Final Project 532

Ahmad Rakin, Azraf
rakin@wisc.edu

Waghray, Kunal
kwaghray@wisc.edu

Piedra, Sebastian
piedra@wisc.edu

Spring, 2020

1 Abstract and Background

1.1 Introduction

K-means is a paramount unsupervised learning algorithm used across a multitude of disciplines. While it has multiple benefits, it suffers from some short comings that can be extremely detrimental to the performance of the algorithm. In practice, it is not uncommon to work with massive amounts of data. In such cases, issues like the speed of convergence of an algorithm are of the utmost importance. Similarly, the robustness of the results is important, a concern that is severely affected by poor convergences to local minimal. The k-means++ algorithm is an improvement upon the k-means algorithm that aims to improve the speed of convergence, can increase the robustness of our solutions, and solves the issue of arbitrarily bad solutions present in k-means. In practice, solving those issues could translate to a major improvement in the performance of our solutions and allow us to tackle previously inaccessible dimensions of data. In this project, we would introduce the k-means++ algorithm using mathematical foundations, toy examples, and a real life application of the technique in the stock market.

1.2 Learning Objectives

1. To understand some limitations of conventional K-Means algorithm, and of the importance of proper initialization of cluster centers.
2. To develop a working knowledge of the k-means++ algorithm
3. To understand the math behind the k-means++ and how it improves the algorithm with respect to traditional k-means
4. To have a clear understanding of the motivation for this algorithm, and a knowledge of some potential applications

5. To analyze the processes behind feature engineering in building machine learning models

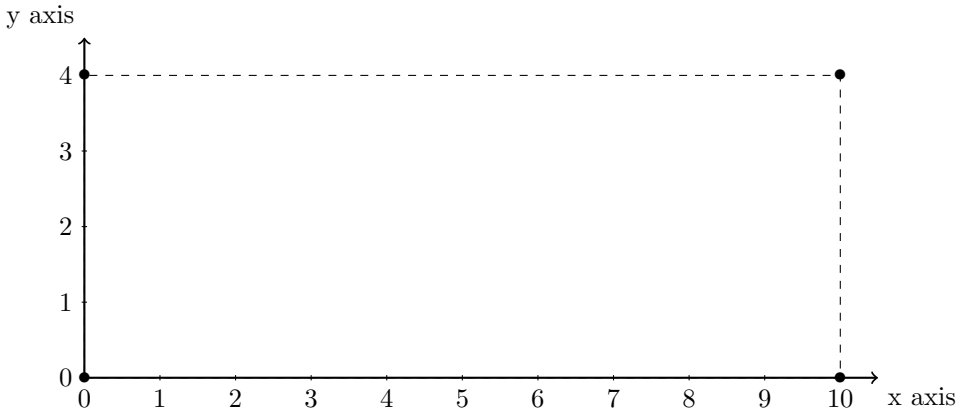
6. To study the application of K-Means++ algorithm in the prediction of securities in the stock market

1.3 Motivation

We start with a simple example to show the shortcomings of the traditional k-means solution. Notably, we attempt to cluster four points in $\mathcal{R}^2$ - (0, 0), (10, 0), (10, 4), (0, 4) - the vertices of the rectangle below. Two clusters (k = 2) are used.

(i) Start with initial clusters (0, 2) and (10, 2). What are the final clusters (plot them)? What is the sum of squared error?

(ii) Start with initial clusters (5, 0) and (5, 10). What are the final clusters converged to (plot them)? What is the sum of squared error and why is this a problem?



1.4 Problems with traditional K-means

We now state some of the most common problems with traditional k-means. We hope this serves as a motivation for k-means++ and conveys the benefits it brings to the table relative to the traditional approach.Yet, it also shows issues to keep in mind in practice that are **not** solved by k-means++.

1. Slower convergence. K-means initializes the starting clusters randomly. Thus, the overall movement that it might need tends to be greater to that required by k-means++, since k-means++ already starts with better clusters that are likely to need less correction.

