International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 5, July - 2012

An Efficient K-Means and C-Means Clustering Algorithm for Image Segmentation

Abstract

In this paper, we present a novel algorithm for performing k-means clustering. It organizes all the patterns in a k-d tree structure such that one can find all the patterns which are closest to a given prototype efficiently. The main intuition behind our approach is as follows. All the prototypes are potential candidates for the closest prototype at the root level. However, for the children of the root node, we may be able to prune the candidate set by using simple geometrical constraints. This approach can be applied recursively until the size of the candidate set is one for each node. Our experimental results demonstrate that our scheme can improve the computational speed of the direct k-means algorithm by an order to two orders of magnitude in the total number of distance calculations and the overall time of computation.

Keywords: Pattern recognition, machine learning, data mining, k-means clustering, nearest-neighbor searching, k-d tree, computational geometry, knowledge discovery.

1. Introduction

Clustering is the process of partitioning or grouping a given set of patterns into disjoint *clusters*. This is done such that patterns in the same cluster are alike and patterns belonging to two different clusters are different. Clustering has been a widely studied problem in a variety of application domains including neural networks, AI, and statistics. Several algorithms have been proposed in the literature for clustering: ISODATA [8, 3], CLARA [8], CLARANS [10], Focusing Techniques [5] P-CLUSTER [7]. DBSCAN [4], Ejcluster [6], BIRCH [14] and GRIDCLUS [12].

The k-means method has been shown to be effective in producing good clustering results for many practical applications. However, a direct algorithm of k-means method requires time proportional to the product of number of patterns and number of clusters per iteration. This is computationally very expensive especially for large datasets. We propose a novel algorithm for implementing the kmeans method. Our algorithm produces the same or comparable (due to the round-off errors) clustering results to the direct k-means algorithm. It has significantly superior performance than the direct k-means algorithm in most cases. The rest of this paper is organized as follows. We review previously proposed approaches for improving the performance of the k-means algorithms in Section 2. We present our algorithm in Section 3. We describe the experimental results in Section 4 and we conclude with Section 5.

2 k-means Clustering

In this section, we briefly describe the direct k-means algorithm [9, 8, 3]. The number of clusters _ is assumed to be fixed in k-means clustering. Let the _k prototypes be initialized to one of the input patterns Therefore,

wj=i1,j **€{**,……K},l€{the direct kmeans clustering ,…n}algorithm. Cj is the nth cluster whose value is a disjoint subset of input patterns. The quality of the clustering is determined by the following error function:

$$E = \sum_{j=1}^{k} \sum_{i_l \in C_j} |\ i_l - w_j\ |^2$$

The appropriate choice of _ is problem and domain dependent and generally a user tries several values of _. Assuming that there are _ patterns, each of dimension @, the computational cost of a direct k-means algorithm per iteration (of the repeat loop) can be decomposed into three parts:

1. The time required for the first *for* loop in Figure 1 is O(nkd)

2. The time required for calculating the centroids (second *for* loop in Figure 1) is O(nd)

3. The time required for calculating the error function is O(nd)

The number of iterations required can vary in a wide range from a few to several thousand depending on the number of patterns, number of clusters, and the input data distribution. Thus, a direct implementation of the k-means method can be computationally very intensive. This is especially true for typical data mining applications with large number of pattern vectors

```
function Direct-k-means()
  Initialize $k$ prototypes $(w_1, \ldots, w_k)$ such that $w_j = i_l,\ j \in \{1, \ldots, k\},\ l \in \{1, \ldots, n\}$
  Each cluster $C_j$ is associated with prototype $w_j$
  Repeat
      for each input vector $i_l$, where $l \in \{1, \ldots, n\}$, do
          Assign $i_l$ to the cluster $C_{j*}$ with nearest prototype $w_{j*}$
          (i.e., $|\ i_l - w_{j*}\ | \leq |\ i_l - w_j\ |,\ j \in \{1, \ldots, k\}$)
      for each cluster $C_j$, where $j \in \{1, \ldots, k\}$, do
          Update the prototype $w_j$ to be the centroid of all samples currently in $C_j$, so that $w_j = \sum_{i_l \in C_j} i_l / |\ C_j\ |$
      Compute the error function:
          $E = \sum_{j=1}^{k} \sum_{i_l \in C_j} |\ i_l - w_j\ |^2$
  Until $E$ does not change significantly or cluster membership no longer changes
```

Figure 1. Direct kmeans clustering algorithm

There are two main approaches described in the literature which can be used to reduce the overall computational requirements of the k-means clustering method especially for the distance calculations:

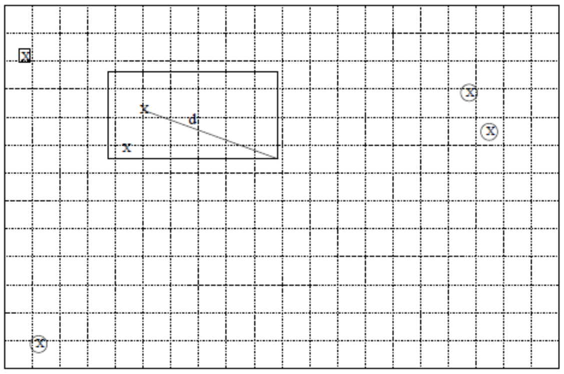
1. Use the information from the previous iteration to reduce the number of distance calculations. PCLUSTER is a k-means-based clustering algorithm which exploits the fact that the change of the assignment of patterns to clusters is relatively few after the first little iteration [7]. It uses a heuristic which determines if the closest prototype of a pattern E has been changed or not by using a simple check. If the assignment has not changed, no further distance

calculations are required. It also uses the fact that the movement of the cluster cancroids is small for consecutive iterations (especially after a few iterations).

