CSC 350/500 – Project 3: Clustering

**Due Date:** Friday, March 28, 2025

# Introduction

In this project, you will implement the k-Means Clustering algorithm and investigate its result on several pre-made data sets. I supply starter code in Java and Python; you can choose which language to use. The starter code takes various command-line arguments, including an input file and a destination output file. Java Documentation for included code can be found in java/doc/KMeansClusterer.html. (Python use is virtually identical to Java.) Descriptions of the input and output file formats are on Page 6 of this document, but code is provided to read and write to these files.

This is a group project. A group can have up to 5 students. A group must be either all undergraduate students or all graduate students, no mixing.

## File to Edit

You should only need to edit KMeansClusterer.java or KMeansClusterer.py. Please *do not* change the names or headers of any of the provided functions or classes. Modifying code that you were instructed not to modify is grounds for losing credit on this assignment.

## Getting Help

You are not alone! If you find yourself stuck on something, contact me for help sooner rather than later. I want these projects to be rewarding and instructional, not frustrating or demoralizing. But I do not know how or when to help unless you ask.

## What to Submit

Zip all project files for your chosen language, as well your answers for written questions and the attached p3-results.xlsx spreadsheet. One person per group should submit on Blackboard by the due date. Any submission after the due date is subject to the late penalties described in the syllabus.

## How You Are Graded

Your grade is determined based on the implementation of the algorithms, answers to written questions, and the data in your spreadsheet. The grading rubric for this assignment is on the final page of this document.

# Assignment Questions

For all questions below, commands and function names refer to Java. Python commands and names should be similar.

## Exercise 1 (15 points) – *k*-Means with given *k*

For all exercises, assume the [Standard k-Means Clustering Algorithm](https://en.wikipedia.org/wiki/K-means_clustering#Standard_algorithm_(naive_k-means)) with [Forgy initialization](https://en.wikipedia.org/wiki/K-means_clustering#Initialization_methods).

Implement the kMeansCluster() function in KMeansClusterer.java so that it computes *k* clusters. Note that you will also need to implement three other functions as part of this: getWCSS(), assignNewClusters(), and computeNewCentroids().

Your command to run this program would take the form:  
java KMeansClusterer -k {k} -in {input file} -out {output file}

Example:  
java KMeansClusterer -k 2 -in bullseye2.dat -out   
bullseye2-out.dat

See Page 5 of this assignment for guidance on how to visualize the 2D input and output files.

### Questions

1a. What are the assumptions that the k-means algorithm makes of the input data?

1b. Each input file has the original generating clusters in the filename before the “.dat” extension. For example, bullseye2.dat was generated with 2 clusters. For each input file provided, run *k*-means 3 times with the k value indicated by the filename.

Record the WCSS of each run in the Columns B-D of the given p3-results.xlsx Excel file. Be sure to visualize the results for each run. For which files does *k*-means clustering appear to succeed always/sometimes/never in your sample of 3 runs?

1c. For those problems where *k*-means appears to always fail, which assumptions (if any) of the *k*-means algorithm are violated?

1d. Is it possible for *k*-means to fail if no assumptions are violated? Why or why not?

======================================================

## Exercise 2 (12 points) – Iterated *k*-Means with given *k*

Create an augmented version of the first program that, for iter > 1, performs iter independent runs of the *k*-means algorithm and outputs only the clustering result with the lowest WCSS value.

This augmented version can be a different class (in Java) or script (in Python) – you may want to start from a copy of your working version of Exercise 1.

Your command to run this program would take the form:  
java KMeansClusterer -k {k} -iter {i} -in {input file} -out {output file}

Example:  
java KMeansClusterer -k 2 -iter 10 -in bullseye2.dat -out  
bullseye2-out.dat

### Questions

Use 10 iterations to answer the following questions. Be sure to visualize your results to help answer the questions.

2a. Why should iterated use of the *k*-means algorithm help in some cases with the quality of output clusters?

2b. For each input file provided, run iterated *k*-means one time each. Enter the resulting WCSS in Column H of the Excel sheet. For which files does *k*-means clustering appear to succeed in your iterated sample run?

2c. For which types of clustering problems does/doesn’t this iterated approach help?

===========================================================

## Exercise 3 (15 points) – *k*-Means using the Gap statistic to choose *k*

Create an augmented version of the second program that has all of the previous functionality, but additionally chooses the best *k* value in a range of *k* values using a simplified gap statistic (defined below). The range of *k* values is specified via variables *kMin* and *kMax*.

This augmented version can be a different class (in Java) or script (in Python) – you may want to start from a copy of your working version of Exercise 2.

Your command to run this program would take the form:  
java KMeansClusterer -kmin {min k} -kmax {max k} -iter {i} -in {input file} -out {output file}

Example:  
java KMeansClusterer -kmin 2 -kmax 9 -iter 10 -in bullseye2.dat -out bullseye2-out.dat

One very difficult problem concerns automatically determining the number of clusters in a data set. There are many suggested techniques (if you’re interested: [Examples from Wikipedia](https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set), [Stack Overflow Example in R](https://stackoverflow.com/questions/15376075/cluster-analysis-in-r-determine-the-optimal-number-of-clusters/15376462#15376462)).

Perhaps the simplest technique is to plot the lowest WCSS found for each k and look for an “elbow” – that is, a k value after which reductions in WCSS are not significant as k increases. This approach is referred to as the [Elbow Method](https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set#Elbow_method).

The difficulty with the Elbow Method is that the most significant elbow bend is not always discernible from this graph. Alternately, we would prefer a way to compare the gap between the expected WCSS on uniformly-distributed random data and the WCSS obtained on the given data see which k yields the most significant gap. This measure is known as the Gap Statistic (if you’re interested, here’s [the original paper on the gap statistic](https://hastie.su.domains/Papers/gap.pdf)).

