JakeCroninHW3: KMeans Clustering

by Jake Cronin

Summary

I built a KMeans classifier in MatLab.

Given a CSV file with numerical or categorical attributes, a k value, and an output file, my KMeans classifier clusters the provided data into k clusters and writes the cluster id of each data entry as well as the sum of squared error to the output file. My KMeans implementation uses Manhattan distance to better handle high dimensional data.

Functions and Scripts

TestJakeKMeans

How To Run

type "TestJakeKMeans" in the MATLAB command line

##### What It Does

This script runs my KMeans classifier on the Iris dataset with 3 clusters, and runs my KMeans classifier on a large movie database for 1-10 clusters. The movie database file is very large, so allow about 5 minutes for this script to run. After execution, two figures will be created with the SSE of the clusters generated by my KMeans implementation and of those generated by MATLAB's built in KMeans funciton over the number of clusters.

---

runJakeKMeans

How to Run

type "errors = runJakeKMeans(\<inputFile>, \<k>, \<outputFile>)"

Optional Parameters:

includePlots: a boolean flag that if set to true will generate 2d plots of the clustering **FormatSpec**: For reading in complex data files. Default is set for reading the iris dataset.

#### What It Does This function classifies the data from the CSV input file into k clusters based on Manhattan distance. The cluster assignments and SSE are written to the output file.

---

jakeKMeans

How to Run

type "[ids, centers, squareError, errors] = jakeKMeans(data,k)"

Data: 2d numerical array K: number of clusters

ids: 1d array of cluster assignments **centers**: 2d array of cluster centroids **squareError**: SSE of final clustering **errors**: 2d array of clustering SSE on each iteration

Optional Parameters:

plotClusters*: Boolean flag that signal to generate a plot of the clusterings for each iteration of KMeans* ***plotCenters****: Boolean flag that signals to generate a plot of the centroids across iterations*

What it Does

This is my implementation of the KMeans algorithm. The algorithm first chooses k random points from the dataset to be starting centroids. The distance from each point to the closest centroid is calculated by Manhattan distance, and the points are labeled to their nearest centroid with the ids array. From here, the algorithm iteratively recalculates the centroid position by the average of all data points assigned to this centroid and then recalculates the nearest centroid for each data pont. This loop continues until there are no changes to cluster assignments between iterations.

Other Functions and Scripts

loadDataScript - Script that loads complex data from movies_metadata.csv file

prepData - Script that removes unwanted data from movies_metadata.csv dataset

AnalyzeData - Script to compare the performance of jakeKMeans against Matlab's KMeans function.

Results

The two data sests that I looked at are the Iris dataset and an imdb movie dataset.

Iris: 150 records with 4 unlabeled attributes.

IMDB : 45573 movie records. - I ignored all non-numeric attributes - Looked at Budget, Popularity Rating, Revenue, Runtime, Average Rating, and Rating Count.
- Ignored all incomplete records, leaving 45203 records

Pre-Processing

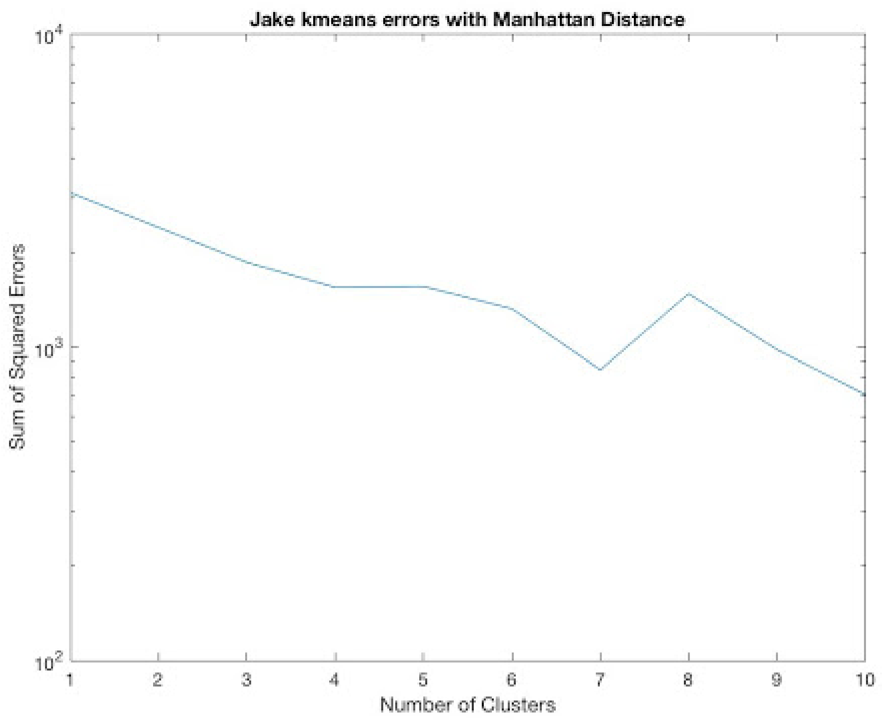
After eliminating non-numeric attributes and removing incomplete records, I normalized all attributes for both datasets onto the range of -1 to 1.

