Active Distance-Based Clustering using K-medoids

Amin Aghaee, Mehrdad Ghadiri, and Mahdieh Soleymani Baghshah

{aghaee,ghadiri}@ce.sharif.edu
soleymani@sharif.edu

Abstract. k-medoids algorithm is a partitional, centroid-based clustering algorithm which uses pairwise distances of data points and tries to directly decompose the dataset with n points into a set of k disjoint clusters. However, k-medoids itself requires all distances between data points that are not so easy to get in many applications. In this paper, we introduce a new method which requires only a small proportion of the whole set of distances and makes an effort to estimate an upper-bound for unknown distances using the inquired ones. This algorithm makes use of the triangle inequality to calculate an upper-bound estimation of the unknown distances. Our method is built upon a recursive approach to cluster objects and to choose some points actively from each bunch of data and acquire the distances between these prominent points from oracle. Experimental results show that the proposed method using only a small subset of the distances can find proper clustering on many real-world and synthetic datasets.

Keywords: Active k-medoids, Active clustering, Distance-based clustering, Centroid-based clustering.

1 Introduction

As the production of data is expanding at an astonishing rate and the era of big data is coming, organizing data via assigning items into groups is inevitable. Data clustering algorithms try to find clusters of objects in such a way that the objects in the same cluster are more similar to each other than to those in other clusters. Nowadays clustering algorithms are widely used in data mining tasks.

There are different categorization for clustering algorithms, e.g., these algorithms can be categorized into Density-based, Centroid-based and Distribution-based methods. In centroid-based clustering methods, each cluster is shown by a central object. This object which can be a member of the dataset denotes a prototype of the whole cluster. When these algorithms are appointed to find K clusters, they usually find K central objects and assign each element to the nearest centroid. As they go on, they attempt to decrease the energy and total error of clusters by finding better central elements. *K-medoids* and *K-means* are the two most popular centroid-based algorithms. Although, they both partition the data into groups such that the sum of the squared distances of the data points to

the nearest center to them is minimized, they have different assumptions about centroids. Indeed, the k-medoids algorithm chooses the centroids only from the data points and so these centroids are members of the whole dataset while k-means algorithm can select the centroids from the whole input space.

In [3], some usages and applications of k-medoids algorithm are discussed. According to this study, in resource allocation problems, when a company wants to open some branches in a city, in such a way that the average distance from each residential block to the closest branch is intended to be minimized, the k-medoids algorithm is a proper option. Additionally, in mobile computing context, it is an issue to save communication cost when devices need to choose super-nodes among each other, which should have minimum average distances to all devices and k-medoids can solve this problem. Furthermore, as reported by [3], medoid queries also arise in the sensor networks and many other fields.

Active learning is a machine learning field that have bee attended specially in the last decade. Until now, many active methods for supervised learning that intend to select more informative samples to be labeled have been proposed. Active unsupervised methods have also been received attention recently. In an unsupervised learning manner, finding the similarities or distances of samples from each other may be difficult or infeasible. For example, the sequence similarity of proteins [23] or similarity of face images [7] which needs to be obtained from human as an oracle, may be difficult to be responded. The active version of some of the well known clustering algorithms have been recently presented in [14, 26].

In this paper, we propose the active k-medoid algorithm that inquires a subset of pairwise distances to find the clustering of data. We use a bottom-up approach to find more informative subset of the distances to be inquired. Extensive experiments on several data sets show that our algorithm usually needs a few percentage of the pairwise distances to cluster data properly.

In the rest of this paper, we first discuss about the works that have been done in the field of active clustering in Section 2. In Section 3, we introduce our algorithm. The result of experiments on different datasets have been presented in Section 4. At last, we discuss about some aspects of our algorithm and conclude the paper in Section 5.

2   Related Work

Active learning is a machine learning paradigm that endeavors to do learning with asking labels of a few number of samples which are more important in the final result of learning. Indeed, most of supervised learning algorithms need a large amount of labeled samples and gathering these labeled samples may need unreasonable amount of time and effort. Thus, active learning tries to ask labels for more important samples where important samples may be interpreted as most informative ones, most uncertain ones, or the ones that have a large effect in the results [18]. The active clustering problem has been recently received much attention. Until now, the active version of some well known clustering methods

has been proposed in [14, 26]. In the active clustering problem, a query is a pair of data whose similarity must be determined. The purpose of the active learning approach is reducing the number of required queries via active selection of them instead of random selection [21].

