ESE 545 - Project
Clustering

Tejas Srivastava

May 03, 2020

Preprocessing of Dataset

- The dataset was downloaded and read as a Pandas Dataframe. The dataframe had about 1865473 rows and 34 columns originally. Each row of the dataset corresponds to a movie and each column corresponds to a unique attribute of that movie. Some basic preprocessing steps were applied to the dataset in order to clean the data.
- We observed the mean values of the attributes startYear and runtimeMinutes was 2001.8942000232648 and 44.27949962288385 which were comparatively large as compared to the mean values of the other columns which were all less than 1, thus decided to perform min max normalization to obtain values between 0 to 1, so as to avoid the bias of these values in distance calculation. Also, we dropped the string based columns such as originalTitle, tconst,titleType in order to have measurable distance metric for the clustering algorithm.

Online Version of K Means Clustering

- We try to formalize the K means clustering problem as an optimization problem and use minibatch gradient descent to update the centers thus learning them in an online manner.
- We randomly intialize our cluster centroids for this implementation and iteratively improve over the solution until a local minima is reached.
- This method is certainly better than LLoyd's Algorithm since LLoyd's Algorithm requires k passes over the whole dataset with N points and has the complexity of the order O(knd), where k is the number of centroids, n is the number of datapoints in the dataset, and D is the dimension of each datapoint.
- The basic steps in implementing the k Means algorithm by using minin batch gradient descent were
  - Random selection of centroids
  - For each iteration, random selection of datapoints in the mini batch to be considered.
  - Further for each point in the mini batch, distance is calculated from all the centers.

- Note on Learning Rate η: We also maintain an array v, of size equal to the number of centroids or 'k', which is basically incremented for a centroid, when the centroid is the nearest centroid for a given point in mini batch among all the centroids. And for each point in the minibatch, this v value of the nearest centroid is used to set up the learning rate ets, which is taken as $\frac{1}{v[nearestcentroid]}$, thus for a centroid which is the nearest centroid to a lot of points in the mini batches, the value of v is large and thus the learning rate is small, and the centroid does not change much; whereas for a centroid which is not the nearest centroid to a lot of points, the eta value is large and thus changes a lot in order to reach convergence faster.
- Update the centroid with the minimum distance using the gradient descent update equation.

- Since the results of K means clustering are heavily influenced by the initialization of the centroids and the value of k chosen, we ran the algorithm for different values of k and tried to find the distance of each data point to the nearest centroid and averaged over all the points. Further, We have plotted the minimum, maximum and mean over all the clusters using a custom evaluate function we wrote.

- We used the following values of k to initialize the number of centroids [5, 10, 20, 30, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500] and based on these values of k, we calculate the centroids and compute the mentioned distance metrics.

- The results obtained for this are as shown below:

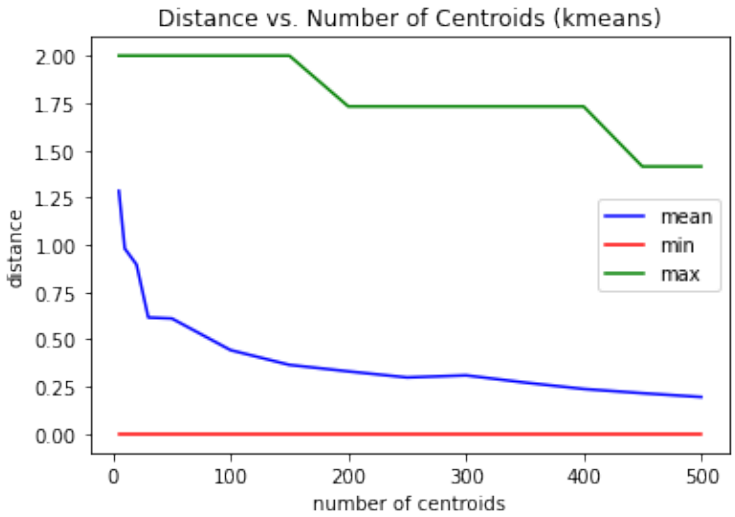

Figure 1: Min, Max and Average distances of datappoints from their nearest centroid for diff k

- We observe that as we increase the number of centroids for clustering i.e. k the calculated distances go down, and as the number of clusters increase, the sum of squared distances will tend to zero.