Here is a simple algorithm for the Gap Statistic:

* For each *k* from *kMin* to *kMax* (inclusive):
  + Run *iter* iterations of k-means (just like Exercise 2). Calculate the log of the minimum WCSS from those runs. Call this value *logMinWCSS*.
  + Generate 100 data sets of an equal number of data points randomly distributed within the [minimum, maximum] range for each dimension of the given data. Run 1 iteration of *k*-means on each of those 100 data sets. Compute the average of the log of the WCSS of the hundred data sets. Call this value *avgRandWCSS*
  + Compute the gap statistic for this *k* value, where *GapK = (avgRandWCSS – logMinWCSS)*.
* Output the *k* (and the associated clusters and centroids) with the maximum *GapK*.

### Questions

3a. For each input file provided, run your program 8 times with *kMin*=2 and *kMax*=10 and *iter*=10. From each run, record the number of clusters *k* that yielded the maximum gap statistic in Columns J-Q of the spreadsheet.

3b. For which data set(s) does this technique consistently succeed in discerning the correct number of clusters?

3c. For which data set(s) does the discerned number of clusters vary from run to run?

3d. For which data set(s) does this technique consistently return the incorrectnumber of clusters? What do you observe about the nature of the data in these case(s)?

===========================================================

# Visualizing the Input and Output

You can visualize the input and output using the two provided Python scripts (even if you are a Java user, you must use the Python scripts), data2d.py (for input) and cluster2d.py (for output). Both scripts take an input or output file as a command-line argument.

Example usage of the two scripts. (Mac/Unix users should say python3 instead of python):

python data2d.py bullseye2.dat

python cluster2d.py bullseye2-out.dat

Before running the scripts, you must have the matplotlib library installed. If you’ve never installed a Python library, follow the instructions below to use pip (Package Installer for Python) to install the library. These instructions assume that Python 3 is already installed on your machine.

1. Add pip’s directory to your PATH. On my Windows computer, this is the directory: C:\Users\myusername\AppData\Local\Programs\Python\Python311\Scripts

Yours will have a different username, and the Python311 may be different if you’re using a different version. Mac/Unix users: See me if you can’t track down pip’s location or if the pip command doesn’t already work.

1. From the command line/Terminal, run the following command to install matplotlib. (Mac/Unix users probably need to say pip3 instead of pip.)

pip install matplotlib

There are dependencies for matplotlib, so don’t be surprised if several other libraries are installed.

# Data Set Input Format

For all data files, the first two lines of input data are:

% {number1} dimensions

% {number2} points

After these lines, the next {number2} lines contain {number1} space-separated numbers, where {number2} and {number1} are positive integers.

# Clustered Output Format

For all data files, the first four lines of input data are:

% {number1} dimensions

% {number2} points

% {number3} clusters/centroids

% {number4} within-cluster sum of squares

After these lines, the next {number3} lines contain the coordinates of the centroids, in ascending order by centroid cluster number (starting from zero). These lines contain the cluster number and the coordinates of the centroid.

The next {number2} lines contain the same datapoints as the input file, in the same order, with the cluster they were assigned to *preceding* the coordinates.

# Rubric

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Question** | **3 points** | **2 points** | **1 point** | **0 points** |
| Exercise 1 – Code | Algorithm is (nearly) correct and behaves as expected on example inputs. | General idea behind algorithm is correct, but there are isolated situations leading to flawed output or crashes. | Algorithm is severely flawed, possibly crashing in some cases. | Does not compile or consistently crashes due to runtime errors. |
| Question 1a | Assumptions match those discussed in class. | Assumptions overlap some with those discussed in class. | Assumptions do not match those discussed in class. | Off-topic or no response. |
| Question 1b | Excel sheet properly filled in and correct response to questions. | Some data missing or aspects of response missing, or disagreement between those. | Limited response or significant missing data. | No data or response. |
| Question 1c | Answer is correct and consistent with results | Answer is partially correct and consistent with results | Answer is flawed or inconsistent with results | No response |
| Question 1d | Mostly correct | Partially correct | Flawed response | No response |
| Exercise 2 – Code | Algorithm is (nearly) correct and behaves as expected on example inputs. | General idea behind algorithm is correct, but there are isolated situations leading to flawed output or crashes. | Algorithm is severely flawed, possibly crashing in some cases. | Does not compile or consistently crashes due to runtime errors. |
| Question 2a | Mostly correct | Partially correct | Flawed response | No response |
| Question 2b | Excel sheet properly filled in and correct response to questions. | Some data missing or aspects of response missing, or disagreement between those. | Limited response or significant missing data. | No response |
| Question 2c | Answer is correct and consistent with results | Answer is partially correct and consistent with results | Answer is flawed or inconsistent with results | No response |
| Exercise 3 – Code | Algorithm is (nearly) correct and behaves as expected on example inputs. | General idea behind algorithm is correct, but there are isolated situations leading to flawed output or crashes. | Algorithm is severely flawed, possibly crashing in some cases. | Does not compile or consistently crashes due to runtime errors. |
| Question 3a | Excel sheet properly filled. | Some data missing. | Significant data missing. | No data |
| Question 3b | Answer is correct and consistent with results | Answer is partially correct and consistent with results | Answer is flawed or inconsistent with results | No response |
| Question 3c | Answer is correct and consistent with results | Answer is partially correct and consistent with results | Answer is flawed or inconsistent with results | No response |
| Question 3d | Answer is correct and consistent with results | Answer is partially correct and consistent with results | Answer is flawed or inconsistent with results | No response |