LAB8 – K-Means Clustering

In this lab, we are going to do some painting!

Task 1: Constructing our input

The original file and its black pixel coordinates are going to be our x1 and x2 values. Here are instructions on how you can extract pixel coordinates from an image:

First, you have to install the "Pillow" package. As you already know, this can easily be done from within the IDE (PyCharm, Anaconda etc.). Inside the Pillow package (which is an extension of the Python Imaging Library PIL), there is a class called "Image". You have to import the Image class using this import line:

```
from PIL import Image
```

The Image class and its functions can be seen here:
https://pillow.readthedocs.io/en/3.3.x/reference/Image.html

After opening an Image object using the image provided in the LAB8 folder, you have to:

1. Change the mode of the image to "1" (using the "convert" function).
2. Resize it to (320, 256) to not deal with long running times (using the "resize" function).

Now, we want to get the black pixels' coordinates. Unfortunately, there is no function that returns pixel coordinates directly. To achieve this, we will iterate through all of the pixels, and store black pixels as we go:

- Open two empty arrays (one for the x-coordinates, another for y-coordinates).
- Inside two nested "for" loops, check if coordinate (i, j) is black. You can use the "getpixel" function for this purpose.
- If the pixel is black (If the pixel value is 0), append i and j values to your arrays.

Task 2: Separating the input into 4 clusters

We are going to use the K-means clustering method to divide our input into 4 different clusters. We are going to use sklearn's "KMeans" class for this purpose. See the documentation here:
https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Similar to every other regression method, we will first instantiate a Kmeans object with the number of clusters (or K) given as a parameter. We will then call the fit(X) and predict(X) functions, respectively. (When calling these, X should be a matrix, so don't forget to merge your arrays from before into a matrix!)

The predict(X) function returns an integer array. For each (x1, x2) pair, this array holds an integer from 0 to K (K=4 in our case). These are our labels for the data points. Your next task is to separate your input into 4 different pieces (clusters). You will simply look at the labels and decide to which cluster a data point belongs to. Example: If labels[i] is 0, then my i^{th} data point belongs to the first cluster. If it is 1, then my i^{th} data point belongs to the second cluster, and so on.

Task 3: Plotting

Now, we have two figures. In one figure, you will plot the non-separated data (your x1 in one axis and x2 on another) in black. You should see the black and white flower image in this figure.

In the next figure, plot the separated data, each one in another color.

Note that you can simply change your K value (when instantiating your Kmeans object) to any value smaller than 4 to get different results. Try to experiment!

All of the plots should be scatter plots. Here's the result: