Ira A. Fulton Schools of Engineering

CSE 575 Statistical Machine Learning

Portfolio Report

Arizona State University

Hannah O. Ajoge

ASU ID: 1217581417

Date Due: May 09, 2020
Date Submitted: May 08, 2020

Table of Contents

Introduction

The aim of data clustering is to divide a set of objects into groups of objects such that objects in the same group/cluster are more similar to each other (according to measured or perceived intrinsic characteristics) than to objects from other group/clusters [1-2]. Clustering analysis is an unsupervised learning process, one of the fundamental methods of discovering and understanding underlying patterns embodied in data and is often used as a preliminary step for data analytics [1-2]. Clustering analysis is widely utilized by many disciplines that include image bioinformatics, financial analysis, segmentation, text mining and wireless sensor networks [2]. The three main purposes of data clustering are understanding the: (1) underlying structure of the data - gain insight into data, generate hypotheses, detect anomalies, and identify salient features; (2) natural classification of the data - identify the degree of similarity among forms or organisms; and (3) compression of data - as a method for organizing the data and summarizing data through cluster prototypes [3]. Among the many clustering algorithms that have been developed in the past sixty years, the k-means algorithm is one of the oldest and most popular clustering algorithms [1]. This is because k-means clustering is simple, efficient and easy to implement [2].

K-means algorithm (adapted from Jain 2010 [3])

Let $X=\{x_i\}$, $i=1,\ldots,n$ be the set of n d-dimensional points to be clustered into a set of clusters, $C=\{c_k, k=1,\ldots,K\}$. K-means algorithm finds a partition such that the squared error between the empirical mean of a cluster and the points in the cluster is minimized. Let μ_k be the mean of cluster c_k. The squared error between μ_k and the points in cluster c_k is defined as:

$$J(c_k) = \sum_{x_i \in c_k} ||x_i - \mu_k||^2$$

The goal of K-means is to minimize the sum of the squared error over all K clusters,

$$J(C) = \sum_{k=1}^{K} \sum_{x_i \in c_k} ||x_i - \mu_k||^2$$

Despite the popularity of k-means clustering, it suffers from a number of limitations, such as initialization sensitivity, susceptibility to noise and vulnerability to undesirable sample distributions [1-2]. To compensate for these limitations, the basic k-means algorithm has been extended in different ways [3]. One such extension is the k-mean ++ algorithm, which reduces the problem of sensitivity to initialization [4].

Solutions

In this project, I implemented two strategies of k-means algorithm - the basic k-means algorithm and the k-mean ++ algorithm (an extension of the basic k-means algorithm). The implementation was done from scratch in Python. I utilized the Jupyter notebook (lab) provided within Coursera platform for my coding.

Strategy 1:

I entered the four last digits of my student ID and with that generated my set of initial centroids. I first viewed the initial utilized the initial centroids locations on the data through scatter plot. From time to time I visualized my centroid and assigned clusters through colored scattered points.

Then I created a function “dist_toCentroid” that takes a set of centroids and a data frame with two columns (x and y). The function then generate a new set of centroids and a data frame. The generated data frame contains distances of each point from the centroids, a column known as closest which indicate the cluster that each point is assigned, and a column named color which contains letters respective to the assigned cluster (the letter can be used to assign colors to data points in scatter plots).

Then I produced while loop that ensures that the function “dist_toCentroid” is used to iteratively generate new centroids until the centroids don’t change (when no single data points change clusters anymore).

Finally I calculated the objective function, by calculating the squared errors for each cluster data point to the cluster’s centroid, and then summed up the squared errors. This sum of squared errors is the objective function or J(C).This process was executed for both k1 (K=3) and k2 (K =5).

Strategy 2:

For Strategy two I started with only the first centroid and then computed the remaining initial centroids by assigning data point that is furthest from the previous centroid(s). After obtaining the remaining initial centroids, I computed the final centroids and the objective function as in Strategy1.

Strategy 2b:

Since the assigned Ks for the two strategies are different – 3 and 5 (for strategy 1) and 4 and 6 (for strategy 2); it may not be plausible to say that the extension to the basic k-mean algorithm provided better result (lower objective function). To remove this implausibility, I implemented strategy 2 again which I now call strategy 2b. In strategy 2b, I reimplemented strategy 2 but: (1) utilized the same Ks as in strategy – 3 and 5; and (2) utilized the same first initial centroids as in strategy 1.

Results

Below are the resulting final centroids and objective functions of the two strategies.

Strategy 1:

K1 (K=3) Result
Final centroid: {1: [[7.2397511895844495, 2.4820826910731952]], 2: [[4.833753175392286, 7.316058236043574]], 3: [[3.2489642305948876, 2.5802769113756954]]}
Objective function: 1338.1059838

K2 (K=5) Result
Final centroid: {1: [[2.8749081274874264, 7.010822811156844]], 2: [[2.681986334188929, 2.094615867800809]], 3: [[7.556167822397726, 2.235167959857534]], 4:

[[5.257224105388598, 4.255186256593418]], 5: [[6.775311757464789, 8.115401936406947]]}

Objective function: 598.678798534

Strategy 2:
K1 (K=4) Result
Final centroid: {1: [[7.252626831256577, 2.4001582635520533]], 2: [[3.2285300905383707, 2.5240486292057867]], 3: [[6.6259253846324615, 7.576149167622678]], 4: [[2.9054774114449513, 6.905122763339948]]}

Objective function: 789.237972218

K2 (K=6) Result
Final centroid: {1: [[4.022935100912096, 3.870824728369343]], 2: [[7.756483249146484, 8.556689279063415]], 3: [[2.825447560760127, 1.6546753620598549]], 4: [[2.512257357886769, 7.057170033892222]], 5: [[7.477682316761515, 2.2990031525317582]], 6: [[5.464277356727894, 6.837713536435891]]}

Objective function: 484.04926087

Strategy 2b:
K1 (K=3) Result
Final centroid: {1: [[7.2397511895844495, 2.4820826910731952]], 2: [[4.830919584356354, 7.299599586723327]], 3: [[3.234890046359086, 2.5530321964002036]]}

Objective function: 1338.10760165

K2 (K=5) Result
Final centroid: {1: [[5.339072124427115, 4.465511754739297]], 2: [[7.299749694765005, 8.413318379491713]], 3: [[2.681986334188929, 2.094615867800809]], 4: [[3.1507276114491836, 7.121929064073764]], 5: [[7.556167822397726, 2.235167959857534]]}

Objective function: 592.528384259

Conclusion:
In this project I showed that:

1. The extension to the basic k-mean algorithm (k-mean ++) is capable of providing slightly better clustering result or in other words lesser objective function. This can be seen when comparing strategy 1 and strategy 2b, in which the K=5 for strategy 2b provided better result.
2. However the basic K-mean algorithm still performs quite similar to k-mean ++), as there was no difference in objective function when K was 3.
3. As expected, increase in K leads to better clustering result – lesser objective function.

Contributions

The project was not a group project, so I executed all the steps of the project by myself by following the instructions given by the course professor and the graduate student assistants. I also did all the write up by myself.

Lessons Learned

The new expertise I obtained through this project include ability to:

1. Understanding the concept of k-mean clustering algorithm, as well as it's extension - k-mean ++.
2. Implementing from scratch in Python, the algorithm necessary to solve a k-mean clustering problem.

References

[1] G. Gan, M. and K.-P. Ng, *"*k-means clustering with outlier removal,*"* *Pattern Recognition Letters*, vol. 90, pp. 8-14, 2017, http://www.sciencedirect.com/science/article/pii/S0167865517300740

[2] H. Xie, L. Zhang, C. P. Lim, Y. Yu, C. Liu, H. Liu and J. Walters, *"*Improving K-means clustering with enhanced Firefly Algorithms,*"* *Applied Soft Computing*, vol. 84, Article 105763, 2019, http://www.sciencedirect.com/science/article/pii/S1568494619305447

[3] A. K. Jain, *"*Data clustering: 50 years beyond K-means,*"* *Pattern Recognition Letters*, vol, 31, no 8, pp. 651-666, 2010, http://www.sciencedirect.com/science/article/pii/S0167865509002323

[4] Lecture videos