Dimensionality reduction was not necessary for the Iris dataset, because there are only four attributes. Dimensionality reduction in the IMDB dataset also did not dramatically improve the clustering, so it was not used in the final results.

Results

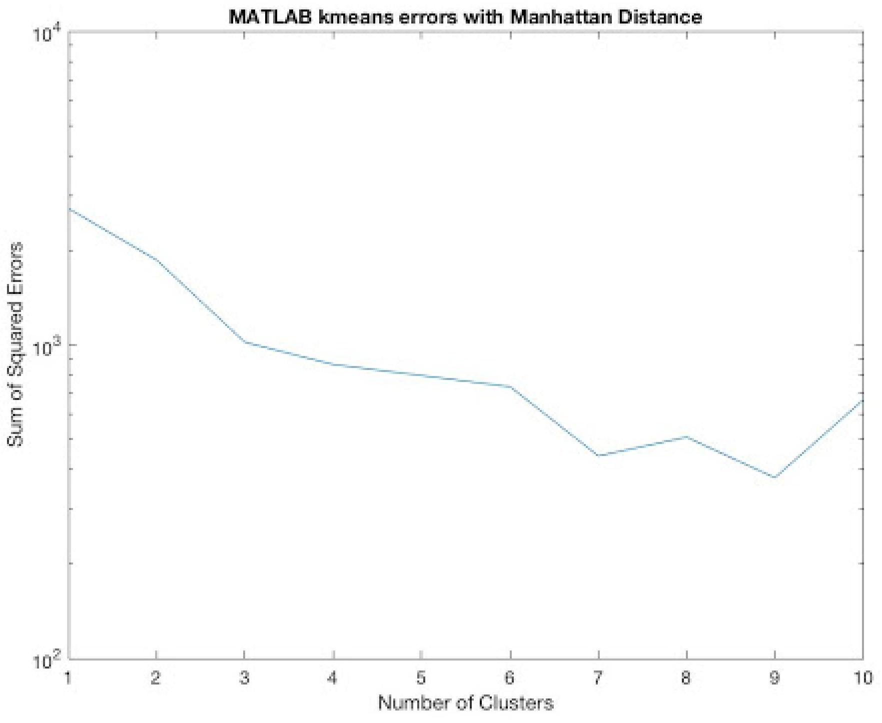
IMDB Dataset

My Kmeans function was run for K = {1 .. 10} clusters and the SSE was calculated for each iteration. Plotting the SSE values over the number of clusters yields the plot blow.