The existing active clustering methods can be categorized into constraint-based and distance-based ones [24] . In the most of the constraint-based methods, must-link and cannot-link constraints on pairs of data points indicating these pairs must be in the same cluster or different clusters are inquired. Some constraint-based methods for active clustering have been proposed in [24, 28, 11, 26, 7, 6, 25]. In distance-based methods, the response to a query on a pair of data points is the distance of that pair according to an objective function. Distance-based methods for active clustering have been recently attended in [14, 10, 13, 19, 27, 23].

In [14], an algorithm for active DBSCAN clustering is presented. In this algorithm, the distances that have not been queried are estimated with a lower bound. A score indicating the amount and the probability of changes in the estimated distances by asking a query is used to select queries. Moreover, an updating technique is introduced in [14] that update clustering after a query.

In [19, 27], distance-based algorithms are presented for active spectral clustering in which a perturbation theory approach is used to select queries. A constraint-based algorithm has also been presented in [26] for active spectral clustering that uses an approach based on maximum expected error reduction to select queries.

An active clustering method for k-median clustering has also been proposed in [23]. This method selects some points as the landmarks and ask the distances between these landmarks and all the other data points as queries. Finally, k-median clustering is done using these distances.

3   Proposed Method

In this section, we introduce the proposed active k-medoids clustering. We assume that our algorithm intends to partition n samples into k different clusters. As mentioned above, many clustering algorithms such as K-medoids, PAM [12], and some other distance-based methods, calculate an $n \times n$ distance matrix at first and perform the clustering algorithm on this distance matrix. We show the distance matrix by D where d_{ij} denotes the distance between the ith sample and the jth one.

We introduce a method to estimate unknown distances during an active clustering process. In a metric space, a satisfying and eminent upper-bound estimation for any distance metric can be obtained by the triangle inequality. For example, when we know the exact distances between d_{ax}, d_{xy} and d_{yb}, we can determine the upper-bound estimation for d_{ab} as:

$$d_{ab} \leqslant d_{ax} + d_{xy} + d_{yb} \qquad (1)$$

We find an upper-bound estimation of the distances using the triangle inequality and the known distances asked from an oracle already. Therefore, we have

$$\forall i, j, 1 \leqslant i, j \leqslant n : D(i,j) \leqslant D_e(i,j) \tag{2}$$

where D_e shows the estimated distances.

First, upper-bound estimations for all distances are infinity and we update these distances by asking some of them and make better estimations for the other unknown distances using the triangle inequality and new distances taken from the oracle. The update will be done by replacing exact values for the asked distances and getting the minimum of the old and the new upper-bound estimation for unknown distances. By asking the landmark distances, we intend to take a better estimation of distances required for the k-medoids algorithm.

Consider some data points which are partitioned to m groups where the distances within each group are known or estimated. However, the distances between data points from different groups are unknown. Our goal is to estimate these unknown distances instead of asking them. In such situation, we can choose t finite points from each group and ask the distances between these mt points between different groups and estimate the other distances using these asked distances. The number of these distances is $\binom{m}{2}t^2$. Figure 1 gives an intuition about this distance estimation method. We want to estimate the distance between a and b and the distances between those points that are connected by solid lines and dotted lines are known.
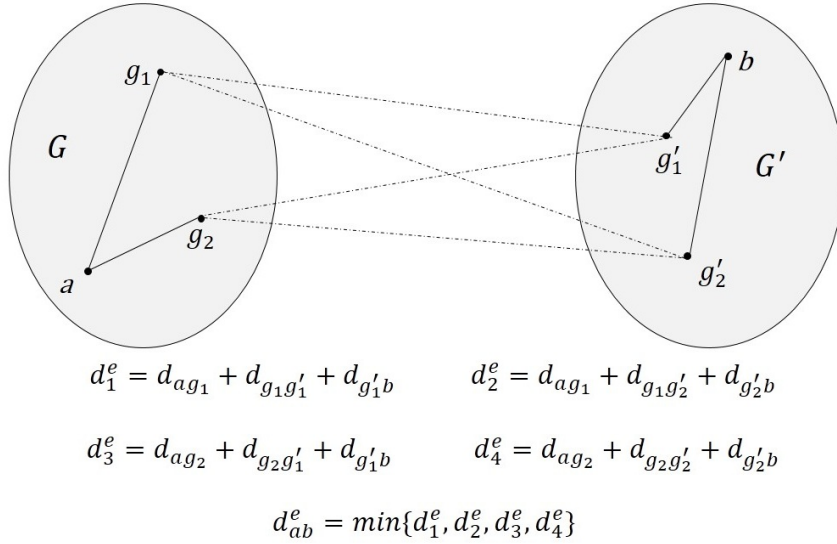

