Project 1 - README

ΜΥΣΤΑΚΙΔΗΣ ΙΩΑΝΝΗΣ - 1115201600113
ΧΟΥΣΙΑΔΑ ΕΥΑΓΓΕΛΙΑ - 1115201600200

Github: https://github.com/myioannis/Project-1.

How to build:

You can compile the code using the *make* tool in the source directory.

comments:

- There are three executables produced:
  - lsh: runs the classification algorithm using the lsh technique
  - cube: runs the classification algorithm using the lsh technique
  - cluster: runs the clustering algorithm with the method specified by the flag -m
- We've made use of conditional compilation in some of the files in order to produce a different executable in each of the aforementioned cases (and in many cases we tried to get rid of duplicate code that was dependent on the method used specifically)
- We used the -O3 optimization flag during the compilation

How to run:

You can run the executables exactly as specified in the assignment.

comments:

- For the classification, the code reads only 3 queries from the queryset. You can change this in the *read_Queryset*() function in Input.cpp.
- In each case, there are some arguments that are mandatory for the execution:
  - In classification: the dataset path, the queryset path and the output path are mandatory

  - In clustering: the dataset path, the output path, the configuration file path and the method are mandatory

  all other parameters (e.g. L, k etc.) can be ommited because they have default values (except the number of clusters in the configuration file for the clustering)

- For each file, you have to provide its **whole/absolute** path

Modules:

- **Image** module: This is a class that contains the structure of an image, i.e. the length of the image, the pixels themselves and order of insertion/creation of the image (its identifier).
- **Utils** module: It contains some utilities, e.g. modular exponentiation, endianness conversion and the manhattan distance between two images, that serve the other modules.
- **Input** module: This is a class that serves as the main interface between the user and any algorithms/methods used. It contains all the data/queries and parameters, and creates the appropriate objects for classification/clustering depending on the executable running. The main function is communicating through an instance of this class. That is, we do classification or clustering *on this Input*.
- **Technique** module: This is an abstract class used in classification. Its subclasses are LSH,Hypercube and Exact (see below). Apart from serving as an abstract base class, it is also used explicitly in the Input module, where a pointer of the base class type is used to achieve run time polymorphism.
- **LSH** module: This is a subclass of *Technique* that contains algorithms for the classification using locality sensitive hashing.
- **Hypercube** module: This is a subclass of *Technique* that contains algorithms for the classification using randomized projections.
- **Exact** module: This is a subclass of *Technique* that contains algorithms for the classification using exhaustive search.
- **Cluster** module: This is a class that contains all the information needed to create the clusters from the dataset and compute the silhouette coefficients.

Benchmarks:

- It can be shown, experimentally, that the best K for clustering is 10.
- The best value that worked for us for w in classification is around 10.000.