DistClusTree: A Framework for Distributed Stream Clustering

Zhinoos Razavi Hesabi[1(✉)], Timos Sellis[2], and Kewen Liao[2]

[1] School of Computer Science and IT, RMIT University, Melbourne, Australia
zhinoos.razavi@rmit.edu.au
[2] Department of Computer Science and Software Engineering,
Swinburne University, Hawthorn, VIC, Australia
{tsellis,kliao}@swin.edu.au

Abstract. In this paper, we investigate the problem of clustering distributed multidimensional data streams. We devise a distributed clustering framework DistClusTree that extends the centralized ClusTree approach. The main difficulty in distributed clustering is balancing communication cost and clustering quality. We tackle this in DistClusTree through combining spatial index summaries and online tracking for efficient local and global incremental clustering. We demonstrate through extensive experiments the efficacy of the framework in terms of communication cost and approximate clustering quality.

1 Introduction

Classical clustering algorithms are mainly designed for static datasets while anytime clustering of massive data streams is highly demanded in modern distributed data acquisition systems. Continuously changing data distributions raises a challenge: new data should be able to efficiently locate its cluster, and the clustering structure should be updated incrementally in a continuous online manner, that is, the structure is reorganized once the distribution with new data significantly invalidates the older organization. Further, in the setting of distributed clustering, updating master clustering via communications and its entire reorganization become far more challenging.

Hence, the problem we target to solve is on optimizing the *lifecycle* of distributed stream clustering – how can we build, organize, track, and update high quality summaries/approximations of clusters in an effective and efficient manner?

ClusTree [8] is one state-of-the-art centralized multidimensional stream clustering approach that leverages spatial index and microclusters together as hierarchical summaries of clusters. It achieves effective and efficient anytime clustering. However, for a distributed framework, communication cost is often the main bottleneck and the quality of global clustering is paramount. Hence, the focus of the study is to track and balance these criteria in our proposed DistClusTree framework while in terms of the clustering quality performing no worse than

J. Wang et al. (Eds.): ADC 2018, LNCS 10837, pp. 288–299, 2018.
https://doi.org/10.1007/978-3-319-92013-9_23

ClusTree. Specifically, the new framework DistClusTree defines the effectiveness of global clustering at a central site as a cost function of communication and degree of cluster approximation. This means the framework is able to monitor changes in local sites and strategically send updates to the central site. In this way, communication is only triggered between local sites and the central site whenever the quality of clustering degrades beyond a threshold.

To the best of our knowledge, this is the *first attempt* of a distributed multidimensional stream clustering framework that combines index data structure for summarizing and maintaining local microclusters, and online tracking for the monitoring and maintenance of global clustering.

As a summary, our main contributions of the paper are as follows:

- We extend ClusTree into a distributed framework DistClusTree.
- We propose adaptable solutions to select local microcluster summaries to be sent to the central site. The central site then refines the quality of clustering according to real-time network traffic and degree of privacy.
- We model the monitoring and maintenance of distributed clustering as online tracking and extend a 1-to-1 multidimensional online tracking scheme into m-to-1 (m local sites with one central site).
- We demonstrate through extensive experiments the performance of our framework in balancing communication cost and clustering quality.

2 Related Work

Due to space limit, here we only briefly review the typical distributed clustering algorithms related to our work. In [2], two types of continuous distributed clustering algorithms were proposed: local and global. The algorithm is formulated based on k-center clustering algorithm. The main objective in k-center is to minimize the maximum radius/diameter of the clusters. In their local solutions, each local site builds and keeps its local model, and only updates are sent to the central site. In their global algorithm, a global model is created iteratively in a central site by message passing between local sites and the global site. Then the global model is sent to all local sites and in this way all sites have the same view of the clustering model. Local sites insert their data points into the global clustering and continuously send their updates to the central site. The central site decides whether it needs to recompute the global model and sends the new clusters to the local sites. As in most of the distributed clustering settings, distributed clustering results are compared with their centralized counterpart.

