# TP 4 – AOS1
# PCA

## 1 Python warm up: PCA by hand

(1) Generate a dataset with the following instruction

```
X = np.random.multivariate_normal([1, 3], [[2, 1], [1, 2]], 100)
```

How many samples are generated? How many features? What is the underlying distribution of samples in $X$?

(2) Verify the relation that exists between singular values and eigenvalues using a matrix $X$. To use the functions provided by the `scipy` library, use the following command:

```
import scipy.linalg as linalg
```

and look at the functions `linalg.eig`, `linalg.eigh`, `linalg.eigvals`, `linalg.eigvalsh`, `linalg.svd` `linalg.svdvals`

(3) Compute the principal directions and principal components by hand using the unbiased variance–covariance estimator. Verify that they coincide with the ones computed by `scikit-learn`.

## 2 PCA for dimension reduction

In this section, we use the `house_prices` regression dataset. To load it use

```
from sklearn.datasets import fetch_openml
housing = fetch_openml(name="house_prices", as_frame=True)
X = housing.frame
X = X.loc[:, ~X.isna().any() & (X.dtypes.astype(str).isin(["float64", "int64"]))]
y = housing.target
```

(4) Perform a PCA on this dataset and study how many number of principal components should be retained from the two empirical methods seen in class.

(5) Describe the following code. What is it supposed to be doing? Adapt it to determine the optimal number of principal components for the regression task at hand.

```
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

pca = PCA()
lin = LinearRegression()
pca_lin = Pipeline([("scale", StandardScaler()), ("pca", pca), ("lin", lin)])
clf = GridSearchCV(
    estimator=pca_lin,
    cv=10,
    param_grid=dict(pca__n_components=range(1, X.shape[1] + 1)),
)
clf.fit(X, y)
```

## 3 Problem: band reduction in multispectral images

A multispectral image is an image that has several components. For example, a color image has 3 components: red, green and blue and each pixel can be viewed as a vector in $\mathbb{R}^3$. More generally a multispectral image of size $N \times M$ with $P$ spectral bands can be stored as a $N \times M \times P$ array. There are $N \times M$ pixels living in $\mathbb{R}^p$.

When the number of spectral bands $P$ is too large, it is desirable to somehow reduce that number ultimately to 3 for viewing purposes. This process is called band reduction.

Propose a method using the PCA performing a band reduction to 3 bands and use it on the provided multispectral image.

Some multispectral images are available on the internet to test your band reduction algorithm. See for example the following website

- `http://lesun.weebly.com/hyperspectral-data-set.html`

Most of them are available as a Matlab data file (.mat files). It can be loaded with `scipy` with the following function

```
scipy.io.loadmat
```

You will probably have to reshape arrays. It can be done with the `reshape` method. For example, an array of size $6 \times 6 \times 3$ can be “linearized” using reshape

```
X_lin = X.reshape((-1, 3))
```

the $-1$ is automatically inferred from the number of elements in the array. The array is then reshaped into an array of size $36 \times 3$.

It might be handy to be able to rescale the data when it has to belong the some specific range. `scikit-learn` has several rescalers available. For example

```
from sklearn.preprocessing import MinMaxScaler
```

rescales the data between 0 and 1.

`matplotlib` can display images with the function

```
plt.imshow
```

Beware of the type of the array (float or integers)!