Investigating Seed Values for the K-means Clustering Algorithm

David Kronenberg

Abstract

A major shortcoming with the K-means clustering algorithm is that it relies on random seed values to search for the best possible clusters; a common method is to run the algorithm many times and simply use the results which generate the "best" clusters. Clearly this method is very time consuming and will still produce non-deterministic results. This paper will report on different methods that use information from the data set to be clustered to produce seed values that will be deterministic and will produce accurate clusters as quickly as possible. We have implemented and tested eight different methods for calculating these initial seeds, and have found that several of them in particular perform very well when compared to using "random centers", in terms of both their error and performance. We have also studied how many times we would need to generate new "random centers" to produce the optimal clusters and will present those results as well.

Introduction

Clustering is the process of taking a set of data points and assigning them to groups or clusters where the points in the same cluster are more similar to each another than the members of the other clusters. These clusters are then used to help find patterns in the data set that may allow relevant correlations to be inferred. Clustering has many different applications in a variety of fields including: image segmentation, market research for advertising, data mining, several different biological applications, and clustering can also be used in file compression. Because clustering is used in many different fields, it is important that an algorithm is able to generate the most accurate clusters, and converge on those clusters as quickly as possible, because clustering is often performed on massive data sets, or the cluster assignments may need to be determined in real-time.

This paper will focus specifically on the K-means clustering algorithm which is a fairly straightforward algorithm and is one that is relatively easy to implement. The K-means algorithm is a partitional algorithm, that is, one point is assigned to reside in only one cluster and all points are equal members of that cluster. The K-means algorithm is widely used because it is fast and produces exactly the number of clusters specified: K. The results from the K-means algorithm are also sometimes used as the initial values for more time consuming (e.g. hierarchical) clustering algorithms. The K-means algorithm works by initially taking K randomly selected points (within the data range) as the cluster centers, and then assigning all of the points in the data set to the nearest cluster center. It will then recalculate a new center for each cluster by computing the mean of all the points belonging to that cluster. After recalculating all of the new centers in this manner, it will reassign all the points again, using the new centers and repeat this process until all the points no longer change clusters after this reassignment step.

The major downfall with the K-means algorithm is that the final clusters and convergence time are highly dependent on the initial centers. Since some initial centers may produce non-representative clusters, and using "random" initial centers will produce non-deterministic results, the most accepted strategy is to run the "random" algorithm many times and simply use the "most effective results". However, this can be very time consuming as it requires that the algorithm be run hundreds of times. Therefore, we shall now describe eight different methods for pre-computing what could be an effective set of K initial centers using information about the data set (and those particular data points). It is our hope that these methods will produce final clusters that are on par with the results obtained from examining many randomly produced centers, and that these methods will execute in much less time since each one will only need to compute the set of initial centers once (to produce its deterministic result).

A set of effective initial centers will produce a set of clusters which has a small squared error statistic and the algorithm will converge in as few iterations as possible. The squared error statistic is defined as the sum of all the differences between each point in a cluster and the cluster's center. This is simply a measure of how similar the points in the cluster are to one another and reflects how accurate the final centers are. The other relevant performance statistic is how many iterations the K-means algorithm takes before it converges; this measures how quickly the initial centers become the final centers thereby influencing how long it takes the algorithm to execute.

The Methods

We implemented and tested eight different methods to compute the initial centers, using a wide variety of strategies. Some methods use actual points in the dataset while others use means of different sets of points. All of the strategies we tested assumed that the data was sorted in descending order, which our program accomplished before any of the computing was done.

The first method takes N, the number of points in the dataset, and uses the points at the indexes of the integer values of N/K, 2N/K, 3N/K and so on, as the initial center points, until reaching the points at (K-1)*N/K and finally N, the last point in the sorted data set. The idea here presumed that if the initial centers had an equal number of points in between them, the centers would converge to stable values quickly.

The second method we tested relies on a distance D, and determines the K points that have the largest number of points within that distance, and are not within D distance of each other, and assigned those points as the initial centers. To get the value for D, we initially computed the average distance between two consecutive points; we then adjusted it according to how many reasonable centers it produced. The hope was that the points with the highest number of points around them would be the most effective initial cluster-centers.

Our third method was to make K groups of N/K consecutive points and set the means of those groups to be the initial centers. The reasoning behind this method was that if we took the means of a group of points that all had an equal number of points in them, the centers would quickly move towards the true cluster's centers since they were already the means of a group.

Method four was very similar to method one which used evenly distributed centers, but in an attempt to remove the "bias shift" evident in method one, we subtracted N/(2*K) before determining the indices, since the last data point will not be the center of any cluster unless it is a cluster by itself (which is highly unlikely). This shift might move all of the points closer to the middle of the range of data, and hopefully towards the final clusters in fewer iterations.

