Introduction

The Robust K-means algorithm (RKM) [1] is an improvement in accuracy and efficiency over previously used geometric clustering algorithms. It accomplishes this by taking an Information theory approach to geometric clustering using the Information Bottleneck method. The resulting algorithm is more robust to initial centroid placement and has a faster execution time than previous methods.

In this paper, I will test the Robust K-means algorithm, as developed and implemented by S. Still and L. Bottou, on randomly generated data sets taken from Gaussian distributions. I will then compare the results with the Soft K-means algorithm with Deterministic Annealing and the standard K-means algorithm, using several heuristics for initial centroid placement. I will perform experiments in both 2 dimensional and 20 dimensional space, and compare how the algorithms performance degrades as the problem complexity increases.

Robust K-means

The Robust K-means algorithm uses the Information Bottleneck method as a foundation for its solution to geometric clustering problems. Clustering, from an information theory standpoint, is a form of lossy compression based on the ratio of datum to clusters. The goal is to retain as much relevant information about the location of the data points while compressing the data into as few clusters as possible. RKM regulates this compression with the Lagrange parameter $\lambda$. The clustering criterion of RKM is therefore

$$\max \left[ I(x, c) - \lambda I(c, i) \right]$$

where $i$ is the data index, $c$ is the cluster index, and $x$ are the locations of the datum. The goal is the maximum the amount of relevant information retained, given the compression parameter $\lambda$.

The temperature $\lambda$ plays a role in determines how the algorithm behaves. For any value $\lambda < 1$, the algorithm converges to a “hard” K-means solution, though, depending on its exact value, tends to exhibit less sensitivity to initial centroid placement. For a $\lambda$ value of 1, the resulting equations are exactly that of the soft K-means algorithm. Therefore, the RKM algorithm can behave as either previously defined algorithms, yet can be tuned via the $\lambda$ parameter to produce a more efficient and accurate solution.

The parameter $s$ also plays a role in the behavior of RKM. The $s$ variable is known as the initial variance and controls the algorithm's sensitivity to initial centroid placement. The $s$ variable controls the balance between longer execution time and robustness of the algorithm. For simplicity, I used an $s$ value of 1 in my experiments in this paper.
Convergence Rate for RKM

This algorithm was coded so that the stopping parameter was defined as the number of iterations to run the algorithm, regardless of whether the algorithm converged or not. This allowed for an interesting experiment of how many iterations are required to obtain convergence, given different dimensional and data sizes. I generated random data points generated from random numbers of Gaussian distributions. There are 500 data points per Gaussian and each Gaussian has the same variance but different random mean values. Below is the resulting convergence rate of the RKM algorithm.

The results of this test were rather mixed. For most data sizes, the standard K-means algorithm converged on a solution slightly faster than RKM, though it says nothing about the accuracy of the converged result. It seems that RKM takes more iterations to converge on small data sets (3 or less) but, is competitive to K-means on larger data sets. Based on the accuracy of RKM versus K-means, the convergence rate was not significantly slower than the standard K-means algorithm.

The error margins show that K-means also has more variance in its convergence rate, with some outliers being up to 15% larger than the mean. RKM was more consistent, though overall had a slower convergence rate.
**Error Rate**

The largest proposed advantage of RKM is the significantly reduced error rate over K-means. In previous experiments, I compared the Soft K-means algorithm with standard K-means using several heuristics for initial centroid placement. I found that SKM performed 10-15% better than the best K-means heuristic, yet the trade off was a huge increase in execution time. RKM does not suffer from the same execution complexity as SKM, yet promises accurate results. I ran RKM along with SKM and several heuristics for K-means. The results are below.