2. K-means assumes a spherical distribution of the data

3. In general, traditional k-means clustering can be found to be arbitrarily bad, as shown in activity 1.

4. Sensitivity to outliers. The way the error is defined makes these algorithms very sensitive to outlier data. Thus, outliers could end up dragging our clusters outside the actual data clusters, or even getting their own individual clusters. In practice, some preprocessing should be applied before using these algorithms.

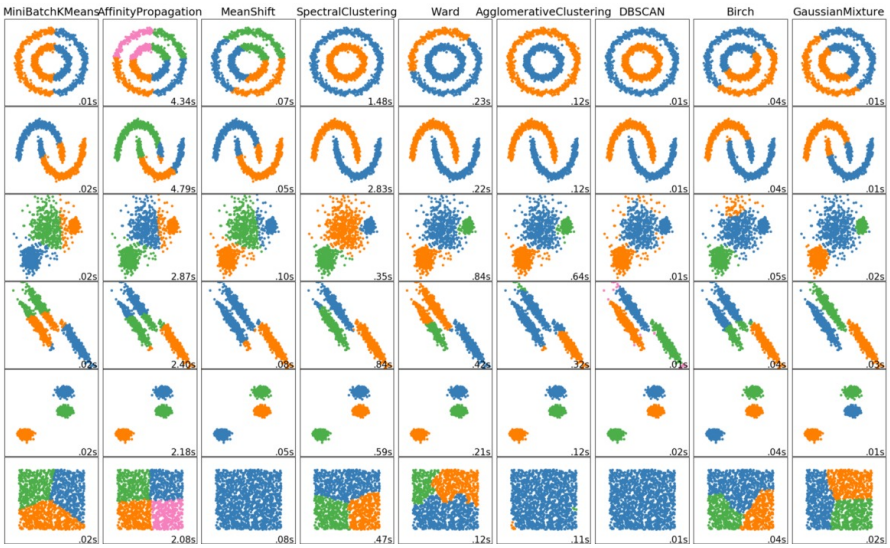


Figure 1: Visual demonstration of problem number 2. K-means (first column) does not find community structures but clusters the data assuming a spherical distribution. Notice that, however, in the first row above we could add a third dimension whose value is the radius from the center and we would obtain correct clustering. Nonetheless, with such a third dimension we would observe the aforementioned spherical distribution

From the aforementioned problems, two can be improved if we use k-means++. Namely, we could increase the speed of convergence and prevent the algorithm to be arbitrarily bad ([1], [3]). However, problems [2] and [4] are present in both k-means and k-means++.

It is also important to consider in this section the existence of some ethical issues that can arise through either of our algorithms. Different issues of real life discrimination are often reflected in the data acquired throughout time. For examples, unfortunate cases of systematic discrimination on the basis of gender, race, religion, or sexual orientation are often reflected in data sets. Then, if the developer of tools such as k-means++ is not careful, the resulting clusters can reflect this biases and perpetuate this discriminating behaviors. For instance, if

existing gender pay gap is not taken into account when we use our algorithms, an algorithm could incorrectly suggest that a man should earn more than a woman. While this is evidently wrong, previous biased data could give such results. In conclusions, both the developers and users of this tool, as well as many other data science tools, should be aware of existing discrimination issues and account for them in order to stop the further propagation of discrimination.

1.5 History and Algorithm

The *k_means++* improvement was proposed in 2007, while the original *k_means* algorithm goes back to the 1950'-1960's. The *k_means++* algorithm was proposed by the computer scientists Sergei Vassilvitskii and David Arthur. The algorithm, simply adds a short and fast preprocessing step to the traditional *k_means* algorithm. This initial step, allows us to choose better starting clusters, instead of choosing the clusters uniformly at random as in the original algorithm. The *k_means++* algorithm not only helps in problem number [3], by improving the selection of the initial k clusters, but also improves the speed of convergence, since selecting better starting clusters leads to a faster convergence, which is the issue mentioned in problem number [1].

Definition: *k_means++* algorithm
For k clusters, do the following:

1. Choose the first cluster center randomly, as you would in traditional *k_means*. That is, choose it to be one of the data points at random
2. For all the data points, compute a weighted probability, where the weight is directly proportional to the distance squared between the data point and the closest cluster to it.
3. Using the weighted probability from (2), select one of the data points as your new clusters.
4. Repeat (2) and (3) until you have the required number of clusters.
5. Run *k_means* using the clusters obtained in the previous steps as your starting clusters.

Mathematically, we can understand the problem in the following way. Let $a \in A$ be your n data points, where $A \subset \mathcal{R}^{mxn}$. Also, let c_i be one of the k starting clusters we want to initialize. As mentioned, above c_1 is chosen at random. Then, the new center c_i for $2 \leq i \leq k$, is chosen to be a data point $a \in A$, with probability $\rho_i(a) = \frac{D_i(a)^2}{\Sigma_{x \in A} D_i(a)^2}$ where $D_i(a) = \min ||a - c_j||_2$ for $1 \leq j \leq i-1$.

2 Activity

1. In the motivation section we presented an example where you have 4 data points, $a \in A$, such that $a_1 = (0,0), a_2 = (0,4), a_3 = (10,0), a_4 = (10,4)$. Suppose you are using traditional k-means, use the code in Part 1 to understand what happens as the rectangle grows. That is, when $a_3 = (x,0), a_4 = (x,4)$ for $x > 10$. Try these values for width: 2, 10, 50, and calculate the errors associated with each

2. In part 1 we saw the conventional k-Means algorithm with poor initialization of clusters. Now we will look at k-Means++, a modified version of the algorithm with an initialization step.

   (a) Before we start, let's finish writing our k-Means++ code. You just need to fill in the line "next centroid = " in the initialize function. This function is used to select the initial cluster centers for k-Means++. The line you need to fill out selects the data point with the maximum distance from its closest cluster as our next centroid. *Hint: We selected a specific point in the data array. Information from the dist array is used to determine what index this is.*

   (b) Now comment out the lines for the optimized initialization, and try the same values as in question 1. What are the values of the centroids and the respective errors?

   (c) Do you notice anything different about the cluster centers and the assignment of the points as the rectangle gets wider? Compare it to the conventional kMeans algorithm.

3. Here we run k-Means on data that naturally belongs in 3 clusters. The code provided creates data in two dimensions distributed around three data points. Data distributed around the same data point is indicated by a different color.

   (a) Run the code for part 3a on jupyter notebooks. This runs k-Means with the initial clusters (3, 3.5), (6, -5), and (8, -5). Plotted are the final clusters the algorithm finds. Are the final clusters found ideal for the data?

   (b) Run the code in section 3b on jupyter notebooks. Here we run k-Means++ on the same data. The first three plots with the title "Select nth centroid" show the algorithm choosing the initial cluster centers using the k-means++ function *initialize*. It plots each initial centroid as it selects them as well as the previously found centroid. The last plot shows the final clusters found after running K-Means on these initial clusters. How do the results compare to part 3a?

4. The complementary code for this question would provide you a matrix A that holds some data points, and a random starting c_1. Calculate $p_2(a)$

for all $a \in A$. That is, get the probabilities for any other data point to be c_2. Complete the code in the activity. What conclusion can you make about c_2 relative to c_1?

3 Bonus Activity: Stock Market

5. For this part, we are going to show the performance of k-means and k-means++ on stock market data. Everyday, billions of dollars are traded in the stock market. Thus, quant firms use advanced computer science techniques to acquire an edge in this business. We compiled and cleaned data for 103 companies (including removing outliers and scaling the axis). For each company, we use 3 indicators popular amongst traders as features: Relative Strength Index, Volume, and price/earning ratios. Run the first part of section 5 in jupyter notebooks too see the data.

   (a) Run part 5a to see the data with clusters found using k-Means++ and the time required for it to run.

