Artificial Intelligence: Lab 1

Unsupervised Learning

In unsupervised machine learning algorithms, we do not have any supervisor to provide any guidance. We do not have correct answers and algorithms need to discover the interesting pattern in data for learning. One of the most used approaches in this domain is clustering. It aims to divide the set of observations into subsets, called clusters, in such a way that observations in the same cluster are similar and they are dissimilar to the observations in other clusters.

For this lab, you need to download the file `l2_lab1_skeleton.py` available at the address :

`https://www.labri.fr/~zemmari/l2-ai`.

Exercise 1. K-Means algorithm

K-means clustering algorithm is one of the well-known algorithms for clustering the data. The algorithm acts in an iterative way and is parametrised by K : the numbers of clusters we want to get.

1. Import the following `python` modules : `matplotlib.pyplot`, `seaborn  sns.set()`, `numpy` and `KMeans`.
2. Execute the following code and observe what you obtain :

```
from sklearn.datasets.samples_generator import make_blobs
X, y = make_blobs(n_samples = 500, centers = 4,
        cluster_std = 0.40, random_state = 0)
plt.scatter(X[:, 0], X[:, 1], s = 50);
plt.show()
```

3. Initialise `km` to be the K-means algorithm, with the required parameter of how many clusters (`n_clusters`).
4. Train the K-means model with the input data.
5. Execute the following code to visualise the result of your training :

```
y_kmeans = km.predict(X)
plt.scatter(X[:, 0], X[:, 1], c = y_kmeans, s = 50, cmap = 'viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c = 'black', s = 200, alpha = 0.5);
plt.show()
```

Exercise 2. K-Means Algorithm for Colour Compression.

One interesting application of clustering is in colour compression within images. For example, imagine you have an image with millions of colours. In most images, a large number of the colours will be unused, and many of the pixels in the image will have similar or even identical colours.

Execute the following code :

```
from sklearn.datasets import load_sample_image
china = load_sample_image("china.jpg")
ax = plt.axes(xticks=[], yticks=[])
ax.imshow(china);
```

It shows an image from datasets of `sklearn`.

1. Explain the result of the following instruction :

   ```
   china.shape
   ```

2. On can see the image as a set of pixels and hence as as a cloud of points in a three-dimensional colour space. Give the instructions to reshape the data to `[n_samples x n_features]`, and rescale the colours so that they lie between 0 and 1.

You can use the function `plot_pixels` in the given file to visualise a subset of pixels.

Now we will use k-means algorithm to reduce these 16 million colours to just 16 colours. We will use a slightly different implementation of this algorithm. We will use the mini batch k-means which operates on subsets of the data to compute the result much more quickly than the standard k-means algorithm.

1. Give `python` instructions to obtain the new colours (the min batch k-means algorithm is implemented using `MiniBatchKMeans` function in `sklearn.cluster` module.
2. Execute the following instructions to observe the result :

   ```
   china_recolored = new_colors.reshape(china.shape)
   fig, ax = plt.subplots(1, 2, figsize=(16, 6),
                          subplot_kw=dict(xticks=[], yticks=[]))
   fig.subplots_adjust(wspace=0.05)
   ax[0].imshow(china)
   ax[0].set_title('Original Image', size=16)
   ax[1].imshow(china_recolored)
   ax[1].set_title('16-color Image', size=16);
   ```