$$d_1^e = d_{ag_1} + d_{g_1g_1'} + d_{g_1'b} \qquad d_2^e = d_{ag_1} + d_{g_1g_2'} + d_{g_2'b}$$

$$d_3^e = d_{ag_2} + d_{g_2g_1'} + d_{g_1'b} \qquad d_4^e = d_{ag_2} + d_{g_2g_2'} + d_{g_2'b}$$

$$d_{ab}^e = min\{d_1^e, d_2^e, d_3^e, d_4^e\}$$

Fig. 1. Upper-bound distance estimation between a, b. Superscript e determines that the distance is an estimation.

Such estimation algorithm can be done in $\mathrm{O}(n^2 t^2)$, when n is the number of data points. If we have $t \ll n$, the time complexity of the algorithm will be $\mathrm{O}(n^2)$.

Based on the estimation method used for unknown distances, we present an active k-medoids algorithm. The approach is based on partitioning the data points into some groups by a hierarchical manner. In the other words, we partition data points into b groups and partition each of these groups to b groups and so on until we get to a threshold like t_h for the number of data points in each partition. In this level, we ask all the distances within each group among all its data points and choose t points from each group (to ask their distances) and using the explained algorithm for estimating distances in a bottom-up approach until we get to the highest level. After that, we cluster the data using the k-medoids algorithm on the estimated distances. According to these explanation, it seems that choosing t points in each group is a critical step and choosing a bad point can lead to unfavourable estimations. For this purpose, consider a group of data points like G which its inner distances are known or estimated. In order to choosing t points, we perform k-medoids algorithm on G and find clusters and medoids for this group. Then, we choose medoids and s random points from each cluster as the points whose distances are needed to be asked. Therefore, the number of the chosen data points in G will be $t = k(s + 1)$. It is obvious that a greater s will lead to more accurate estimations.

Algorithm 1 present the pseudo code of the proposed method. This function clusters n data points into k different categories and return clusters of data. Here, b shows the branching factor which is used to partition data points to b different groups with the same size. The partitioning algorithm will perform for each group recursively. There is also a threshold t_h which clarify the minimum size of a group of data points. Clearly, $t_h \geq k(s + 1)$ since we need to choose at least $k(s+1)$ points in each group. It is also noteworthy that if $n \leq 2t_h$, we need to ask all distance pairs since these data points cannot be partitioned.

Figure 2 shows an example workflow for Algorithm 1 for 1600 data points with branching factor 2 and threshold 400.

Now we calculate the complexity of our Algorithm 1. It makes a tree with the branching factor b and the threshold t_h for the number of the data points in the leaves of the tree. Therefore, the height of this tree is $\lceil \log_b (n/t_h) \rceil$. The number of nodes in the ith level of the tree is b^i and each node of the ith level has n/b^i data points. According to [20], the time complexity of the k-medoids algorithm is $\mathrm{O}(kn^2)$ for n data points and k clusters in each iteration. Consider p as the maximum number of iterations used in the k-medoids algorithm, then the time complexity in each node in the ith level of the tree is $\mathrm{O}((n^2/b^{2i})kp)$. Therefore, the overall complexity of Algorithm 1 is

$$\mathrm{O}\Big(\sum_{i=1}^{\lfloor \log_b (n/t_h) \rfloor + 1} (\frac{n^2}{b^{2i}} kp) b^i \Big) = \mathrm{O}(n^2 kp). \tag{3}$$

A major factor that measures the quality of an active clustering algorithm, is the number of distances that the algorithm demands from the oracle. The

---

Algorithm 1 Active k-medoids

INPUT: D_e, n, k, b, t_h ▷ distance estimation, #data, #clusters, branch factor, threshold