   (b) Run part 5b to see the data with clusters found using k-Means (random initialization) and time required for it to run. Run it a few times to see if anything changes.

   (c) What can you conclude from these examples? In what cases is worth it to run k-means++ and in what cases is not?

4 Solutions

1. (a) The error for k-means is $\Sigma_{i=1}^{k}\Sigma_{a\in A}||x - c_i||_2^2$, using our previous notation. Thus, the required code lines are:

```
error_1 = [np.linalg.norm(A_1[:,d] - centroids[:,0]) for d in range(2)]

error_2 =  [np.linalg.norm(A_2[:,d] - centroids[:,1]) for d in range(2)]
```

   (b) As the width increases in this example the distance from the assigned data points to the corresponding centroids directly increases. Consequently, the error can increase indefinitely.

   (c) Width = 2, Error = 1
   Width = 10, Error = 10
   Width = 50, Error = 25

2. (a) For initialize funtion

```
next_centroid = data[np.argmax(dist), :]
```

   (b) Width = 2: Centroids = (1,0), (1,4) Error (with initialization) = 1
   Width = 10, Centroids = (0,2) (10,2) Error (with initialization) = 2
   Width = 50, Centroids = (0,2) (50,2) Error (with initialization) = 2
   Width = 1000, Centroids=(0,2) (1000,2)Error (with initialization) = 2

(c) As you can see as the widths got bigger the conventional error of the kMeans algorithm increased indefinitely as in part 1. So the fact that the error remained constant for kMeans++ is a significant advantage.

(d)
```
error_1 = [np.linalg.norm(A_1[:,d] - centroids[:,0]) for d in range(2)]

error_2 =  [np.linalg.norm(A_2[:,d] - centroids[:,1]) for d in range(2)]
```

3. (a) Clearly, the clusters found are not ideal for this data. Two of the clusters lie within the blue data which should be in the same cluster. As a result, only one cluster is used to represent the green and red data which should be in its own cluster.

   (b) The initial clusters chosen already offer an improvement to part a, with each initial cluster approximately within each color of data. The final clusters are centered as desired.

4. The key of this question is to recall that $\rho_2(a) = \frac{D_2(a)^2}{\sum_{x \in A} D_2(a)^2}$. At this stage, there is only one other cluster, c_1. Thus, $D_2(a) = ||a - c_1||_2, \forall a \in A$. In this case, there are 5 data points a_i in A, c_1 is chosen to be one of those at random. The student should then, compute $p_2(a_i)$ for the 5 data points, including the one chosen by c_1. Evidently, $p_2(a_i = c_1)$ would have the smallest value. Finally, you must have noticed that c_2 is chosen to be the data point furthest away from c_1 as possible.

5. There are many conclusions that can be drawn from this example and it can depend on what random cluster is chosen for k-Means. By inspection, in this example both k-Means and K-Means++ end up with similar solution but the k-Means version varies slightly and sometimes fails to capture the green cluster. This inconsistency is clearly a disadvantage of k-Means. As a result of the time used to calculate the initial clusters k-Means++ takes a little longer than the normal k-Means. Thus, while k-Means++ can help ensure improve convergence speed, these advantages may not be clear in examples like these which are sufficiently simple and run for a fixed number of iterations. However, in cases where we have massive amounts of data, the time spent in the initialization could be balanced by the improvement in the speed of convergence, and therefore worth it.

5 Citations

1. Arthur D. and Vassilvitskii S. k-means++: *The Advantages of Careful Seeding* http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf

2. Wikipedia, k-means++, https://en.wikipedia.org/wiki/K-means

3. George Seif, The 5 Clustering Algorithms Data Scientists Need to Know, - https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

4. GeeksforGeeks, ML — K-means++ Algorithm, https://www.geeksforgeeks.org/ml-k-means-algorithm/

5. TradingView, Stock Screener - Search and Filter Stocks, https://www.tradingview.com/screener/ https://www.tradingview.com/screener/