K-Means Clustering Algorithms: Implementation and Comparison

Gregory A. Wilkin and Xiuzhen Huang
Department of Computer Science,
Arkansas State University,
State University, Arkansas 72467 USA
awilkin@csm.astate.edu; xzhuang@csm.astate.edu

Abstract

The relationship among the large amount of biological data has become a hot research topic. It is desirable to have clustering methods to group similar data together so that, when a lot of data is needed, all data are easily found in close proximity to some search result. Here we study a popular method, k-means clustering, for data clustering. We implement two different k-means clustering algorithms and compare the results. The two algorithms are Lloyd's k-means Clustering and the Progressive Greedy k-means clustering. Our experimentation compares the running times and distance efficiency.

1. Introduction

The genome projects are generating large amounts of biological data. When viewed as a whole, the data can be perplexing and confusing [1]. Data clustering approaches can group similar data into clusters. The grouped Data will usually reveal important biological meanings.

An easy abstraction for clustering data is based on a proximity relationship. Data that are close to each other tend to share some external relationship. This relationship can be established to group the data into clusters. Other relationships do exist in many different ways. For the sake of simplicity in this paper, we will only concentrate on the distance relationship so that a meaningful and simple analytical conclusion can be made from simpler comparisons.

For proximity relationships, a desired trait of the data is to be clustered in such a way that the squared error distortion is minimized both globally and locally. This will be what we use to measure the effectiveness of the algorithms in question. The mean squared error distortion is defined as $d(V,X) = \frac{\sum_{i=1}^{n} d(v_i,X)^2}{n}$, where X is a set of centers, $d(v_i, X)$ is the distance of point v_i to the closest cluster center and n is the total number of points [1].

The k-means clustering algorithm is one of the popular data clustering approaches. The k-means clustering algorithm receives as input a set of (in our case 3-dimensional) points and the number k of desired centers or cluster representatives. With this input, the algorithm then gives as output a set of point sets such that each set of points have a defined center that they "belong to" that minimizes the distance to a center for all the possible choices of each set.

We have implemented two versions of the k-means clustering algorithm. First, an algorithm is called the Lloyd's k-means clustering algorithm. This is a relatively faster algorithm and is fairly straight forward. The other version of k-means clustering that we implemented is called the Progressive Greedy k-means clustering algorithm. This is a more conservative approach and can take much longer but can sometimes yield better results than the former.

We analysis the algorithms and discuss some experimental results. These results will be based on the running time for the algorithms and the mean squared error distortion and will be compared for analysis of complexity and efficiency.

2. Algorithm descriptions

2.1. Lloyd's K-means clustering algorithm

Lloyd's k-means clustering algorithm, which was designed by S. P. Loyd [2] is fairly straightforward. Given a number k, separate all data in a given partition into k separate clusters, each with a center that doubles as a representative. There are iterations that reset these centers then reassign each point to the closest center. Then the next iteration repeats until the centers do not move. The algorithm is as follows [1]: 1. Assign each data point to the cluster C_i corresponding to the closest cluster representative x_i $(1 \le i \le k)$; 2. Compute new cluster representatives according to the center of gravity of each cluster, that is, the new cluster representative is $\frac{\sum_{v \in C} v}{|C|}$ for every cluster C.

0-7695-3039-7/07 $25.00 © 2007 IEEE
DOI 10.1109/IMSCCS.2007.51

The Lloyd's algorithm often converges to a local minimum of the squared error distortion rather than the global minimum [1].

We use C as the programming language to implement the algorithm. There are two structures for the points, specifically an array of points that will be dynamically declared when the user specifies the number of desired points, and there are k centers as kvalues that are also arrays. The kvalues have three arrays within them that hold the points that "belong" to that particular center value.