![Graph of Error Rates](image)

Figure 2: Average error rates of RKM, SKM, and K-means using several heuristics. The data sets are taken from random Gaussian distributions (500 datum per Gaussian). RKM ran until convergence with $\lambda=0.5$ and $s=1$. “Standard” is the standard K-means algorithm with random initial centroid placement, “Distance” is K-means using a heuristic that places initial centroids far away from each other, “Point” places initial centroids directly on random points, and “Repeat” repeats the standard K-means algorithm 3 times and takes the best result.

The results show that RKM outperforms every previous algorithm and heuristic in terms of accuracy. Even SKM, with much longer execution times, has an average error up to 10% higher than RKM. Perhaps by further changing the temperature of SKM, it could come close to the accuracy of RKM, yet that would further increase its execution time. From this experiment it seems clear the RKM is superior to SKM in every problem domain.

The error bars on the graph show that RKM, in addition to being overall much more accurate, only had a variance of 5% from the mean. Compared to the standard K-means algorithm, which had outliers with up to 10% more error than the mean, RKM is very stable.
**Execution Time**

To test the efficiency in execution time of RKM, relative to the other clustering algorithms, I repeatedly executed each algorithm on data sets generated from random Gaussian distributions. I averaged out the execution time and tested it for different size data sets. Each Gaussian contains 500 data points, so the size of each data set also increases with the number of Gaussians/centroids. Below is the results of the 2D data set test.

![Figure 3: Execution time of each clustering algorithm on random 2D Gaussian data sets. RKM ran until convergence (determined earlier and shown in Fig.1) with $\lambda=0.5$ and $s=100$.](image)

Surprisingly, RKM actually has a faster execution time than even the standard K-means algorithm. I would expect it to have slightly longer execution time, due to the increased complexity of the algorithm, and the slower execution of K-means may be a result of inefficient implementation. It is clear from this experiment that RKM is a superior algorithm to any previously used, even with the use of combination of heuristics.

To see how these algorithms perform under increased complexity, I tested execution time of both RKM and K-means on 20-dimensional data sets. I did not test Soft K-means, as the execution time and complexity of the solution would be too large.

![Figure 4: Execution time of K-means versus RKM in 20D space. RKM ran with $\lambda=0.5$ and $s=100$.](image)
Surprisingly, the results of this experiment were nearly identical to the previous test. Apparently, both of these algorithms have nearly the same execution times, regardless of dimensional size. K-means increased in execution time by a factor less than 2, while RKM increased by even less. This seems to be because the only change in the computational effort is the distance function. The number of points to evaluate as well as the number of clusters is identical to that of the 2D space.

Conclusions

The Robust K-means algorithm has many advantages to all previously used clustering algorithms. Firstly, it uses a information theory approach to geometric clustering, removing the need for explicitly defined similarity measures. As was shown above, it also performs better than any of our previously used algorithms or heuristics in every area tested. The generality, efficiency, and accuracy of this algorithm provides very clear advantages over any geometric clustering algorithm used thus far.

Previously, the trade-off between speed and accuracy meant choosing between Soft K-means, standard K-means, and some combinations of heuristics. The results above show that this trade-off is no longer necessary, as RKM executes faster than the simplest K-means and has lower error rates that the slow Soft K-means. A more thorough investigation regarding execution time may show that the K-means implementation used here is inefficient, but regardless, RKM performs very well in all domains. In addition to performance gains, the ability to set parameters of RKM allow it to be catered to the specific problem, making it an even more attractive solution to many problem domains.

Notes on Provided Code Packages

The code package provided by S. Still contains all necessary tools to run and test RKM. The utils.c and utils.h files contain functions relating distributions and mathematical functions. It provides code to create numbers based on uniform Gaussian distributions. The gendata.c and gendata20.c file is used to create data sets based on 2-dimensional and 20-dimensional Gaussian distributions, respectively. stab.c and stab20.c contain the code to execute the RKM algorithm for 2-dimensional and 20-dimensional data sets. They take in several parameters and a file, which can be generated by piping the output of gendata into a file. I had to edit the code slightly to get it to compile, but once compiled it was an easy process to run and test.

References