Project 2: Getting It Together

Due: Nov 1 by 11:59PM

Instructions

For this project, you and your group will be designing and implementing a system in C or C++, that reads in a series of data points, clusters them using unsupervised clustering techniques, and then analyzes the effectiveness of the clustering. For clustering, you will be using k-means clustering and centroid-linkage hierarchical clustering. For analysis you will be calculating the Dunn index across all clusters, along with gathering a few statistics about the clusters themselves.

Additionally, sample input files can instead be found on the CSE machines at `/home/jeh0289/public/csce2110/fa19/proj2/` You can `cd` into that directory and copy input the files from there.

Further, you will need to utilize the GitLab code repository located at https://apollo.cse.unt.edu/ . You may access it from the website, the terminal, or an IDE of your choice.

Also, as a reminder, all of the work for this project must be the sole product of the group. You may not share code with other groups, or download solutions off the internet, as doing so will be considered cheating. If you are struggling with a concept or implementation, you are encouraged to contact the instructor or your TA's for aid.

Requirements

This assignment has two parts: a report portion and an implementation portion.

Report

For the report portion, the group must generate documentation, in PDF format, describing your system. The purpose of this is for you to explain not just what your system is doing, and how it is doing it, but why. You will need to justify your design decisions in a concise, informative manner. Justifications such as "I did this because it was easy" are not sufficient, as you should actually explain why a particular data structure or algorithm was more efficient, effective, or optimal. Additionally, commented code, while sometimes helpful in small examples, is not a sufficient explanation in and of itself. Your explanations and justifications are expected to be presented in prose and in paragraph format, i.e. not bulleted lists. Further, part of the evaluation of your design document is the apparent amount of thought and effort that went into its creation.

This document should begin with an overall description of the project and then be divided into sections based on major functionality components, as described later. Additionally, the document must follow the provided template.

In the first part, in one or two paragraphs the group should describe the project in their own words and give a brief overview of the problem. Do not copy and paste this from the instruction document.

For each major functionality component, in two or more good quality paragraphs the accountable student should describe the data structures and functionality of their component. What, if any, objects or structs did you create to store data? How did you organize and manage them? What types of formal data structures did you make use of (trees, graphs, arrays, hashes, etc)? How is data moved and transformed? How is it read in? How is it output? What are the various major functions you constructed and how do they work? Be sure to describe your component in detail, and provide sufficient justification of your methodology. You might also consider including diagrams to more easily visualize how all of the pieces fit together.

Implementation

Your program must provide the following functionality and adhere to the following constraints:

- Your int main() must be in its own .c/.cpp file
- All .c/.cpp files, except your main.c/main.cpp, must have associated header files. You may not #include a .c/.cpp file, only header files
- Allow the user to input the configuration file
    - Do **NOT** hardcode the filename into your program
    - The first line will specify the name of the points file to be used
    - The second line will specify the minimum number of clusters to attempt to cluster the points into when using k-means clustering
    - The third line will specify the maximum number of clusters to attempt to cluster the points into when using k-means clustering
    - All other lines will indicate which points to initially place the centroids when performing k-means clustering. Do **NOT** randomly assign centroids when beginning k-means clustering
- The points file will be specified by the configuration file
    - Each line will contain the id of the point, the X position of the point, and the Y position of the point in that order. X,Y coordinate pairs should be stored as some form of floating point value
    - Each individual point should be stored as either a struct or object, and should contain at least their ID, their X coordinate, and their Y coordinate. Additional relevant data may be stored as desired
- Your system should perform the following operations:
    - Read in and store the relevant information from the configuration file and output the total number of k values that will need to be tested when performing k-means clustering
    - Read in and store all points and output the total number of points
    - Perform k-means clustering for all values of k within the range specified by the configuration file
        - Clusters must be labeled starting at 1 and incrementing by 1 for each successive label
        - k-means clustering requires the repeated application of two steps: assignment of points to clusters; updating the centroid of those clusters based on assignments
        - When assigning points to clusters, each point should be assigned to the closest cluster using Euclidian distance. In the case of a tie, assign to the cluster with the smaller id
        - When updating the centroid of a cluster, this should be the placed at the smallest distance between all points currently assigned to that cluster, i.e. the center of the cluster
        - Clustering should continue until no points change clusters between two successive assignments
        - Once clustering has been completed for a value of k, the id of each cluster along with the total number of points in that cluster and the id of the point closest to that cluster's centroid should be output. The Dunn index for that value of k should also be output
    - Perform centroid-linkage hierarchical clustering
        - Initially, each point must be in its own cluster and clusters must be labeled starting at 1 and incrementing by 1 for each successive label. This represents level 0 in the hierarchy
        - At each level, the closest two clusters should be merged. Distance is determined by calculating the Euclidian distance between the centroids of two clusters. In the case of a tie, merge the two clusters with the smaller ids

