Clustering Using the K-Means Algorithm

by Will Potter and Teddy Knox

We investigated the K-Means algorithm and its ability to correctly group similar objects into clusters by shared attributes. Our intended use was to cluster a movie review as either positive or negative based off of the words used in the text. With our given movie review set, we decided to cluster our reviews around the frequency of words used in the review. For example, if a review featured the word “like” four times and then word “love” two times, it would be more likely to cluster with another word that uses “like” and “love” rather than “hate” and “despise”. Unlike a naive bayes classifier however, this model requires no training and allows us to simply analyze the data we have and determine whether a point is in a certain category based off of the shared characteristics (in our case, word frequencies) and group it with like points.

Initially, we implemented a basic K-Means function with no modifications or customizations. We extended upon several existing functions and classes by randomly choosing several points to serve as centers of K clusters. Our algorithm then places each point in one of the K clusters, based off of the shortest distance to the center. After all points are assigned, we recalculate the centers of the clusters based on the mean of all attributes the points in a given cluster. If an attribute is not present, we assume that its value is 0, and still include it in the mean calculation. After recalculating centers, we then repeat the process of clustering points based on the shortest distance to a center. If the shortest distance for one of the points has changed, we move it from one cluster to another, and then recalculate the center points. We continue this process until the centers do not move any more and no points switch clusters. To manage the total distance moved by all centers after an iteration, we sum the differences between the new centers and their corresponding previous centers.

After implementing this algorithm, we ran it on a dataset of roughly 13,500 movie reviews. We discovered that the clustering algorithm worked relatively well with a dataset of this size. Our average weighted purity was roughly 81%, implying that most reviews successfully clustered by the words used. Since the data has many features and there may be substantial overlap between points with different labels, a 100% clustering purity is unreasonable to expect. For this reason, we are satisfied with seeing our roughly 81% average purity.

Data Points as initial centroids	

Unweighted Average Purity	84.623%
Weighted Average Purity	80.620%

Upon further analysis of this algorithm, we were able to count the total centroid movement as well (between all K) centroids. For a K value of 5, all 5 centers were moving a total distance of under 5 units after only 4 iterations. After 1 iteration however, all centroids moved an average combined distance of 327 units. This shows us that for only 4 iterations, we were able to come up with a rough clustering of all points that was indicative of the final clustering. K-means is incredibly efficient in the initial stages of the algorithm, whereas we would see roughly 10 iterations per each clustering call. The additional calls come from the algorithm determining the proper location of a few points that lay on the borders of specific clusters. For this reason, we realized that with a large dataset, it could possibly be very efficient and relatively accurate to limit the number of iterations after the centroids stop moving more than a tuned threshold.

One of the variations that we implemented involved initializing the algorithm with randomly selected points within the vector space of the dataset. This lengthened the initialization process, because it required that we loop through all of the data points once to discover all of the possible different dimensions, and then again to discover the range of those dimensions. With this information, we randomly generate our k centroids in these dimension ranges. To see how this variation impacted the results, averaged the evaluation functions of 10 runs on the movies data. The results showed absolutely no change in effectiveness. We attribute this to the relatively well clustered data, and the general ineffectiveness of the variation overall. If we had more time, we would have improved our algorithm by reducing redundant distance calculations with the triangle inequality. We might also have implemented soft k-means, where each data point is assigned probabilities of belonging to the different clusters.

Random Points as Initial Centroids	
Unweighted Average Purity	84.394%
Weighted Average Purity	80.624%