Some abstract info

X-Means is a more advanced clustering algorithm that can be really useful. Clustering is important with data sets to find groups in the data. Using K-Means helps to find clusters and their average or center. X-Means builds on K-Means but it helps find the correct number of groups in the data.

X-Means

The general design for X-Means is starting with K-Means and then adding another K point. X-Means implements K-Means but adds a cluster centroid if it needs to. It starts by reading in from a file with an x and a y value on each line, then puts it in memory. For each line a cluster point object is created which includes the x and y coordinates and the cluster it belongs to. While the program is parsing it finds the range to create the random centroids. Once the program finds the range it creates random centroids using a javas random. It starts with the min number of centroids that the user specified. Once the centroids are added they’re added with an id and the random location Point. Then all the points are added to the created centroid, then the K-Means runs. K-Means starts by assigning the cluster point to a centroid. While the points are being added the range of the values in the cluster are calculated. Then it averages out the values of the cluster points in the centroid. For each centroid the new location is calculated. Once the centroids have reached a convergence with the cluster points we see if we can add a new point.

Once the clusters are settled they are separated up then they are weighed using the Bayesian information criterion. Then for each cluster a vector is created form the centroid location and two other K points are added on that vector while the old centroid is removed and saved. Then for each cluster the K-Means algorithm is run. Then the two new clusters are weighed using Bayesian information criterion. If the new clusters are weighted heavier than the parent, he points is added. If two or more points need to be added, the point with the biggest difference is added. If no points need to be added, then the program terminates. The testing was done with a fairly big data set one was made using poisson distributions. The other dataset was created using java’s double generator. The numbers were between zero and 1. It was uniformly distributed so running X-Means on it produced different results. So by creating about 100 points between 0 and 1 then adding them to the dataset then adding a constant to the double you can create clusters. So I did that for a few datasets.

Figuring out how to calculate the Bayesian information criterion was the hardest part. There was one point when I needed to calculate whether or not we need to continue but they function call was on the right side of the Boolean operator. Whenever the program reached that point the call short circuited and the function call was never called. The Point data structure java provides shows the point locations in integers. So debugging for small doubles was hard to work with. Some future features I want to implement will be efficient point traversal with KD trees. Also maybe using SIMD instructions to help make the program faster.

Work Load

John started on id3 figuring the information gain calculator. John also started and completed X-Means.

Sources

X-Means Extending K-Means with efficient estimation of number of clusters

By:

Dan Pelleg

Andrew Moore

K-means and KD-trees resources

<http://www.cs.cmu.edu/~dpelleg/kmeans.html>