MAP654I Practical introduction to Machine Learning
Practical Session 1 Unsupervised learning

Rémi Flamary

The objective of this practical session is to manipulate and understand the machine learning problems and methods discussed in the course. The practical session will be done in Python 3 and it is strongly recommended to have a working Anaconda environnement. The different sections of the session should be implemented in Jupyter notebooks and it is strongly recommended to take notes in the notebook during the session.

The individual reports (with figures and discussions) for the session will be uploaded on moodle in python notebook format. It is expected to have a working code (reproducible figures) and a short discussion in markdown format for all the results obtained in the practical session. Note that ALL plotting and visualization questions in the practical session require a **critical discussion** of the visualized result that will be graded.

The end of the report must contain a personal discussion about the session (what was hard to understand and implement, how you would do it next time, what was new, discussion of relation with the course, personal discussion about how to use these tools in a professional setting, ...).

Importing libraries

In this practical session we will use Numpy/Scipy Python libraries for handling numerical data and Matplotlib for plotting them.

```
import numpy as np
import pylab as pl
import scipy as sp
```

Note that a more standard import for matplotlib is `import matplotlib.pyplot as plt`. You can use the name you want for the plotting module but if you choose `plt` you need to replace it in the function names given in the following (`plt.plot` instead of `pl.plot`). We will also need to have access to some functions and classes in the Scikit-learn toolbox.

1 Datasets

In this practical session we will use two simple datasets in numpy format (loaded with `np.load("filename.npz")`) to illustrate and study the different unsupervised learning problems. Those datasets are both easy to understand and interpret and will give a good overview of how to interpret the unsupervised learning results in practical applications.

You will need to apply the learning methods on both datasets but note that due to their properties, interpretation will be done differently (with `plot` on temperatures and `imshow` on digits images for instance).

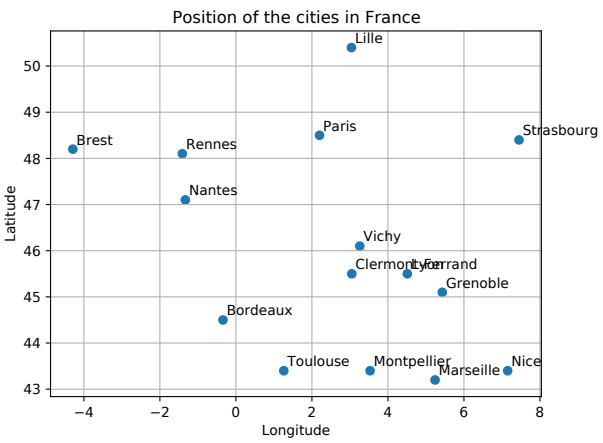
1.1 Temperature dataset

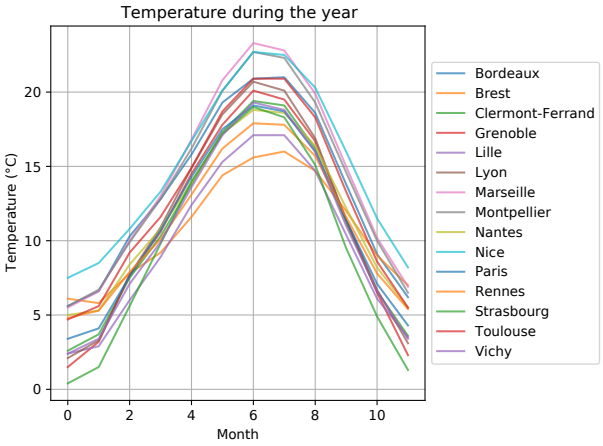
This dataset contains the average (over the years) temperature in Celsius for 15 cities in France. The geographical localization of the cities is also provided because it will help in interpreting the results (but will not be used when applying the methods).

The data file `"temper.npz"` contains the following information:

- `"data"` : a 15×14 matrix containing for each of the 15 french cities the temperature for each month on the first 12 columns and the latitude and longitude of the cities in the last two columns. The data matrix `x1` for this dataset will be the 15×12 matrix with only temperature.
- `"villes"` : a list of the names of the cities that will be used for readable legends in the figures and visualizing the cities in 2D (geographical position and/or dimensionality reduction).
- `"varname"` : a list of names for the variables/columns in the `"data"` matrix, containing month names and latitude/longitude.

A basic loading and plotting of the data leads to the following figures:





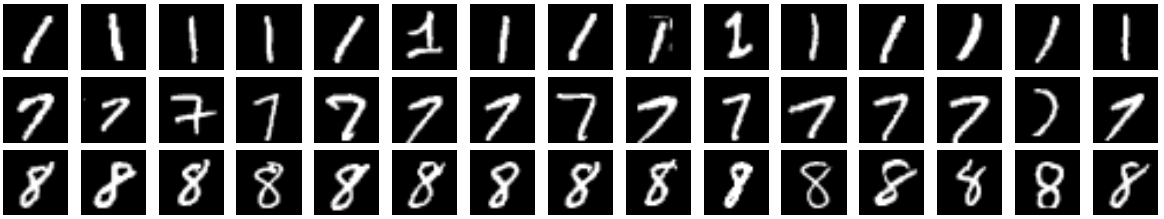
1.2 Digits dataset

This dataset is a small subset of the well known MNIST dataset (3 classes only 1000 samples per class on train data) and contains images of written numbers. One interesting complementary information is of course the class of those images (the number that was written).

The file `"digits.npz"` contains the following matrices:

- `"x"` and `"xt"`: data matrices containing respectively $n = 3000$ and $n_t = 1500$ training example of written digits. Each line in those matrices is a 28×28 image stored as a transposed vector (a line of size 784).

  Some examples of the images in the training sets:



- `"y"` and `"yt"`: the labels of the images described above. they are vectors containing the classes $(1, 7, 8)$ of each images in $\mathbf{x}$ and $\mathbf{xt}$. The samples are sorted by class.

1.3 Loading the data

For both datasets do the following:

code

interpretation

1. Load the data with `np.load("filename.npz")` and store the different matrices in memory (for instance in variables `x1` for temper data and `x2,y2` for digits data). For the digits dataset, it is better to perform one simple pre-processing that scales the values between $[0, 1]$ by dividing the data matrix by 255.
2. Do a quick look at the data, compute the mean values for each variable and visualize it (with `pl.plot` for the temperature signals and `pl.imshow` for the images in digits data).
3. (Bonus) Reproduce the Figures above.

2 Clustering

In this section we will perform clustering with the K-means algorithm and interpret the results.

2.1 For both datasets

1. Perform K-means clustering on the data ($K = 3$, `sklearn.cluster.Kmeans`) and store the cluster centroids and the estimated classes for all samples.
2. Plot the centroids and interpret them (how are the samples clustered? by class on digits? geographical position for the cities?).
3. Change the value of K and repeat the two previous steps. What are good values in your opinion and why?

2.2 Temperature dataset

1. Look at the name of the cities in each clusters. Do the clustering make sens?
2. Plot the cities with a scatterplot using their geographical position and using the estimated class as color. Do the K-means clustering on temperature recover a geographical similarity between cities? Why?

2.3 Digits dataset

1. Does the clusters resemble a true image from the dataset ? Could you tell if a cluster centroid is a true image. Why ?
2. Plot the predicted labels (as a signal). Since the samples are ordered by class in the dataset, a clustering respecting the true class should be piecewise constant. Is that the case? → yes
3. Compute the quality of the clustering using the ground truth labels (`sklearn.metric.rand_score`, `sklearn.metric.adjusted_rand_score`).
4. See the effect of the parameter K on the clustering score. What is the K that leads to the best score? Why? ↳ 3

3 Density estimation

In this section we will perform density estimation with gaussian Mixture Models (GMM) and interpret the estimated model.

3.1 For both datasets

1. Estimate a GMM density on the data with $K = 2$ for temperature and $K = 3$ for digits (`sklearn.mixture.GaussianMixture`). Due to the relatively high dimensional data and small number of samples we will estimate GMM with diagonal covariances (`covariance_type='diag'`).
2. Plot the centers of the Gaussian distributions in the mixture and interpret them.
3. Change the value of K and repeat the two previous steps. What are good values in your opinion and why? Change the shape of the covariance, how well are they estimated on both datasets?
4. Compute the log probability of the samples for the estimated density (`model.score_samples(x)`). Recover the samples with the smallest score (probability) and plot them. They can be considered as outliers for the distribution (least probable samples). Wan you see why?
5. Digits dataset : Estimate the labels of the samples (`model.predict`) and compute the clustering rand score. Is it better than Kmeans? Why?
6. (Bonus) generate samples from the distribution (`model.samples`) and plot them. Could you differentiate them from real data?

4 Dimensionality reduction

In this section we will perform dimensionality reduction and interpret the model and results. All the questions have to be performed on both datasets.

4.1 Linear Projection : Principal Component Analysis

1. Compute the covariance matrix of the data and plot it as an image (`np.cov(x.T)`). Interpret the structure of the covariance matrix using your knowledge of the data (relations between features).
2. Compute the PCA (keeping all dimensions with `n_components=None`) for the data and recover the explained variance ratio (stored in `model.explained_variance_ratio_`) that is the proportion of energy in each principal direction stored. Plot it and discuss the quantity of information preserved when projecting on $p = 2$ dimensions.
3. Project the samples in $p = 2$ dimensions and plot the projected samples (`model.transform(x)`). For temperature data also plot the name of the city close to the sample (`pl.text(x,y,'text')`) for a better interpretation. For the digits dataset, color the samples with their class in the scatterplot. Interpret the relations between the samples, what information is preserved and is it correlated to what your know about the samples (geographic position or true class) ?
4. Get the the two first principal directions (`model.components_`) from the model. Plot them and interpret them. What is the impact of moving along those direction in the original space?
5. Reconstruct the data from the projected samples (`model.inverse_transform(xp)`) for different subspace dimension p. For a few samples in the dataset plot, the sample and its reconstruction. Is it still recognizable? Look at the impact of p on the quality of reconstruction.
6. (Bonus) Estimate and visualize other linear modeling methods such as NMF that can be used on positive of the data.

4.2 Manifold Learning : TSNE

1. Compute the TSNE embedding of the data for in $p = 2$ for both datasets (`sklearn.manifold.TSNE`).
2. Project the samples and plot them. For temperature data also plot the name of the city close to the sample (`pl.text(x,y,'text')`). For the digits dataset, color the samples with their class in the scatterplot.
3. What is the effect of the `perplexity` parameter? Does the embedding recover geographical relations for the temperature dataset? Is the embedding more discriminant in 2D for the digits dataset than the ACP?