2. Organize the prototype vectors in a suitable data structure so that finding the closest prototype for a given pattern becomes more efficient [11, 13]. This problem reduces to finding the nearest neighbor problem for a given pattern in the prototype space. The number of distance calculations using this approach is proportional to iteration. For many applications such as vector quantization, the prototype vectors are fixed. This allows for construction of optimal data structures to find the closest vector for a given input test pattern [11]. However, these optimizations are not applicable to the k-means algorithm as the prototype vectors will change dynamically. Further, it is not clear how these optimizations can be used to reduce the time for calculation of the error function (which becomes a substantial component after reduction in the number of distance calculations).

3 Our Algorithm

The main intuition behind our approach is as follows. All the prototypes are potential candidates for the closest prototype at the root level. However, for the children of the root node, we may be able to prune the candidate set by using simple geometrical constraints. Clearly, each child node will potentially have different candidate sets. Further, a given prototype may belong to the candidate set of several child nodes. This approach can be applied recursively till the size of the candidate set is one for each node. At this stage, all the patterns in the subspace represented by the subtree have the Using this approach, we expect that the number of distance calculation for the first loop (in Figure 1) will be proportional to where K –d Tree is much smaller than H-k Means This is because the distance calculation has to be performed only with internal nodes (representing many patterns)and not the patterns themselves in most cases. sole candidate as their closest prototype.



Example of pruning achieved by our algorithm. X represents the candidate set. D is the MinMax distance. All the candidates which are circled get pruned. The candidate with a square around it is not pruned by our algorithm

This approach can also be used to significantly reduce the time requirements for calculating the prototypes for the next iteration We also expect the time requirement for the second for loop to be proportional to The improvements obtained using our approach are crucially dependent on obtaining good pruning methods for obtaining candidate sets for the next level. We propose to use the following strategy's For each candidate _ 4, find the minimum and maximum distances to any point in the subspace Find the minimum of maximum distances, call out all candidates with minimum distance greater

4 Experimental Results

We have evaluated our algorithm on several datasets. We have compared our results with direct k-means algorithm interns of the number of performed distance calculations and the total execution time. A direct comparison with other algorithms(such as the P-Cluster [7] and [13]) is

not feasible due to availability of their datasets and software. However,
we present some qualitative comparisons. All the experimental results reported are on a IBM RS/6000 runningAIX version 4. The clock speed of the processor is 66 MHz and the memory size is 128 MByte.For each dataset and the number of clusters, we compute the factors K _means and HK means of reduction in distance
calculations and overall execution time over the direct algorithm respectively as well as the average number of distance calculations per pattern The number of distance calculations for the direct algorithm All time measurements are in seconds.
Our main aim in this paper is to study the computational aspects of the k-means method. We used several datasets all of which have been generated synthetically. This was done to study the scaling properties of our algorithm for different values of k and c respectively. For most of our datasets, we found that choosing a leaf size of 64 resulted in optimal or near optimal performance. Further, the overall performance was not sensitive to the leaf
size except when the leaf size was very small.

Dataset	Direct Alg	Our Algorithm			
		Total Time	FRT	FRD	ADC
DS1	115.100	6.830	16.85	64.65	1.01
DS2	114.400	7.430	15.39	50.78	1.28
DS3	115.900	6.520	17.77	66.81	0.97
R1	194.400	24.920	7.80	10.81	6.01
R2	705.745	49.320	14.30	10.80	6.02
R3	160.400	3.730	43.00	133.27	0.49
R4	323.650	5.270	61.41	224.12	0.29
R5	302.300	32.430	9.32	10.72	6.06
R6	606.00	63.330	9.56	10.83	6.00
R7	297.050	32.100	9.25	26.66	2.44
R8	448.750	31.980	14.03	26.66	2.44
R9	408.700	63.920	6.39	6.25	10.41
R10	822.450	132.880	6.18	5.86	11.09
R11	291.400	67.850	4.29	6.30	10.32
R12	585.300	133.580	4.38	6.07	10.72

Table present the performance of our algorithms for different number of clusters and iterations assuming a leaf size of 64. For each combination used, we present the factor reduction in overall time (FRT) and the time of the direct k-means algorithm. We also present the factor reduction in distance calculations (FRD) and the average number of distance calculations per pattern (ADC). These
Results show that our algorithm can improve the overall performance of k-means clustering by an order to two orders of magnitude. The average number of distance calculations required is very small and can vary anywhere from 0.17 to11.17 depending on the dataset and the number of clustersrequired.The results presented in [7] show that their methods result
in factor of 4 to 5 improvements in overall computational time. Our improvements are substantially better. However, we note that the datasets used are different and a direct comparison may not be accurate.