K Means++ initialization technique

- Since the optimization problem being solved here is highly convex, and many local minima exist, thus the initialization can dramatically influence the results. Thus, here we use a smart initialization technique i.e. K Means++, which is based on picking successive centroids from a frequency distribution which is proportional to the square of the distance between the datapoints and the nearest centroids. (D^2 sampling).

- The first centroid is initialized randomly by selecting one of the points from the existing datapoints.

- For the i^{th} centroid, the nearest already existing centroid is found out for every datapoint and this distance along with a normalization factor (which is the sum of this term for all the datapoints) is used to create the probability distribution, and the next centroid is randomly sampled from this probability distribution. Thus trying to pick points which are far away from the current points and having a better spread of centroids.

- Another advantage of stochastically choosing points from this probability distribution is that in case of an outlier which is very far from the actual clusters the outlier is not always selected as the centroid (which is actually a wrong choice of centroid).

- The rest of the algorithm is the same as the previously implemented online version of kmeans algorithm, it is just that the initialization is different in kmeans++, which can significantly effect the performance of clustering.

- To observe the change in the results of clustering by using this method of initialization over the previous method of random initialization, we find the distance of each data to the nearest centroid, and the minimum maximum and mean distance over the clusters. We used the same values of k i.e. [5, 10, 20, 30, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500] for a good comparison of the two methods.

- The Results obtained were as shown below:

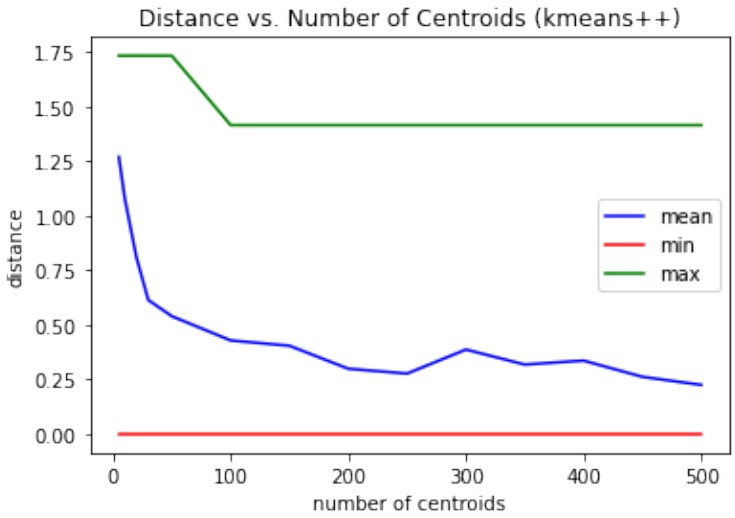

Figure 2: Min, Max and Average distances of datappoints from their nearest centroid for diff k

- When k is in the range [5, 250] as the number of centroids increases, the distance between the dataset and the corresponding closest centroids decreases. But when k crosses 250, the distance starts to increase slighlty but eventually decrease to a value around 0.25.

Comparison of K Means and K Means++

- As we have already seen from the plots in the respective sections, both k means and k means++ converge around a minimum value of around 0.25 for the mean distance. However, an observation to be made here is that the curve is steeper for k means++, thus we can say that convergence rate of k means++ is faster than that of k means.
- In both the cases, the values of go down as we increase the number of clusters and tend to almost zero as we keep increase the number of clusters or k.

