Clustering Project

Setting the seed for random numbers to 1

All program in this project implement randomized algorithms. Without fixing the seed for the random number generator they may give a different results in each run. We ask that you fix the random seed to 1 so that each run should produce the exact same result. This can be done by putting the following lines at the top of your program:

```
--------------------------------------------------
# set the random seeds to make sure your results are reproducible
from numpy.random import seed
seed(1)
--------------------------------------------------
```

Obtaining the data

When we test your programs we will use 50% randomly selected instances from the "iris" dataset. The entire dataset is given in the file "iris-data.csv". (The correct clustering is in the file "iris-labels.csv", also available, but it will not be used it in this project.)
To obtain 50% randomly selected items you can use the program "fraction_xy.py" as follows:

```
Python3 fraction_xy.py  iris-data.csv iris-labels.csv 0.5 7
```

This will create the desired random files, with the random number generator initialized to 7. (You can experiment with different random initializations, replacing 7 with other numbers.)

Grading

We will generate a random dataset by running the program **fraction_xy.py** with a seed that is kept secret. Your programs will be tested on that random dataset. We will evaluate the correctness of your Part I implementation and the correctness and accuracy of σ selected in Part II.
Your programs should work for arbitrary values of k. The correctness will be evaluated with k kept secret. The accuracy will be evaluated with $k = 3$.

Part I

Implement the k-means and the k-means++ algorithms in python. Write the following two python programs:

1. **lloyd.py** : Lloyd's k-means.
2. **kmeanspp.py** : k-means++.

Each program should get the following 3 input values (from the command line):

Input1: A data file. The data is a comma separated matrix of size $n \times m$. Here the data points are the rows (not the columns) of the matrix.

Input2: k, the number of desired clusters.

Input3: r, the number of random iterations.

Each program should have two outputs.

Output1: A comma separated file containing n integer values. Each integer value is in the range $0 \ldots k-1$, specifying the cluster of the corresponding row.

Output2: The quantization error. This is the error E defined in the k-means handout. It should be printed.

The 3 input parameters and the name of the output file should be given in the command line. The quantization error should be printed. Running the program should be as follows:

```
Python3 yourprogram.py inputdata k r outputclusters
```

For example:

```
Python3 lloyd.py data1.txt 2 5 labels1.txt
```

You can visualize your results using the program **plot.py**.

Part II

Implement the spectral clustering technique that partitions the data into k clusters.

- The data should be converted into a complete graph (no nearest neighbors) where the weights w_{ij} are computed as shown in the handout.
- Observe that the value of σ is not given. It is your responsibility to find a good σ value by running multiple experiments.
- Implement the direct k-clustering method as described in the handout, where k-means (or k-means++) is used to compute the clustering in the Laplacian space.

Please name your program for Part II **spectral.py**. It should get the following 3 input values (from the command line):

Input1: A data file. Same as the data file of Part I.

Input2: k, the number of desired clusters. Same as in Part I.

Input3: σ.

The program should have 2 outputs.

Output1: A comma separated file containing n integer values. Same as in Part I.

Output2: The quantization error. Same as in Part I. This value should be printed.

The 3 input parameters and the name of the output file should be given in the command line. The value of the quantization error should be printed. Running the program should be as follows:

```
Python3 spectral.py inputdata k sigma outputclusters
```

For example:

```
Python3 spectral.py data1.txt 3 1.5 labels1.txt
```

Observe that the internal parameter r to be used by the internal k-means algorithm is to be hard coded and not given as input.

What you need to submit

- Source code and documentation for the python scripts.
- The value of σ to be used by your program in Part II.

You **must** be available to demonstrate your program to the TAs. Time slots and additional instructions will be announced later.

Deadline:

TBA.