PCA and t-SNE

Exercise 1

We want to perform PCA on a classical dataset : the Olivetti faces dataset.

1. Import the dataset using the function `fetch_olivetti_faces`
2. Center the faces
   `faces_centered = faces - faces.mean(axis=0)`
3. We now plot the faces

```
import numpy as np
import matplotlib.pyplot as plt
n_row, n_col = 5, 7
n_components = n_row * n_col
image_shape = (64, 64)

def plot_gallery(title, images):
    plt.figure(figsize=(2. * n_col, 2.26 * n_row))
    plt.suptitle(title, size=16)
    for i, comp in enumerate(images):
        plt.subplot(n_row, n_col, i + 1)
        comp = comp.reshape(image_shape)
        vmax = comp.max()
        vmin = comp.min()
        plt.imshow(comp, cmap=plt.cm.gray, vmax=vmax, vmin=vmin)
        plt.xticks(())
        plt.yticks(())

    plt.subplots_adjust(0.01, 0.05, 0.99, 0.93, 0.04, 0.)


# Plot a sample of the input data
plot_gallery("First centered Olivetti faces", faces_centered[:n_components])
```

4. Perform PCA with 20 components. Plot the 20 first components

Exercise 2

Word embeddding is mapping allowing to represent each word of a given vocabulary by means of a high-dimensional numerical vector (around hundred of components).

The idea is that word embeddings should encode the `semantic` (i.e. the sense of the words). It means that when two words that have close meaning, their representation should be close also : the euclidean distance between their embeddings should be small. For example `mom` and `dad` should be closer than `socket` and `dad`.

To understand more about words embeddings, and see how we can visualize clusters of words how we can visualize word clusters, we shall combine `Word2Vec` representation and `t-SNE`. To do so we shall use `gensim` Python library and begin to import the following Python librairies

```
import numpy as np
import gensim.downloader
import matplotlib.pyplot as plt
from matplotlib import cm
from numpy import linalg as LA
```

The embeddings can be downloaded using the following command

```
word2vec = gensim.downloader.load('word2vec-google-news-300')
```

1. We first play with word embeddings to understand more
   (a) Calculate the word embedding of the word `Paris`
   ```
   emb_paris=word2vec['Paris']
   ```
   What is the shape of this vector?
   (b) We now display the 30 most similar words to the word `Paris` using the function `most_similar`. Comment both the result and the format of the output
   (c) To understand more how word embeddings encode semantic, we calculate also the word embedding of the two words `London` and `twitter`. Calculate the cosine similarity of (`Paris`,`London`) and thereafter that of (`Paris`,`twitter`) where

$$cosine\_sim(Paris, London) = \frac{\langle embedding(Paris), embedding(London)\rangle}{\|embedding(Paris)\| \cdot \|embedding(London)\|}$$

   and

$$cosine\_sim(Paris, twitter) = \frac{\langle embedding(Paris), embedding(twitter)\rangle}{\|embedding(Paris)\| \cdot \|embedding(twitter)\|}$$

   (d) Comment
2. We now create synthetic data, naturally associated to clusters.

```
keys = ['Paris', 'Python', 'Sunday', 'Tolstoy', 'Twitter', 'bachelor', 'delivery',
'election', 'expensive', 'experience', 'financial', 'food', 'iOS', 'peace',
'release', 'war']

embedding_clusters = []
word_clusters = []
for word in keys:
    embeddings = []
    words = []
    for similar_word, similarity in word2vec.most_similar(word, topn=30):
        words.append(similar_word)
        embeddings.append(word2vec[similar_word])
    embedding_clusters.append(embeddings)
```

```
        word_clusters.append(words)

embedding_clusters = np.array(embedding_clusters)
n, m, k = embedding_clusters.shape
```

   (a) Comment the shape of the object `embedding_clusters`? To what are related n, m and k?
   (b) What is the type of the object `embedding_clusters`? Can be it directly used as an input of the PCA algorithm of scikit-learn?

3. We now use PCA with two components on our synthetic clusters
   (a) Perform PCA on the synthetic dataset `embedding_clusters` *Hint :* you will need to reshape the object `embedding_clusters`
   `https://www.w3schools.com/python/numpy_array_reshape.asp`
   (b) What is the explained variance? Project the dataset on the two first principal components.
   (c) Visualize the different clusters related to each `key` using the following visualisation function

```
[ ] def plot_similar_words(title, labels, embedding_clusters, word_clusters):
        plt.figure(figsize=(16, 9))
        colors = cm.rainbow(np.linspace(0, 1, len(labels)))
        for label, embeddings, words, color in zip(labels, embedding_clusters, word_clusters, colors):
            x = embeddings[:, 0]
            y = embeddings[:, 1]
            plt.scatter(x, y, c=color, label=label)
            for i, word in enumerate(words):
                plt.annotate(word, alpha=0.5, xy=(x[i], y[i]), xytext=(5, 2),
                             textcoords='offset points', ha='right', va='bottom', size=8)
        plt.legend(loc=4)
        plt.title(title)
        plt.grid(True)
        plt.show()
```

   Caveat : to use it you should reshape the output

4. Perform t-SNE and and visualize the different clusters related to each `key` in the t-SNE space

5. Compare both