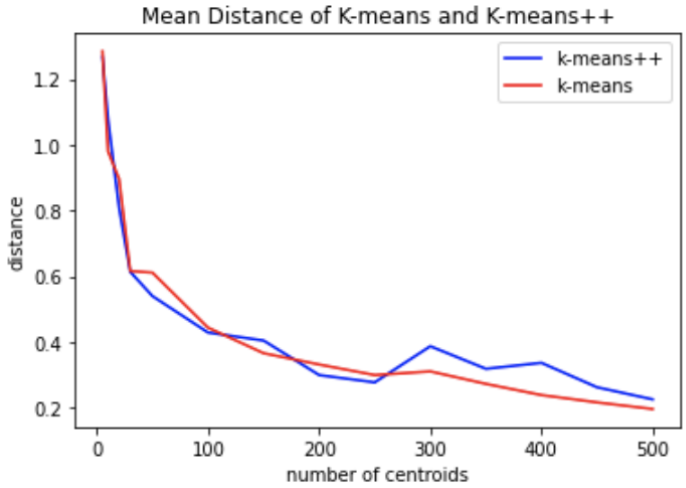

Figure 3: Average distances of datapoints from their nearest centroid for diff k

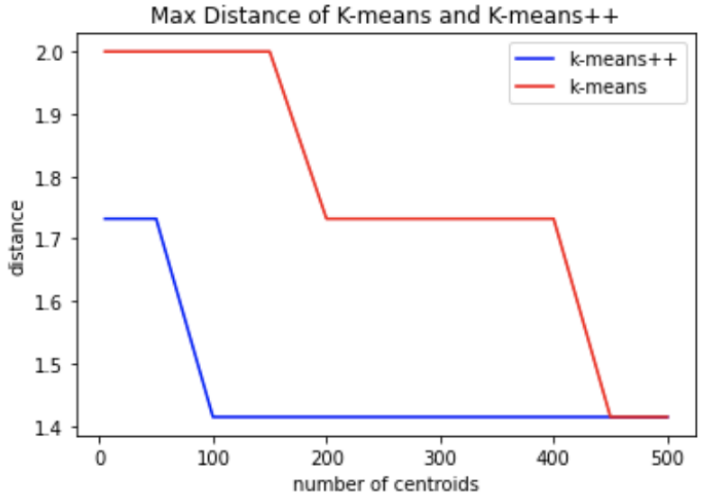

Figure 4: Max distances of datapoints from their nearest centroid for diff k

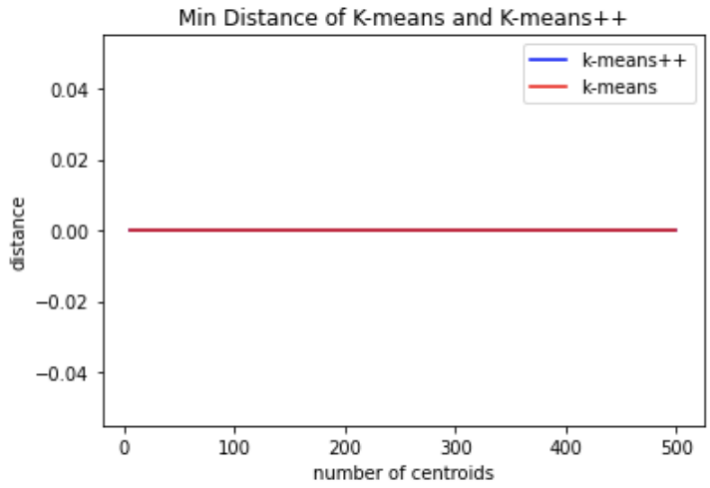

Figure 5: Min distances of datapoints from their nearest centroid for diff k

- We observe that the maximum distances for kmeans++ for a given value of k are significantly smaller than the distance values for kmeans, which signifies that the smart initialization was indeed useful, and the optimization algorithm reached a better minima value when initialization was done using kmeans++. The minimum distances for both the algorithms are zero, which might be the case that datapoints are same as the centroids.