Empirical/Programming Assignment 3

Due date: Monday, 11/6

Please note new submission times: electronic submission is due by 12:00 noon of the due date. Paper copies are due by the beginning of class.

In this assignment you will implement and test properties of the $k$-means algorithm.

1 Data

For this assignment we use 8 datasets. The first 5 datasets were generated artificially. All these datasets were generated using 5 classes each corresponding to a Gaussian sphere. The main difference between these datasets relates to how close the spheres are to one another relative to their width. The closer they are the harder it is to cluster the dataset. We additionally have 3 real datasets from the UCI machine learning repository. The features in all files have been processed (standardized or normalized) so that no further pre-processing is required. We explicitly provide the class labels for the purpose of calculating NMI with the labels, but the class feature should not be used for the purpose of clustering. The data files are accessible through the course web page.

2 Your Tasks

2.1 Implementing $k$-Means

Implement the $k$-means algorithm in order to cluster the data mentioned above. You can reuse your arff parser from project 1 to read the arff files, but please do not use parsers from any other source.

Recall that $k$-means iteratively updates the means and the cluster assignments. It sometimes occurs that due to this process a cluster becomes empty during the update. In this case the mean of any empty cluster is initialized with a random example from the dataset.

As noted above, although this information is given in the same data file, please make sure not to use the class label for distance calculation.

The $k$-means code should report the cluster scatter $CS = \sum_j \sum_{x \in C_j} \|x - \mu_j\|^2$ and the NMI of the computed clustering to the class labels. Explanation and equations for NMI are given in the lecture slides.

2.2 Investigating the Effect of Initialization on $k$-means

Here we explore two approaches to pick a good initialization for $k$-means. In this part, set $k$ to be the number of classes in the corresponding dataset (you can identify this from the data file).

The first approach uses random restarts. The means are initialized to a random selection from the dataset. This is repeated multiple times with the intention of selecting the run with the best score. Your code should run the algorithm 10 times with different random initializations. For each run you record the CS and NMI.

The second attempts to use a “smart initialization”. Here we start with one mean being initialized to a random example. Then repeat the following until all means have been assigned: (1) Pick 10 yet unused random examples. (2) Among these select the example which is farthest away from all previously selected means. That is, for each candidate you calculate the minimum distance to all previous means. Then you select the example with the maximum minimum-distance. Run $k$-means once using this procedure.

Having done this, for each dataset you have 11 results. Plot a bar graph for CS and NMI showing these 11 results. Try to structure your plot so that any correlation between CS and NMI would be visible.

Both CS and NMI can be used as quality criteria for the clustering (except that in a real application we will not have access to labels and hence cannot calculate NMI). Write a short report on the results: Are CS and NMI stable across multiple random initializations? Are their quality judgements in agreement? How does the smart initialization fare relative to the best random initialization? and to the best that we would have chosen if using CS as a criterion? What other observations can you make from the results?
2.3 Selecting $k$

Here we attempt to check the “knee criterion” for selecting the number of clusters. In particular, for each dataset run $k$-means with $k = \{2, \ldots, 22\}$. To avoid the variability observed in the previous part, for each value of $k$ run the algorithm 10 times and pick the clustering result with the smallest CS from these runs. Now plot CS as a function of $k$.

Write a short report on the results: is there “visual evidence” suggesting that CS can be used as a criterion for selecting $k$? What can you conclude from the relative performance on the artificial datasets? Are the results consistent across the datasets? if not, what observations can you make from the results?

3 Programming Language and use of Libraries/Modules

You may write your program in any language as long as your code runs on our server homework.eecs.tufts.edu.

You are of course allowed to use basic I/O, math library, randomization and other basic language facilities. But you should write your own code for all the portions described above.

4 Submitting your assignment

4.1 What to submit

You should submit the following items both electronically and in hardcopy:

- *All your code* for data processing, learning algorithms, experiments, test programs, etc. Please write clear code and document it as needed.
  
  Please make sure that your code runs on homework.eecs.tufts.edu. Please include a README file with instructions how to compile and run your code to reproduce the results of experiments. If this is nontrivial please include a script to compile and run your code.
  
  Your submitted code should assume that the data files are in the same directory as the program. There is no need to submit the original data files. There is no need to submit any intermediate files or results. These should be generated when your program is run.

- A short report with the results and plots as requested and a discussion with your observations from these plots. Please make sure that your discussion responds to questions posed in the assignment text.

4.2 When and How to Submit

- Please submit electronically using provide by 12:00 noon of the due date. Put all the files mentioned in the previous subsection into a zip or tar archive (no RAR please). For example call it myfile.zip. Then submit using provide comp135 pp3 myfile.zip.

- Please submit a printed copy (of everything) by the beginning of class (4:30pm).

4.3 Grading

Your assignment will be graded based on the code, its clarity, documentation and correctness, the presentation of the results/plots, and their discussion.