The fifth method we tested involved breaking up the range of the data points. To do this we took the minimum and maximum values of each dimension, which was very straightforward since the points were now sorted. We then took this range and divided it by K+1, and then used the multiples of those values, added them to the dimension minimums and those values became the centers. The idea behind this method was similar to method one. By placing the centers equidistantly from one another, it might allow them to converge to the final clusters in fewer iterations, but instead of breaking up the points this method broke up the entire range for each dimension.

The sixth method we tested found the K-1 largest gaps between consecutive points, which again was simple to do since the points were already sorted, and calculated the means of the points in between them to determine the initial centers. The assumption with this method was that the clusters would be located between the largest gaps in the data, and that taking the means of those points would produce the clusters in fewer iterations than any other method.

We also tested the global K-means algorithm proposed by a group from the University of Ioannina, Greece, to compare how their method performed. Briefly, the global K-means strategy ran the K-means algorithm, for each integer from 1 up to K, adding in the center which generated the lowest error to the previous set of "stable centers", after the number of clusters was incremented.

In order to compare all the method's results to some "baseline" to see how well they performed, we attempted to find the best and worst possible centers, and the statistics they generated. To find the "best" centers, we ran the algorithm with random centers over a thousand times and kept the centers with the lowest squared error, and then used several methods to attempt to find the "worst", (or at least "reasonably poor") initial centers. In the first such method, we set the initial centers as the first K points in the data set, likewise our second method used the last K points in the data set, and a third method used half the centers being the first K/2 points as well as the other half coming from the K/2 final points in the data set. The last method assigned a random number of points to each cluster, while also ensuring that all of them had at least one point assigned to it.

In addition to trying to find an efficient and effective method for determining initial seed values, we also investigated how many times it was necessary to run the K-means algorithm to produce a quality set of clusters. This should help to determine whether or not the pre-processing on the data is worth the effort, with regards to the time it takes, as contrasted against the number of times it takes the K-means algorithm, seeded with many different sets of random centers, to produce a quality result on average. Although pre-computing initial center strategies will almost always take less time, running the algorithm with many random centers, in most cases, will produce better results.

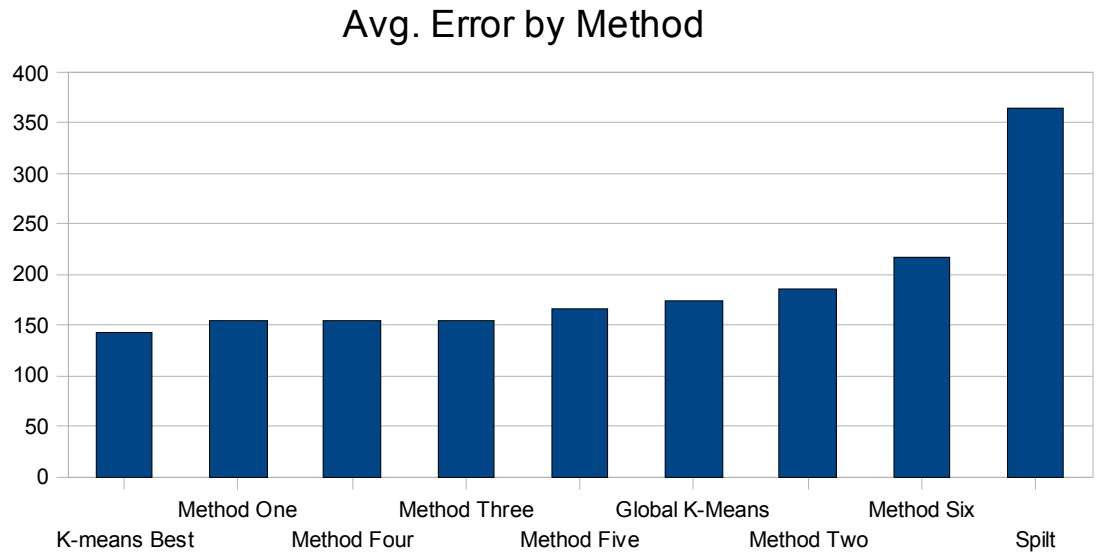
How the Testing was done.

To test all of our different methods, we ran them all on 19 different, one dimensional data sets, each comprising approximately 300 points. All methods were tested with K=20, which was a reasonable value for the most "appropriate number" of clusters for these data sets. We then recorded the error produced for each dataset as well as the number of iterations it took each method to converge to the final clusters. We then took these values and averaged them for each method over all of the datasets, and these averages are what appear in graphs A and B, representing squared error and number of iterations, respectively. The final graph, graph C, is an attempt to find the number of times, on average, to run the the algorithm with random centers, to produce clusters comparable to what we would get if we ran K-means 1000 times.