OUTPUT: $C_1, \ldots, C_k$

 1: **procedure** ACTIVEKMEDOIDS(D_e, n, k, b, t_h)
 2:    **if** $n \le 2t_h$ **then**
 3:        Update D_e by querying all distances
 4:        $C_1, \ldots, C_k \leftarrow kmedoids(D_e, k)$ ▷ kmedoids function is a regular k-medoids
 5:        **return**
 6:    **end if**
 7:    Partition data to b equal size groups, like $G_1, \ldots, G_b$
 8:    **for** i from 1 to b **do**
 9:        $T_1, \ldots, T_k \leftarrow ActiveKmedoids(D_e(G_i), |G_i|, k, b, t_h)$ ▷ $D_e(G_i)$ is the part of the estimated distance matrix corresponding to G_i
10:        $G_i^c \leftarrow$ medoids of $T_1, \ldots, T_k$ and s random points from each of them.
11:    **end for**
12:    Update D_e by querying distances between all those pairs that one of them is in G_i^c and the other is in G_j^c.
13:    Update D_e by the triangle inequality and new inquired distances.
14:    $C_1, \ldots, C_k \leftarrow kmedoids(D_e, k)$
15: **end procedure**

---

number of the asked queries in the internal nodes of the tree is $\mathrm{O}\big(b^2(k(s+1))^2\big)$ and in the leaves is $\mathrm{O}\big(nt_h\big)$. Since, in the proposed method, there are n/t_h leaves and each of them has t_h data points, the ratio of the asked distances to all of the distances is almost equal to

$$(b^2(k(s+1))^2 \frac{b^{\lfloor \log_b (n/t_h) \rfloor} - 1}{b - 1} + nt_h) / \binom{n}{2} \tag{4}$$

where $(b^{\lfloor \log_b (n/t_h) \rfloor} - 1)/(b - 1)$ shows the approximate number of the internal nodes in the tree.

In order to improve the accuracy of the estimated distances, we can increase the s value. However, it may raise running time of calculating upper-bound estimations and the number of queries that should be asked.

To have a good baseline for comparison, we introduce an algorithm based on random selection of the distance queries, called *Random-Rival* and compare results of our algorithm with those of this algorithm. In Section 4, we show that asking queries using our method is much better than asking them randomly that is the aim of any active clustering algorithm.

Random-Rival(RR) algorithm, considers data points as the vertices of a weighted graph in which the weight of each edge shows the distance between its endpoints. Firstly, RR asks some distances randomly and then estimates all unknown distances using the triangle inequality and *Floyd-Warshall* [9, p. 693] algorithm. In the other words, we find shortest path distances of all pairs using the available distances in the aforementioned graph. Since our distance function
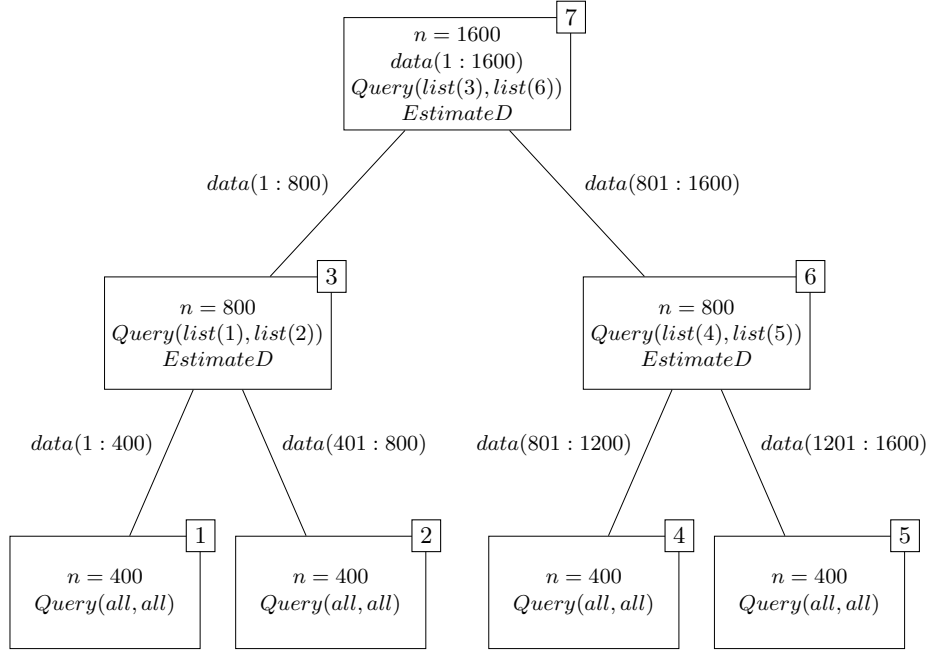
Fig. 2. ActiveKmedoids workflow for 1600 points($b = 2$, $t_h = 400$)

is a distance metric, the length of these shortest paths is an upper-bound estimation of the true distances. Finally, RR clusters data runs the k-medoids algorithm on the estimated distances. Since Floyd-Warshall worst case time complexity is $O(n^3)$ [9, p. 695], we can consider the runtime complexity of RR as $O(n^3)$ at worst. Algorithm 2 shows the pseudo code of the RR algorithm.