2.2. Progressive greedy K-means clustering algorithm

The Progressive Greedy k-means clustering algorithm is similar to Lloyd's in that is searches for each point the best center of gravity to "belong to" but different in the assignments. Lloyd's algorithm, in each iteration, reassigns up to every point to a new center and then readjusts the centers accordingly then repeats. The Progressive Greedy approach does not act upon every point in each iteration, rather the point which would most benefit moving to another cluster. Every iteration in the Progressive Greedy algorithm calculates the "cost" of every point. The cost is calculated in terms of a Euclidean distance: $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$. Each point p of coordinates (x_p, y_p, z_p)has a cost associated with it in terms of the current center C_i of coordinates (x_i, y_i, z_i) it belongs to. The point is a candidate to be moved if the cost can be reduced by moving that point from one cluster C_i to another cluster C_j of coordinates (x_j, y_j, z_j) with that cluster having a closer center. In other words a point is a candidate to be moved from C_i to C_j if

.

$$\left(i = \sqrt{(x_i - x_p)^2 + (y_i - y_p)^2 + (z_i - z_p)^2} - \sqrt{(x_j - x_p)^2 + (y_j - y_p)^2 + (z_j - z_p)^2} \right) > 0$$

Once all the candidates are calculated, the point with the largest i is then moved. After all points have value $i \leq 0$, the algorithm is finished.

Each iteration of the Progressive Greedy k-means clustering algorithm does the following: 1. Calculate the cost of moving each point to each of the other cluster centers as well as the cost of its current cluster center. For every point, store the best change if less than the cost of its current cluster center; 2. If there is a point with a best change, move it. If there is more than one, pick the one point that when moved sees the greatest improvement; 3. If nothing else can be done, finished.

Progressive Greedy k-means clustering is slower but the sacrifice is an attempt to minimize the squared error distortion mentioned earlier.

For the implementation of Progressive K-means clustering, the two C structures for points are the same as those for Lloyd's.

3. Experimental results

3.1 Analysis biological data

The work of M. B. Eisen et. Al [3] is one of the first to apply the clustering approach to analysis the gene expression data.

We apply the two clustering algorithms to analysis micro array data. The clustering algorithms could classify gene expression data into clusters such that functionally related genes might be grouped together. In the following example [1], the expression information of ten genes is recorded at three difference times. The distance matrix of the ten genes could be calculated based on the Euclidean distance in three-dimensional space.

Table 1. The expression level of ten genes at three different times.

Time	1hr	2hr	3hr
g1	10.0	8.0	10.0
g2	10.0	0.0	9.0
g3	4.0	8.5	3.0
g4	9.5	0.5	8.5
g5	4.5	8.5	2.5
g6	10.5	9.0	12.0
g7	5.0	8.5	11.0
g8	2.7	8.7	2.0
g9	9.7	2.0	9.0
g10	10.2	1.0	9.2

We apply the clustering algorithms on the above data set. The clustering algorithms "group the genes into clusters satisfying the following two conditions [1]": Within a cluster any two genes should be highly similar to each other. That is, the distance between them should be small. This condition is called "Homogeneity". From any two clusters, two genes should be very different from each other. That is, the distance between them should be large. This condition is called "separation".

The three clusters of the ten genes are as follows: {g1, g6, g7}, {g3, g5, g8}, and {g2, g4, g9, g10}. Table 2. and Table 3. are the running result comparisons when we apply the two clustering algorithms to the above biological data.

Table 2. The running time comparison of the two algorithms for different k values.

	k = 2	k = 3	k = 4	k = 5
Progressive	0.14	0.21	0.25	0.28

Lloyd's	0.46	0.47	0.48	0.62

Table 3. The MSD comparisons for the two approaches.

	k = 2	k = 3	k = 4	k = 5
Lloyd min MSD	0.751	0.686	0.000	0.686
Lloyd max MSD	19.131	7.000	7.000	7.000
Lloyd global avg. MSD	**9.941**	**2.812**	**1.952**	**2.812**
Progressive min MSD	0.751	0.751	0.125	0.686
Progressive max MSD	19.131	7.642	7.000	9.980
Progressive global avg. MSD	**9.941**	**5.131**	**2.692**	**3.805**

3.2 Analysis randomly generated data set

We let the computer generate random points and see the results. Below is a set of figures, Figure 1-6, which show the running times of various runs. Each time represented is a representative mean time for a run through of that particular example.

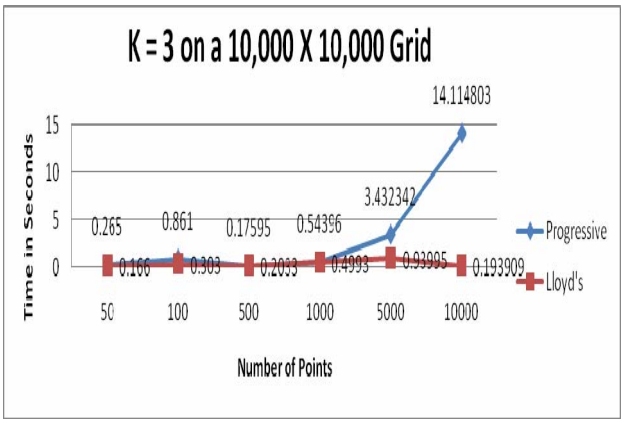

Fig. 1. Running time comparison when k=3.