Dataset	k	Direct Alg	Our Algorithm			
			Total Time	FRT	FRD	ADC
DS1	16	6.140	1.510	4.06	26.69	0.64
DS2	16	6.080	1.400	4.34	34.47	0.49
DS3	16	6.010	1.370	4.38	35.68	0.48
R1	16	8.760	1.890	4.63	17.82	0.95
R2	16	17.420	3.130	5.56	27.38	0.62
R3	16	7.890	1.290	6.11	98.66	0.17
R4	16	16.090	2.750	5.85	53.02	0.32
R5	16	15.560	2.510	6.19	14.62	1.16
R6	16	31.200	4.480	6.96	17.77	0.96
R7	16	15.340	6.630	2.31	9.08	1.87
R8	16	22.200	6.800	3.26	9.08	1.87
R9	16	16.120	7.300	2.20	3.75	4.53
R10	16	33.330	11.340	2.93	4.96	3.43
R11	16	14.200	11.260	1.26	2.21	7.68
R12	16	28.410	22.110	1.28	2.21	7.68
DS1	64	23.020	2.240	10.27	54.72	1.19
DS2	64	22.880	2.330	9.81	43.25	1.50
DS3	64	23.180	2.340	9.90	52.90	1.23
R1	64	38.880	5.880	6.61	10.61	6.12
R2	64	141.149	11.770	11.99	10.64	6.11
R3	64	32.080	1.780	18.02	99.66	0.65
R4	64	64.730	3.090	20.94	139.80	0.46
R5	64	60.460	7.440	8.12	10.49	6.20
R6	64	121.200	14.670	8.26	10.65	6.10
R7	64	59.410	8.820	6.73	24.15	2.69
R8	64	89.750	8.810	10.18	24.15	2.69
R9	64	81.740	14.060	5.81	6.13	10.61
R10	64	164.490	28.640	5.74	5.82	11.17
R11	64	58.280	15.340	3.79	5.92	10.97
R12	64	117.060	29.180	4.01	5.85	11.10

The overall results for 10 iterations

5 Conclusions

In this paper, we presented a novel algorithm for performing-means clustering. Our experimental results demonstrated that our scheme can improve the direct k means algorithm by an order to two orders of magnitude in the total number of distance calculations and the overall time of computation. There are several improvements possible to the basic strategy presented in this paper. One approach will be to restructure the tree every few iterations to further reduce the value of The intuition here is that the earlier iterations provide some partial clustering information. This information can potentially be used to construct the tree such that the pruning is more effective. Another possibility is to add the optimizations related to incremental approaches presented in [7]. These optimizations seem to be orthogonal and can be used to further reduce the number of distance Calculations with using of C-Means Clustering

References

[1] K. Alsabti, S. Ranka, and V. Singh. An Efficient K-Means Clustering Algorithm. http://www.cise.ufl.edu1997.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. McGraw-Hill Book Company, 1990.

[3] R. C. Dubes and A. K. Jain. *Algorithms for Clustering Data*.Prentice Hall, 1988.

[4] M. Ester, H. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proc. of the 2nd Int'l Conf. on Knowledge Discovery and Data Mining*, August 1996.

[5] M. Ester, H. Kriegel, and X. Xu. Knowledge Discovery inLarge Spatial Databases: Focusing Techniques for Efficient
[6] J. Garcia, J. Fdez-Valdivia, F. Cortijo, and R. Molina. Dynamic Approach for Clustering Data. *Signal rocessing*,44:(2), 1994.

[7] D. Judd, P. McKinley, and A. Jain. Large-Scale Parallel Data Clustering. *Proc. Int'l Conference on Pattern Recognition*, August 1996.

[8] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data:an Introduction to Cluster Analysis*. John Wiley & Sons,1990.

[9] K. Mehrotra, C. Mohan, and S. Ranka. *Elements of ArtificialNeural Networks*. MIT Press, 1996.[10] R. T. Ng and J. Han. Efficient and Effective Clustering Methodsfor Spatial Data Mining. *Proc. of the 20th Int'l Conf. on Very Large Databases, Santiago, Chile*, pages 144–1551994.

[11] V. Ramasubramanian and K. Paliwal. Fast K-Dimensional Tree Algorithms for Nearest Neighbor Search with Application to Vector Quantization Encoding. *IEEE Transactions on Signal Processing*, 40:(3), March 1992.

[12] E. Schikuta. Grid Clustering: An Efficient Hierarchical Clustering Method for Very Large Data Sets. *Proc. 13thInt'l. Conference on Pattern Recognition*, 2, 1996.

[13] J. White, V. Faber, and J. Saltzman. United States Patent No.5,467,110. Nov. 1995.

[14] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases.*Proc. of the 1996 ACM SIGMOD Int'l Conf. on Management of Data, Montreal, Canada*, pages 103–114, June 1996.