Algorithm 2 Random Rival
___
INPUT: n, k, B ▷ #data, #clusters, budget
OUTPUT: $C_1, \ldots, C_k$
 1: **procedure** RANDOMRIVAL(n, k, B)
 2:      $D \leftarrow n \times n$ infinity matrix
 3:      $(x_1, y_1), \ldots, (x_B, y_B) \leftarrow$ random pairs such that $1 \leqslant x_i < y_i \leq n$
 4:      $\forall i, 1 \leqslant i \leqslant B :$ update $D(x_i, y_i)$ by querying distances
 5:      $D_e \leftarrow FloydWarshall(D)$
 6:      $C_1, \ldots, C_k \leftarrow kmedoids(D_e, k)$
 7: **end procedure**
___

4   Empirical results

In this section, we show the results of our algorithm on some synthesized and real world datasets. General information about these datasets are presented in Table 1. Most of them are real world datasets, but some are synthesized which are marked with letter s in Table 1. NORM-10 [4] contains 10000 data points having 20 features. This dataset has been generated by choosing 10 real centers uniformly at random from the hypercube of side length 50. Then, for each of the real centers, 1000 points from a Gaussian distribution of variance one centered at the corresponding point is generated. We converted samples in NEC_animal [15] and ALOI200 [1] dataset into 32×32 grayscale images. ALOI [1] (Object Viewpoint version) has 1000 classes but we use only 200 classes of it.

Table 1. General information about datasets

Data	#Samples	#Features	#Classes	#Distances	Ref.
vary-density(s)	150	2	3	11175	[2]
seeds	210	7	3	21945	[5]
mouse(s)	500	2	4	124750	[2]
fisheriris	150	4	3	11175	[5]
data_2000(s)	2000	2	5	1999000	[8]
Trace	200	275	4	19900	[29]
multi-features	2000	649	10	1999000	[5]
TwoDiamonds(s)	800	2	2	319600	[22]
EngyTime(s)	4096	2	2	8386560	[22]
COIL100	7200	1024	100	25916400	[16]
NORM10(s)	10000	20	10	49995000	[4]
NEC_animal	4371	1024	60	9550635	[15]
ALOI200	14400	1024	200	103672800	[1]

Although active version of some clustering algorithms like DBSCAN and spectral clustering have been introduced in [14, 26], these clustering algorithms are substantially different from the k-medoids algorithm. For example, DBSCAN and spectral clustering methods can find clusters of different shapes while k-medoids cannot. Thus, we cannot compare results of our active k-medoid with those of active DBSCAN and active spectral clustering methods. One way to evaluate an active clustering method that asks distances is to compare it with a clustering method that asks a random subset of distances. , we compare our method with the Random-Rival algorithm introduced in section 3 that tries to use a random subset of distances to estimate the whole distance matrix (using shortest paths on the graph of data points and the triangle inequality). It must be mentioned that in both the proposed active k-medoid and the Random-Rival algorithm, the clustering algorithm that is run on the obtained distance matrix will be k-medoids. One of the most common measures for comparison of clustering algorithms is *normalized mutual information* (NMI) [17]. This measure

shows the agreement of the two assignments, ignoring permutations. In the other words, NMI for the clustering obtained by an algorithm shows the agreement between the obtained grouping by this algorithm and the ground truth grouping of data.

We run our algorithm with $s = 1, 3$ and branching factor $b = 2$ over all the datasets. Threshold t_h is set to the minimum possible value which is equal to the number of classes for each dataset. Greater value of s, branching factor b, or threshold t_h can improve NMI score for some datasets. However, it would also increase the number of queries which is usually quite unsatisfactory. We also run Random-Rival over these datasets which requires a specified proportion of distances starting from 5% to 100% (with the step 5 percent).