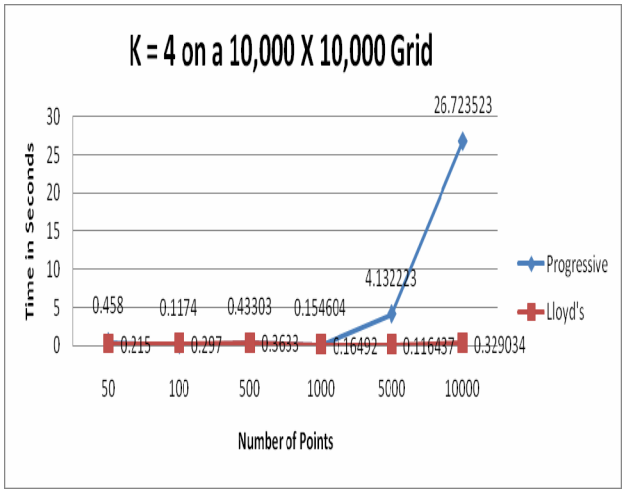

Fig 2. Running time comparison when k=4.

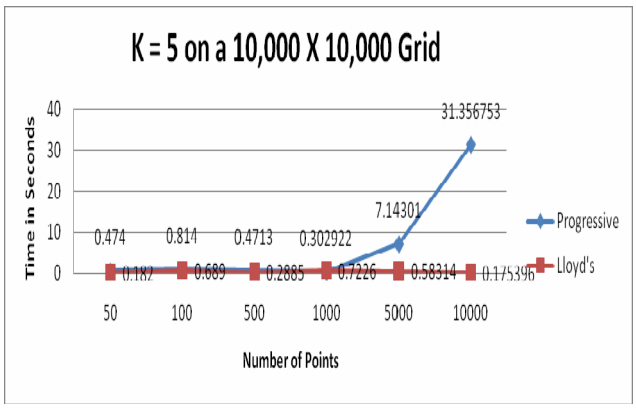

Fig. 3. Running time comparison when k=5.

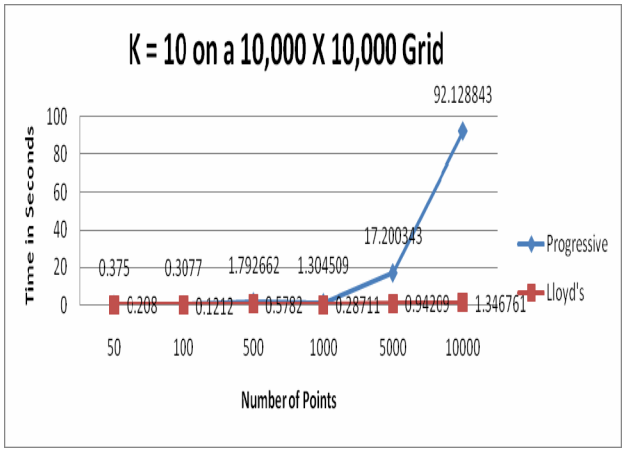

Fig. 4. Running time comparison when k=10.

Table 4. The MSD comparisons for the two approaches with diff. number of points when k=5.

K = 5	50 pts	100 pts	1000 pts	5000 pts	10000 pts
Lloyd min MSD	4078108.2	5222050.1	8116749.3	7997228.2	8089368.7
Lloyd max MSD	9622152.1	9250185.5	11439146.3	11259349.8	10874936.4
Lloyd global avg. MSD	**7141665.6**	**7423396.7**	**9603851.5**	**9915411.8**	**9803566.6**
Progressive min MSD	7069837.3	5749880.1	7538925.3	8031278.7	8070571.7
Progressive max MSD	10442232.1	10240177.6	11760624.0	11246938.1	10858752.9
Progressive global avg. MSD	**8609715.2**	**8395369.7**	**9715337.5**	**9915827.8**	**9803683.9**

Table 5. The MSD comparisons for the two approaches with diff. number of points when k=10.

