CS565 Project #1

Summer 1 2019

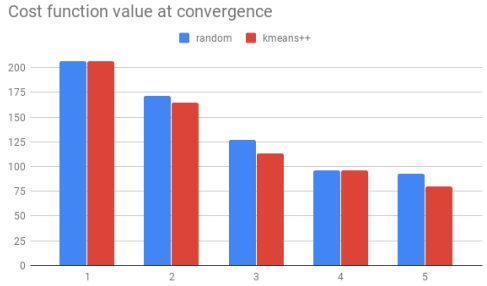
Alex Schwartzman

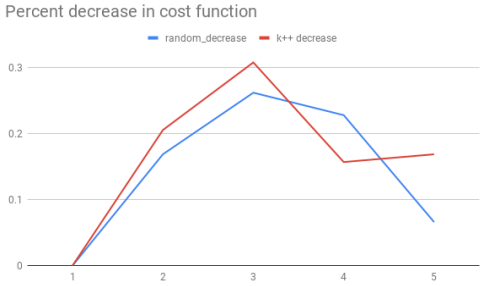
Introduction:

For this project I used object oriented Python, relying heavily on pandas and slightly on additional libraries for plotting and similar uses. I first worked with all the real values as well as the following categories: genres, production companies, production countries, and spoken languages. Initially I cleaned the data of two non-numerical entries in cells that are supposed to be real valued. Then, for the real values, I carried out Z-Score normalization, and mapped each set of values to the interval (0, 1) for consistency in distances. Upon further inspection I found that the inclusion of categorical data as binary distances (meaning distance = 0 if entries are the same, 1 otherwise) caused a very large skew, and as a result the algorithm did not work as its supposed to in theory: instead of monotonically decreasing, the cost function exhibited irregular, stochastic-like behavior which resulted in unreliable clustering. After many attempts at properly scaling the data, I decided to forgo the categorical data and focus merely on the real valued data. As seen in my code, there is a commented out class that implements categorical distances. The code is functional but it is inadvisable to run it due to restricted capabilities.

Initial implementations:

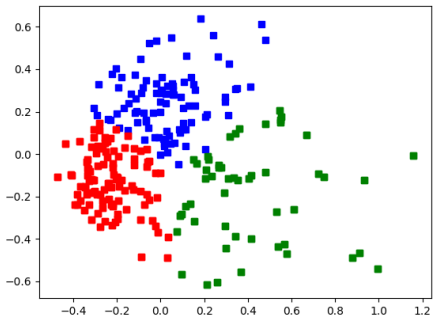
From here, I proceeded to implement kmeans++, and ran both algorithms for various values of k between 1 and 5, under convergence condition = 1. The reason for the convergence condition is that after empirical tests I found that the classification for 0.1 and for 1 results in nearly the same error, but 1 takes significantly less time. It was determined that the **optimal value of k is 3**, since the percent decrease in error from 2 to 3 was the largest in the experiment set - 26 $\sim$ 30% decrease for random k means and kmeans++ respectively. This is depicted in the charts below:





Running PCA:

In addition to the above, I implemented PCA using the sklearn module for Python.As kmeans++ performed both significantly better and significantly faster than random k means, I used that as a clustering algorithm while following the procedure in question 3. Graphing the results gave a somewhat peculiar output - some data points, especially around the borders of the clusters, seem to have been assigned to the wrong cluster, as shown below.



Inaccuracy in PCA:

I believe that this is due to the high sensitivity to decimal accuracy in the data: mapping budget that ranges from 0 to a few hundred millions to Z-Score, and then to the unit interval means that very large accuracy is required to maintain the characteristics of the data. As a result, when PCA is carried out and even when the data is stored, the required accuracy is not achieved and thus borders between clusters become somewhat blurry. This happens to only a small percentage of the data, but nonetheless it is an implementation problem that can be addressed by changing the normalization procedure.

1D clustering with dynamic programming:

Next, I implemented the dynamic programming solution to the 1Dimensional K means problem, by first running PCA that results in 1 component, and then using dynamic programming to compute unit cost in constant time. Unfortunately, I keep doing something wrong and the algorithm does not seem to be working quite right. It computes an optimal clustering cost but I'm not sure if its right. I've been working on this since Sunday morning non stop and I just can't figure it out. The code for this is in the class KMEANS, under the method D1_init. Any insights on what I did wrong will be vastly appreciated.

Clustering Disagreement:

Finally, I computed the disagreement distance between random clustering and kmeans++ (to make up for the errors in 1D), by using the provided disagreement distance function. The disagreement distance that I got was –disagreementdistance–.

Running instructions:

First, please make sure you have the following python libraries isntalled:
pandas,**numpy**, **matplotlib**, **sklearn** .

To run the program, use the following command in the command prompt: python source.py $< k >< init >$ where k is an integer and init is the initialization type. For init, the following are acceptable arguments:

'random' - runs vanilla kmeans.
'kmeans++' - runs kmeans++.
'PCA' - carries out PCA analysis with two components, plots the data (note that for this option, k must be less than 10).
'1D' - runs 1 dimensional clustering. (currently gives only optimal cost, not sure if the cost is right)
'dist' - runs kmeans++ and 1D clustering, computes the disagreement distance and reports it.

The program above stores all labels in a file called 'output.csv'.