Results of our method with the parameters mentioned in the previous paragraph are shown in Table 2. For each algorithm and each dataset, NMI score and the ratio of the asked distances are presented in the table cells. For the Random-Rival algorithm, the results for the proportion of distances (between 5% and 100%) that is the first place where the number of inquired distances is greater than or equal to the inquired ones in our method are reported.

Table 2. NMI results of the methods. Numbers in parenthesis show percent of the asked distances.

Data	RR	s = 1	s = 3
vary-density	70.8 (10.0%)	95.0 (9.7%)	96.6 (10.5%)
seeds	55.7 (10.0%)	90.3 (7.1%)	89.5 (7.3%)
mouse	58.1 (5.0%)	75.5 (4.1%)	73.6 (4.3%)
fisheriris	65.3 (10.0%)	85.6 (9.6%)	89.3 (10.2%)
data_2000	88.5 (5.0%)	77.1 (1.9%)	78.3 (1.9%)
Trace	45.7 (10.0%)	51.2 (9.6%)	52.4 (10.4%)
multi-features	46.9 (5.0%)	78.7 (2.8%)	77.6 (2.8%)
TwoDiamonds	97.5 (5.0%)	100.0 (1.8%)	100.0 (1.8%)
EngyTime	71.2 (5.0%)	99.9 (1.6)	99.6 (1.6%)
COIL100	42.3 (10.0%)	75.6 (7.6 %)	75.6 (8.0%)
NORM10	95.5 (5.0%)	94.3 (1.6%)	94.6 (1.6%)
NEC_animal	23.6 (10.0%)	66.6 (7.6%)	67.3 (7.9%)
ALOI200	48.0 (10.0%)	79.6 (7.7%)	79.7 (8.0%)

Moreover, results of the Random-Rival algorithm which uses 0 percent of distances up to 100% of them is presented for some datasets in figure 3. According to figure 3, RR shows an ascending trend by asking extra distances and it gets close to the maximum value by asking about 20% of distances. Although it sounds to be a good algorithm, the proposed active k-medoids algorithm gets better NMI values with asking fewer number of distances. Moreover, the time complexity of our active k-medoid is also better. These results state the power of our algorithms which find accurate clusters by asking only a small subset of distances.
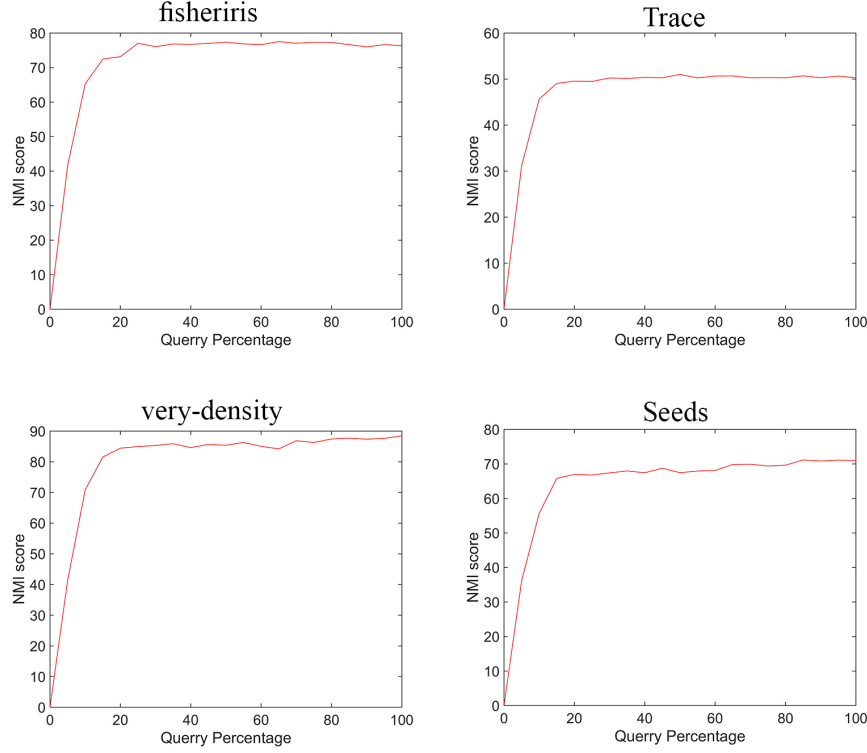
Fig. 3. Random-Rival over four datasets.