Results

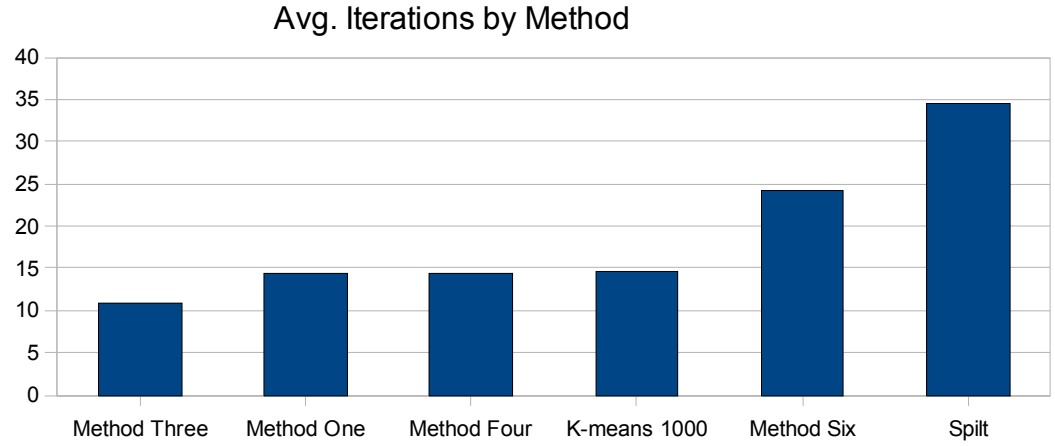
Graph A shows the squared error produced by all of the methods that we implemented, in addition to the K-means algorithm run with random initial centers, and our attempt to produce poor clusters. K-means (choosing the best from 1000 random centers) clearly performed the best and that was to be expected, but method one, three and four all performed very well in comparison and at about the same level as one another; each one did have a few datasets on which they did perform poorly, however, these three methods along with method six and the global K-means, were the only ones to not fail to produce K clusters on any one of the datasets. A method is said to fail to produce K clusters when it assigns zero points to a cluster (and its center), and would then proceed with just K-1 clusters. Both method two and five failed on around one third of the datasets and we simply tabulated the error produced at the time of such a failure. Method five and global K-means did not perform quite as well as those first three methods, although method five did fail on six of the fourteen datasets. Method two and six both performed slightly worse on the datasets, although method six did not fail on any of the datasets. The split method, which did not fail on any of the datasets, performed poorly (as expected) and was included in both graphs for comparison purposes. The other methods that were intended to perform poorly were omitted from the graphs since they were slightly worse than the split method and we wanted to emphasis the other methods (not those four contrived approaches).

Graph A



Graph B depicts the average number of iterations each method used (Methods two and four were omitted because they occasionally failed). The most effective method in terms of iteration count, was method three which averaged 11 iterations per dataset. Method one, method four and the average for the standard K-means (run one thousand times), came in at around 14 iterations per dataset. Method three did perform significantly better than any other. The global K-means algorithm was not considered because K-means was performed 1000 times; therefore the results can be very misleading, as it is a very time consuming method. Method six did very poorly needing around 24 iterations to converge on average. The split method needed 34 iterations on average and was again the only one of the contrived methods included in the graph.