        - When merging two clusters, the smaller id should be used when labeling the new cluster, and the new cluster should be added to the new level along with all of the clusters that were not merged that level
        - New levels in the hierarchy should be generated until a level contains only a single cluster
        - For every ten levels of the hierarchy, including level 0 and the final level containing only a single cluster, the level number and the total number of clusters in that level should be output. The Dunn index should be calculated and output every ten levels, except level 0 and the final level, after less than half of the total clusters remain (i.e. over half of the points have been merged)
    - Analyze the clusters from each clustering algorithm using the Dunn index
        - The Dunn index is the ratio between inter-cluster distance and intra-cluster distance. This can be calculated by determining the largest cluster diameter, i.e. the largest distance between any two points inside of a single cluster, and the smallest cluster distance, i.e. the smallest distance between two points in different clusters, and dividing the smallest cluster distance by the largest cluster diameter. Ignore clusters that contain only a single point when calculating the index. Ideally, we want tight clusters that are far away from each other, which results in larger values for the Dunn index
        - A small summary of the performance of each clustering algorithm should be provided at the end of the program stating:
            - The value of k, when performing k-means clustering, that provided the greatest Dunn index, and its Dunn index
            - The number of clusters, when performing hierarchical centroid-linkage clustering, that provided the greatest Dunn index, and its Dunn index
- See the example output files for formatting of each of the outputs
- Each major functionality component must be constructed in some function that is declared and defined outside of your main.c/main.cpp . Remember, function declarations must be stored in a header file, while definitions must be stored in a .c/.cpp file. You may have additional functions that support your major functionality component function.
- Your code must be well commented.
- Please do not commit the example input and output files to the remote server as that may cause a space issue on the server.
- You must provide a .txt format README file which includes:
    - The names of all group members
    - Instructions on how to compile your program
    - Instructions on how to run your program
    - An indication on whether you implemented the bonus or not. By default, the TA's will assume you did not attempt the bonus unless you indicate otherwise.
- Additionally, you may write a makefile, but please specify if it needs any additional flags to function properly
- Each student should be using GitLab during the development of their code. This means that each group member must have their own branch in the repository and should be making regular commits to it during development. These branches must eventually be merged into the master branch once development of their components is completed. Additionally, students should be creating meaningful commit messages. "Fixed bug" or "New code" are not meaningful, so try to be more specific about what was fixed or what was added.
- Each student must be accountable for one or more major functionality components, and may not swap after they sign up for a component barring an exceptional circumstance. Failure to be accountable for any major functionality component will result in a 0 for the coding portion of the project. Keep in mind that some components build on others, so be careful about who takes ownership of which pieces and manage your time to

avoid a crunch near the due date. Also, the group should strive to balance the work across all members. The major functionality components are:

1. Reading in the configuration file, configuring the system properly for the requested clustering, reading in and storing the data points, and providing requested output
2. Performing k-means clustering for each value of k in the range, and providing requested output
3. Performing centroid-linkage hierarchical clustering, and providing requested output
4. Calculating Dunn's Index for k-means clustering, single-linkage hierarchical clustering, complete-linkage hierarchical clustering, performing the appropriate analysis, providing requested output, and generating the summary

Project Submission

At least one group member must submit a .zip file containing the following:

1. All files necessary to compile and run your program (Please do not include other files that are not required for compilation or execution)
2. A .txt format README file explaining how to compile and run your program
3. Your report in PDF format

Rubric

The entire assignment is worth 100 points, and each student will receive a single grade with respect to both the entire project and to the portions they were individually responsible for. The breakdown of those points is as follows.

- 30 points: Report
- 15 points: Correct overall project functionality
- 40 points: Your major functionality component(s) satisfy requirements
- 5 points: Adequate comments
- 10 points: Adequate repository commits

If your code fails to compile on the CSE machines you will not receive credit for the code portion of the assignment. I recommend not making changes to your code without checking for compilation before you submit.

Bonus

For 10 bonus points, you will need to additionally implement k-medoid clustering. This is similar to k-means clustering, except that instead of using some (X,Y) coordinate pair for the centroid of a cluster, you will use one of the cluster's points to represent its centroid. In this way, whenever you attempt to shift the centroid during the update step, you will potentially move the centroid to another point's location. This clustering should be performed for all values of k within the same range as performed by k-means cluster. Additionally, once clustering has been completed for a value of k, the id of each cluster along with the total number of points in that cluster and the id of the point representing that cluster's centroid should be output. The Dunn index for that value of k should also be output, and similar to the results for k-means clustering, the results for k-medoid clustering should be included in the summary.