K = 10	50 pts	100 pts	1000 pts	5000 pts	10000 pts
Lloyd min MSD	7453 82.5	26326 17.6	490460 7.3	482307 3.3	5014771.6
Lloyd max MSD	7349 801.8	69157 23.5	649697 0.1	598265 8.9	6097398.3
Lloyd global avg. MSD	**4354 746.8**	**48454 63.5**	**551908 2.1**	**553801 9.1**	**5587778.8**
Progressive min MSD	7453 82.5	26326 17.6	457760 1.7	460204 0.4	4713063.5
Progressive max MSD	9721 423.5	69699 95.3	622167 9.5	610552 3.9	6148742.0
Progressive global avg. MSD	**5265 944.6**	**49780 84.8**	**556412 6.5**	**550341 3.1**	**5547333.1**

Now comparisons of mean square differences will be made. Examples of the same types were conducted. The maximum and minimum local cluster mean square will be given, and also the general global average. Please refer to the Table 4 and Table 5.

4. Summary

Our experimental results are based on randomly distributed data. Data that have a more inherent relationship that are not randomly distributed would most likely yield different results from what has been found here. Other considerations must be made when considering which algorithm to use for different circumstances. That is something yet to be determined.

The advantage of Lloyd's algorithm compared to Progressive Greedy is clear from the comparisons. Based

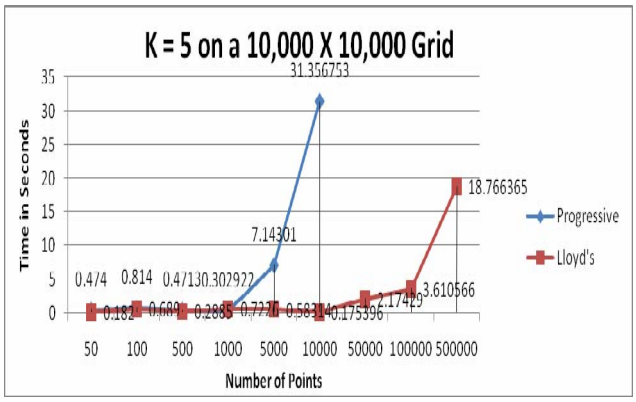

Fig. 5. Running time comparison when k=5 (includes the running times of Lloyd's algorithm when the number of points exceeds 10,000).

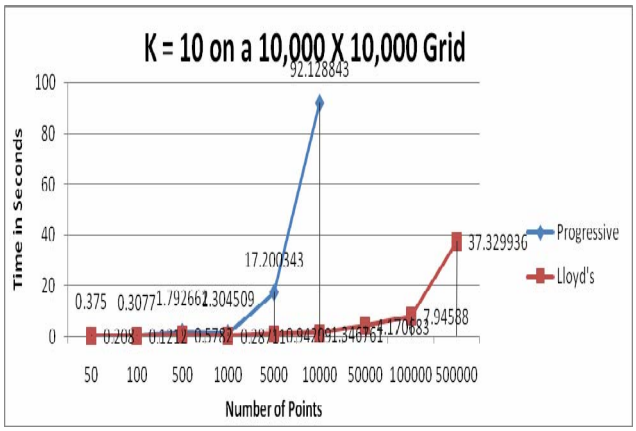

Fig. 6. Running time comparison when k=10 (includes the running times of Lloyd's algorithm when the number of points exceeds 10,000).

on our implementation, not just in time, but also in mean square difference, Lloyd's algorithm is more efficient. For very large data, Lloyd's algorithm definitely works faster. When the number of points exceeds 10,000 Progressive Greedy needs altering/optimization to handle the very large floating point values associated with finding the mean square difference.

5. References

[1] N. C. Jones and P. A. Pevzner, *An Introduction to Bioinformatics Algorithms*, the MIT Press, 2004.

[2] S. P. Loyd, "Least squares quantization in PCM", *IEEE Trans. on Infor. Theory*, 28, pp. 129-137, 1982.

[3] M. B. Eisten, P.T. Spellman, P. O. Brown, and D. Bostein, "Cluster analysis and display of genome-wide expression pattern," *Proc. Of National Academy of Sciences*, 95, pp. 14863-14868, 1998.