ISSN:0975-9646

IC: Intelligent Clustering, a new time efficient data partitioning methodology

Done Stojanov [#1]

[#]*Faculty of Computer Science, University „Goce Delcev" - Stip*
Toso Arsov 14, Republic of Macedonia

Abstract: A new, time efficient data partitioning methodology is presented. First, the input data set is mapped into a set of radius vectors, which is further sorted, then partitioned into initial clusters. However, that is not a guarantee that all objects have been partitioned in the appropriate clusters. Therefore, objects being inappropriately partitioned are moved from one cluster into another neighbouring cluster. Finally, clusters of radius vectors are formed. Applying reverse mapping, clusters of radius vectors are back-mapped into clusters of original objects, being appropriately partitioned.

Keywords: sorting-based, time efficient, mapping, clustering methodology.

I. INTRODUCTION

Several clustering techniques have been used as solutions of the well-known data partitioning problem. Given a set of n objects, put the most similar objects into a common group (cluster). Clusters are disjoint sets, containing at least one object.

K-means clustering algorithm [1], presented by MacQueen in 1976, solves data partitioning problem. Being based on initial selection of k random objects as centroids, joining the rest of n-k objects to the nearest centroid, then recalculating the centre of each updated cluster, until centres convergence has been achieved, requires $O(nkI)$ time, where n is the number of objects being partitioned, k is the number of clusters, while I is the number of iterations. Best time performance results are obtained, if non-neighbouring initial centroids, being as far as possible distanced, are chosen.

However, k-means clustering is highly sensitive in terms of data outliers. Since the mean value for each attribute is recalculated, extreme values easily disrupt general data tendency. PAM (Partitioning Around Medoids) [2], is data outliers less sensitive clustering approach. The general idea behind PAM clustering is instead of recalculating the mean value for each attribute for all objects within a cluster, the most centred object within a cluster is used. Medoids are changed if that would result with a better data clustering, until medoids no further change has been achieved. PAM computational complexity is $O((n-k)^2kI)$. PAM effectively partitions small data sets. When large data sets are partitioned, PAM clustering is computationally more expensive than k-means.

Instead of applying PAM clustering on the whole data set, a set of samples from the original data set could be selected. For each sample PAM clustering is applied. That is the idea behind CLARA [3] clustering approach, requiring $O(ks^2+k(n-k))$ time, if samples of size s are chosen. However, CLARA partitioning result does not always match the best clustering, especially if the best clustering medoids are not among the samples being chosen.

CLARA clustering effectiveness is highly influenced by the sample size. CLARANS [4] clustering process is alike searching a graph of nodes, where each node (medoid) might be part of the optimal solution. During the search, medoids are swapped with non-medoids, if that would diminish clustering configuration cost. CLARANS time complexity is $O(kn^2)$.

In this paper, a new data partitioning methodology is presented. The best case time complexity of IC clustering is $O(n+T)$. The worst case time complexity is $O(n^2+T)$, where n is the number of objects being partitioned and T is the number of transitions between the neighbouring clusters.

II. IC METHODOLOGY

Data partitioning problem is considered. Given a set of n objects, $O=\{o_1,o_2,\ldots,o_{n-1},o_n\}$, where each object is represented with m properties, $o_i=(xi_1,xi_2,\ldots,xi_{m-1},xi_m)$, form k clusters, containing set O objects. Each object o_i is mapped into its corresponding radius vector, $R_i=(xi_1^2+xi_2^2+\ldots+xi_{m-1}^2+xi_m^2)^{1/2}$, resulting with a set of radius vectors, $R=\{R_1,R_2,\ldots,R_{n-1},R_n\}$. Each mapping $\boldsymbol{f}$: $O\rightarrow R$ is tracked by a tuple (R_i,i), where R_i is object's o_i corresponding radius vector.

Afterwards R is sorted, being transformed into non-decreasing ordered set, $\boldsymbol{s}$: $R\rightarrow Rs$, $Rs=\{Rs_1,Rs_2,\ldots,Rs_{n-1},Rs_n\}$, $Rs_i \leqslant Rs_{i+1}$, for $1\leqslant i\leqslant n-1$.

Sorted radius vectors set Rs is partitioned into k clusters, distributing data initially as given bellow:

Cluster 1: $c_1=\{Rs_1,Rs_2,\ldots,Rs_{[n/k]}\}$
Cluster 2: $c_2=\{Rs_{[n/k]+1},Rs_{[n/k]+2},\ldots,Rs_{2[n/k]}\}$
…
Cluster k-1: $c_{k-1}=\{Rs_{(k-2)[n/k]+1},Rs_{(k-2)[n/k]+2},\ldots,Rs_{(k-1)[n/k]}\}$
Cluster k: $c_k=\{Rs_{(k-1)[n/k]+1},Rs_{(k-1)[n/k]+2},\ldots,Rs_n\}$

The mean value for each cluster, $mc_i=(\sum Rs_i \epsilon c_i)/|c_i|$, where $|c_i|$ is the number of elements in cluster c_i, is calculated.

Nevertheless, there might be radius vectors initially partitioned into cluster c_i, which are closer to cluster c_{i+1} and radius vectors partitioned into cluster c_{i+1}, which are closer to cluster c_i. Cluster c_i radius vectors being closer to cluster c_{i+1}, are moved into cluster c_{i+1}. Accordingly, cluster c_{i+1} radius vectors being closer to cluster c_i, are moved into cluster c_i, for $1\leqslant i\leqslant k-1$. In these constellations, also is true that if radius vector Rs_x belongs to cluster c_i, then radius vector Rs_y, $Rs_y \leqslant Rs_x$, certainly does not belong to cluster

c_{i+1}. Based on the previous, radius vectors are appropriately partitioned into k different clusters, by the following code:

```
i=1
while(i≤k-1) do {
set index1 to the current value of i
set index2 to 0
while((|Rs_index1×[n/k]-index2-mc_i+1|<|Rs_index1×[n/k]-index2-mc_i|)
and (|c_i|>1))
{
move Rs_index1× [n/k]-index2 into cluster c_i+1
increase index2 for 1
recalculate clusters' c_i and c_i+1 mean values
}
set index2 to 1
while((|Rs_index1×[n/k]+index2-mc_i|<|Rs_index1×[n/k]+index2-mc_i+1|)
and (|c_i+1|>1))
{
move Rs_index1× [n/k]+index2 into cluster c_i
increase index2 for 1
recalculate clusters' c_i and c_i+1 mean values
}
increase i for 1
}
```

At this stage, a set of k clusters is formed. Applying reverse mapping $\boldsymbol{f_r}$: R→O, passing from the domain of radius vectors in the domain of objects, being possible by using mapping data information from mapping tuples (R_i,i), each cluster of radius vectors is translated into a cluster of objects, back-mapping radius vectors R_i into the corresponding objects o_i, contained in the original data set O.

III. An example, demonstrating intelligent clustering

Data set of 11 two attributed objects, O={(1,2), (3,4), (7,5), (2,2), (3,3), (1,1), (7,8), (8,8), (9,7), (5,5), (4,4)}, is partitioned into k=3 clusters, according to the presented methodology. First, data set O is mapped into corresponding set of radius vectors, R={2.236, 5, 8.602, 2.828, 4.243, 1.414, 10.63, 11.314, 11.402, 7.071, 5.657}. Sorting set R, non-decreasing ordered set of radius vectors Rs is obtained, Rs={1.414, 2.236, 2.828, 4.243, 5, 5.657, 7.071, 8.602, 10.63, 11.314, 11.402}. Each mapping $\boldsymbol{f}$: O→R is tracked by a mapping tuple. For example, mapping tuple (1.414, 6), indicates that set O object at position 6 corresponds radius vector of 1.414. Sorted data is partitioned into k=3 clusters.

Mapping tuples are: (1,414, 6), (2.236, 1), (2.828, 4), (4.243, 5), (5, 2), (5.657, 11), (7.071, 10), (8.602, 3), (10.63, 7), (11.314, 8), (11.402, 9). The following initial clusters are formed:

Cluster 1: c_1={1.414, 2.236, 2.828}
Cluster 2: c_2={4.243, 5, 5.657}
Cluster 3: c_3={7.071, 8.602, 10.63, 11.314, 11.402}

The last element in cluster c_1 is most likely to be inappropriately partitioned. Therefore, the distances between 2.828 and clusters' c_1 and c_2 mean values are calculated. Distance results are: $d(2.828,mc_1)$=|2.828-2.159|=0.669, $d(2.828,mc_2)$=|2.828-4.967|=2.139. Being cluster c_1 less distanced than cluster c_2, 2.828 and the remaining elements in cluster c_1 have been appropriately partitioned, since none of them is cluster c_2 less distanced than 2.828.

The first element in cluster c_2 is most likely to be inappropriately partitioned. Therefore, the distances between 4.243 and clusters' c_1 and c_2 mean values are calculated. Distance results are: $d(4.243,mc_1)$=|4.243-2.159|=2.084, $d(4.243,mc_2)$=|4.243-4.967|=0.724. Interpreting distance results, the uncertainty in terms of which cluster 4.243 belongs is completely eliminated. The first element in cluster c_2 is part of cluster c_2. The remaining elements in cluster c_2 can't be part of cluster c_1, because none of them is cluster c_1 less distanced than 4.243.

Applying the same logic, we get that 5.657 has been appropriately partitioned in cluster c_2. When considering the first element in cluster c_3, we get that 7.071 is closer to cluster c_2, $d(7.071,mc_2)$=|7.071-4.967|=2.104, $d(7.071,mc_3)$=|7.071-9.804|=2.733, causing radius vector 7.071 rearrangement, being moved into cluster c_2.

Since clusters c_2 and c_3 have been changed, clusters new mean values are recalculated. Cluster c_2 new mean value is 5.493, while cluster c_3 new mean value is 10.487. After the rearrangement, 10.63 is the first element in cluster c_3. Analysing distance results between 10.63 and clusters' c_2 and c_3 new mean values: $d(10.63,mc_2)$=|10.63-5.493|=5.137, $d(10.63,mc_3)$=|10.63-10.487|=0.143, is more than obvious that 10.63 belongs to cluster c_3. Certainly, since none of the remaining elements in cluster c_3 is cluster c_2 less distanced than 10.63, they are all part of cluster c_3.