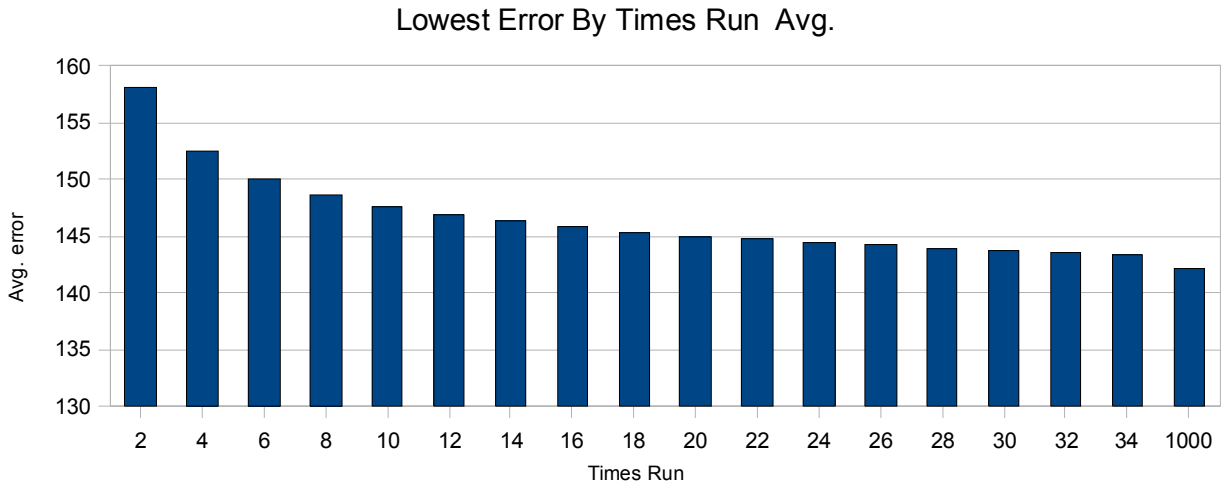
Graph B



Graph C illustrates how many times the K-means algorithm should be run with random centers to produce a quality result. To do this we ran the algorithm with random centers a certain number of times and averaged it over one thousand runs to give an accurate measurement of the average error the algorithm would produce when run that many times. We did this for every even integer up to 34 (averaged over one thousand times for each data set). The results were what was expected: trying more randomly generated centers lowered the average error of the "best" cluster assignment for that group of

generated clusters, and the larger the number of times run, the smaller the error became. The graph below resembles the second quadrant of a graph of y=1/x; as the number of times the algorithm is run increases, the line approaches a horizontal asymptote which represents the best possible error for that dataset. This shows that there is some reasonable number of times to run the algorithm-which is smaller than one thousand- so that it will generate on average, a result similar to the best possible.

Graph C



Conclusions

Of all the strategies we tested, the most effective was method three. Although it ranked third in terms of standard error, it was still very close to the best of all the methods, and it performed significantly better than all of the others in terms of number of iterations. One of the reasons it had so few iterations may be because it already used a mean of a set of points in its pre-processing, so that the centers started off as part of a cluster, and from there it had to do less work to converge to the final centers points. Method one also worked very well and had the best squared error, and it also performed well with regards to the number of iterations. Method one took all of the points and distributed the initial centers equally between them; since every center is a point, it is highly unlikely that this method will fail to produce K clusters. Method four was the last of the three methods that performed fairly well. Its performance was between the other two methods just mentioned, in terms of iterations and squared error, which is odd since it was supposedly an improvement on method one and it actually performed worse in terms of squared error.

Method five performed next best in terms of squared error, although it failed on 6 of the 14 data sets; these failures were probably due to the fact that it broke up the data range with no regard as to where the actual points where, so it is likely that it simply put a initial centroid in a large gap in the data and was then unable to maintain the correct amount of clusters. Global K-means came in next, in terms of squared error, although we did not attempt to calculate the number of iterations it took to converge as it would run the algorithm K times. Though the basic idea behind it seemed sound, its performance was below average primarily because the last run through the algorithm was the most effective for that many centers, but that does not necessarily make it the most effective starting point for its next iteration. Method two, which attempted to find the centers by selecting the points that had the most other points within a certain range of that point, performed very poorly because when it found the points within the range, the points it choose as centers were not necessarily in different natural clusters. This meant that if a large cluster was more dense than the others, the method would place two centers

in an area that truly only contained one actual cluster. (Method two also failed on 5 of the 14 data sets.) Method six did the worst of any method (that was not intended to perform poorly). It took the K-1 largest gaps between two consecutive points and took the means of those groups of points; the problem with this strategy was that it could potentially use a gap that was contained in a legitimate cluster or could create one large cluster out of two natural clusters if those two clusters were close enough together.

Every proposed method was somewhat inconsistent to some degree, with some results being poor on one or two of the datasets, but then quite appropriate on other ones. This is due in part to the way the data in the sets is distributed, which makes finding a method that will work universally well, all on datasets, that much more difficult.

References

Anderberg, Michael R. , Cluster Analysis for Applications, Academic Press, 1973.

Bradley P. and Fayyad U., "Refining Initial Points for K-Means Clustering", *Proceedings of the 15th International Conference on Machine Learning (ICML98),* Microsoft Research, May 1998.

Kaufman L. and Rousseeuw P., "Finding Groups in data: An Introduction to Cluster Analysis".

Likas A., Vlassis N. and Verbeek J., "The global K-means clustering algorithm", *Pattern Recognition 36 (2003)*.

Pena J., Lozano J. and Larranaga P., "An Empirical comparison of four initialization methods for the K-means algorithm", *Pattern Recognition Letters 20 (1999),* 1027-1040.