b) use cross-validation to select the number of principal axis in a procedure performing a PCA followed by a logistic regression step. Use for that a `pipeline` framework `scikit-learn`.