At this stage the following clusters have been obtained:

Cluster 1: c_1={1.414, 2.236, 2.828}
Cluster 2: c_2={4.243, 5, 5.657, 7.071}
Cluster 3: c_3={8.602, 10.63, 11.314, 11.402}

Applying reverse mapping $\boldsymbol{f_r}$: R→O, clusters of radius vectors are back-mapped into objects, all of them being appropriately partitioned:

Cluster 1: c_1={(1,1), (1,2), (2,2)}
Cluster 2: c_2={(3,3), (3,4), (4,4), (5,5)}
Cluster 3: c_3={(7,5), (7,8), (8,8), (9,7)}

IV. Time complexity analysis

Intelligent and k-means clustering have been implemented in C++. The running time of these techniques, for sets with different length, have been measured on Acer Aspire 5570Z computer, with Genuine Intel CPU at 1GHz and 1.5 GB RAM. The time complexity improvement of the intelligent clustering over the k-means algorithm is evident – TABLE I, Fig. 1, especially if the objects' set of radius vectors is partially or fully sorted. For example, the running time of the intelligent clustering for partitioning the first data set (TABLE II first row): (1,1), (2,2), (3,3), (4,4), (6,5), (5,5), (4,6), (8,8), (7.5,9),(7.5,8.5), into three clusters is 0.016 seconds, while the running time of the k-means algorithm, if (1,1), (4,4) and (7.5,8.5) have been selected as initial centroids, is 0.078 seconds. The significant time complexity improvement in this case is due to the structure of the set of radius vectors, which is partially sorted. In the case of the third data set (TABLE III third row), when: (1,1), (2,2), (3,3), (4,4), (6,5), (5,5), (4,6), (8,8), (7.5,9), (7.5,8.5), (3,2), (4,8), (5,5.5), (6,6.7), (7,7.5), (0.5,0.5), (0.7,0.7), (1.2,1.2), (1.35,1), (1,1.8), are partitioned into three clusters, if (1,1), (4,4), (7.5,8.5) have been also

selected as initial centroids, intelligent clustering running time equals k-means running time, 0.094 seconds. This is due to the low sorting degree of the set of radius vectors, i.e. more time is wasted on sorting.

TABLE IV
COMPARING K-MEANS AND INTELLIGENT CLUSTERING TIME PERFORMANCE

Data set	Time performance analysis		
	Length of the data set	k-means running time (s)	Intelligent clustering running time (s)
1	10	0.078	0.016
2	15	0.109	0.078
3	20	0.094	0.094
4	25	0.234	0.124
5	30	0.219	0.156
6	35	0.202	0.172
7	40	0.203	0.203
8	45	0.219	0.218

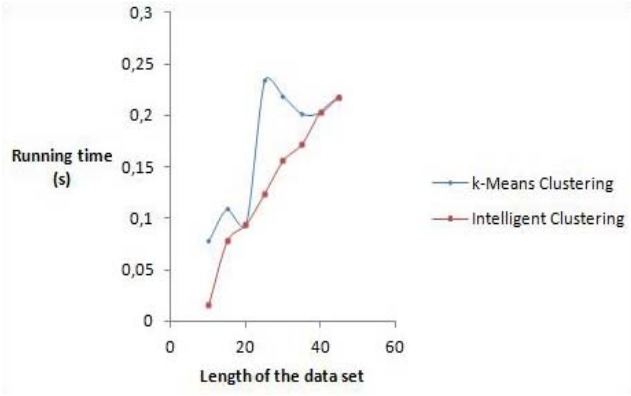

Fig. 1 Time performance graphs

Intelligent clustering best case time complexity is $O(n+T)$, where n is the number of objects being partitioned, while T is the number of transitions between the adjacent clusters, assuming that the set of radius vectors is approximately or fully sorted. Intelligent clustering worst case time complexity is $O(n^2+T)$, if the set of radius vectors is far from being sorted.

V. CONCLUSIONS

A new time efficient data partitioning methodology has been presented. When compared with k-means clustering choosing initial centroids as far as possible, intelligent clustering demonstrated better time performance. Since k-means clustering approach, especially if initial centroids are chosen as far as possible, is faster than PAM clustering technique, a conclusion for superior time performance over PAM clustering can be easily deduced. Compared with CLARA clustering approach, which is extremely fast, intelligent clustering results is always the best one, what is not always a case with CLARA.

REFERENCES

[1] J. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations," in *Proc. of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281-297.

[2] L. Kaufman and P. Rousseeuw, "Clustering by means of medoids," in *Statistical Data Analysis Based on the L1 Norm*, 1987, pp. 405-416.

[3] L. Kaufman and P. Rousseeuw, *Finding Groups in Data, An Introduction to Cluster Analysis*, 99th ed., New Jersey, USA: Willey-Interscience, 1990.

[4] R. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," in *Proc. of the 20th VLDB Conference*, 1994, pp. 144–155.