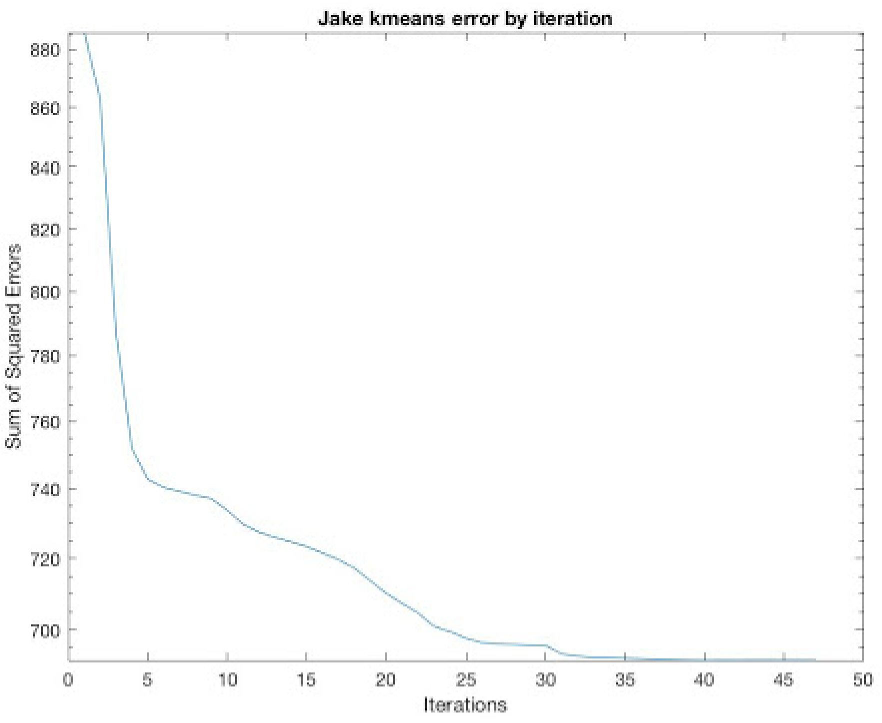


Looking at this plot with a logarithmic scale Y axis, the elbow technique shows that there are 7 distinct clusters of movies in this dataset. Due to the high dimensionality of the data, it is difficult to comment on the composition of the various clusters.

In addition, I noticed a discrepancy between my KMeans function and MATLAB's built in KMeans function (plot below) revealing limitations to the traditional KMeans algorithm. The bare bones KMeans algorithm, what I implemented with jakeKMeans, is susceptible to stopping on local minima, but MATLAB's implementation is modified to better find the global minima. As a result, the SSE values are lower on the error graph for MATLAB's built in KMeans algorithm vs my implementaiton.

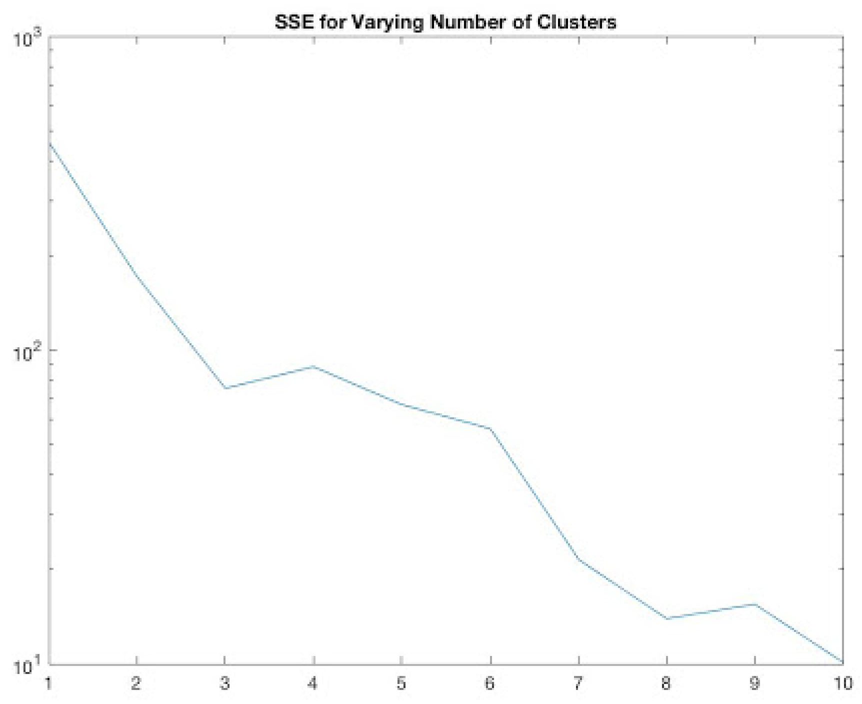


To ensure that the algorithm was working correctly, I plotted the SSE at each iteration of my KMeans algorithm for k = 10 clusters:



Iris Dataset

My Kmeans function was run for K = {1 .. 10} clusters and the SSE was calculated for each iteration. Plotting the SSE values over the number of clusters yielded the plot below.



Looking at this plot with a logarithmic scale y axis, the steepest decline occurs between 2 and 3 clusters. Subsequent changes to the number of clusters yield dramatically smaller improvemtns to the SSE. Therefore, by the elbow technique, there are 3 distinct clusters in the Iris dataset, which matches the number of classes offered in the labeled data.