5   Conclusion

In this paper, we introduce an innovative active distance-based clustering method. Its goal is to cluster n points from a metric dataset into k clusters by using lowest number of distances that is possible. We design a recursive model that makes a tree and split data with a branching factor b unless the number of objects is less than a threshold t_h. Then, it actively selects and ask some pairwise similarities from and oracle. After that it tries to make an upper-bound estimation for unknown distances utilizing triangular inequality. Eventually, it clusters data with a simple k-medoids algorithm. We run our algorithm over some synthesized and real world datasets. In order to show privilege of our method and to compare the results, we also introduce an algorithm which randomly selects pairwise distances and estimates unknown ones using Floyd-Warshall algorithm.

References

1. Amsterdam library of object images (aloi). `http://aloi.science.uva.nl/`, 2004.

2. Example data sets for elki. `http://elki.dbs.ifi.lmu.de/wiki/DataSets`, 2013.

3. A. Arbelaez and L. Quesada. Parallelising the k-medoids clustering problem using space-partitioning. In *Proceedings of the Sixth Annual Symposium on Combinatorial Search, SOCS 2013, Leavenworth, Washington, USA, July 11-13, 2013.*, 2013.

4. D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 1027–1035, 2007.

5. A. Asuncion and D. Newman. Uci machine learning repository datasets. `https://archive.ics.uci.edu/ml/datasets.html`, 2007.

6. S. Basu, A. Banerjee, and R. J. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA, April 22-24, 2004*, pages 333–344, 2004.

7. A. Biswas and D. W. Jacobs. Active image clustering with pairwise constraints from humans. *International Journal of Computer Vision*, 108(1-2):133–147, 2014.

8. M. Chen. Synthesized dataset for k-medoids. `http://www.mathworks.com/matlabcentral/fileexchange/28898-k-medoids/`.

9. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

10. B. Eriksson, G. Dasarathy, A. Singh, and R. D. Nowak. Active clustering: Robust and efficient hierarchical clustering using adaptively selected similarities. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 260–268, 2011.

11. N. Grira, M. Crucianu, and N. Boujemaa. Active semi-supervised fuzzy clustering. *Pattern Recognition*, 41(5):1834–1844, 2008.

12. L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.

13. A. Krishnamurthy, S. Balakrishnan, M. Xu, and A. Singh. Efficient active algorithms for hierarchical clustering. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

14. S. T. Mai, X. He, N. Hubig, C. Plant, and C. Böhm. Active density-based clustering. In *2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, December 7-10, 2013*, pages 508–517, 2013.

15. H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, pages 737–744, 2009.

16. S. Nayar, S. A. Nene, and H. Murase. Columbia object image library (coil 100). *Department of Comp. Science, Columbia University, Tech. Rep. CUCS-006-96*, 1996.

17. X. V. Nguyen, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, pages 1073–1080, 2009.

18. B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

19. O. Shamir and N. Tishby. Spectral clustering on a budget. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 661–669, 2011.
20. S. S. Singh and N. Chauhan. K-means v/s k-medoids: A comparative study. In *National Conference on Recent Trends in Engineering & Technology*, volume 13, 2011.
21. S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.
22. A. Ultsch. Fundamental clustering problems suite (fcps). Technical report, Technical report, University of Marburg, 2005.
23. K. Voevodski, M. Balcan, H. Röglin, S. Teng, and Y. Xia. Active clustering of biological sequences. *Journal of Machine Learning Research*, 13:203–225, 2012.
24. V. Vu, N. Labroche, and B. Bouchon-Meunier. Improving constrained clustering with active query selection. *Pattern Recognition*, 45(4):1749–1758, 2012.
25. K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 577–584, 2001.
26. X. Wang and I. Davidson. Active spectral clustering. In *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, pages 561–568, 2010.
27. F. L. Wauthier, N. Jojic, and M. I. Jordan. Active spectral clustering via iterative uncertainty reduction. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 1339–1347, 2012.
28. C. Xiong, D. M. Johnson, and J. J. Corso. Active clustering with model-based uncertainty reduction. *CoRR*, abs/1402.1783, 2014.
29. B. H. N. B. A. B. A. M. Yanping Chen, Eamonn Keogh and G. Batista. Ucr time series classification archive. `http://www.cs.ucr.edu/~eamonn/time_series_data/`, 2015.