A distributed extension of the Expectation Maximization (EM) algorithm called CluDistream is proposed in [12]. The authors introduced a framework for clustering distributed data streams in the presence of noisy and incomplete data. The underlying distribution of data has been learnt by maximizing the likelihood of the data clusters. Local sites monitor the current model till they could not describe a new chunk of data. Then a clustering is performed to account for the changing data distribution. In this way, they reduced communication cost by

just sending updates from local sites to the central site. In [5], centralized density based clusterings algorithm (DBSCAN) [3] was extended to the distributed model. Each local site performs a clustering on its own data and sends its representatives to the central site. Local representatives are grouped into each other to represent the final clustering in the global site. The work shows that the results of clustering in centralized and distributed model are significantly close to each other. There are some other works that have addressed clustering distributed data streams such as [6,7] and there is a recent survey [9] on distributed clustering of ubiquitous data streams.

3 Preliminaries

In this section we explain in more detail: (i) the ClusTree approach adopted for local clustering in DistClusTree; (ii) the underlying R-tree data structure in which local cluster summaries are stored and maintained.

ClusTree- ClusTree is introduced in [8] as a stream clustering algorithm with the ability of anytime clustering. ClusTree consists of two phases: online and offline. In the online phase, it collects summary of data in the form of a vector of number of data points (N), linear sum (LS), and Squared Sum (SS) of data points – the *Cluster Feature Vector (CFV)* considered as a microcluster. Storing these summaries instead of raw data helps to save space. Also, these summaries are sufficient to compute statistical parameters such as mean, variance, centroid, radius, diameter for further off-line clustering. Updating *CFVs* is easy and computationally fast because of their additivity and subtractive properties. ClusTree uses an index tree data structure, essentially an R-tree, to maintain collected summary statistics (i.e. the microclusters). The idea is to build a hierarchy of microclusters at different levels of granularity. This data structure accelerates the process of searching the right microclusters for inserting new data points as they arrive. When a new data point arrives, it descends the index tree till arriving the leaf node which contains a microcluster that has the minimum distance to the inserted data point. From time to time microclusters are transfered to a disk to be kept for further offline processing. For example a data partition clustering like k-means or DBSCAN is performed on microclusters to form final clusters in an offline phase. The outputs of the offline phase are macroclusters whose size is relatively small compared to the entire data stream.

R-tree- ClusTree leverages an R-tree [4] data structure to maintain the collected summary statistics of data (i.e., *CFVs*) online. R-tree is a tree data structure used as a spatial index. R-tree clusters multidimensional data based on their proximity. It represents nearby data objects with their minimum bounding boxes in different levels of the tree. The main goal of this data structure is to group adjacent data objects and represent them with their Minimum Bounding Rectangle (MBR) in the next higher level of the tree. Since all data objects fall within this MBR, a query can be answered if it intersects any bounding rectangle. Aggregation/Summarization of objects occurs at the higher levels of the tree while the root represents aggregation of all data objects. From a clustering point of view, descending the tree reduces within-cluster sum of squares

error. The within-cluster sum of squares measures the variability of the data points within each cluster. Usually a cluster with a small sum of squares is more compact than a cluster that has a large sum of squares. This translates to an increased granularity of clustering at the leaf level of the tree where data objects are indexed. This can also be seen as an increasingly coarser approximation of data distribution as we move up in the tree.

4 DistClusTree

In the framework, m local sites are distributed in a network and each of them receives and incrementally clusters a continuous stream of data, possibly with an infinite length. A master/central site instead clusters and maintains the union of the local site data to produce the final global clustering result. We are particularly interested in devising mechanisms that allow local sites to communicate with the central site efficiently. Leveraging summarized but still informative data to be sent to the central site for global clustering instead of sending the actual massive data points is a key approach to reduce communication cost and preserve privacy. Moreover, a communication can be triggered between local sites and the central site every time a new data object arrives at each local site so as to update clusters maintained at the central site. This poses an expensive communication rounds of $O(n)$, where n is the total number of arrived data points. Therefore, a plausible way to balance the communicate cost and clustering quality is to trigger communication periodically and only send the selected updated summaries/representatives to the central site. Our studies shows the choice of proper local representatives has a significant impact on communication cost and central clustering result. The representatives form a summary of local models at a given time snapshot. The summary is sent to the central site and kept being locally updated as new data points (stream snapshots) arrive.

In essence, DistClusTree consists of four stages: (1) Continuous local clustering; (2) Extracting local representatives; (3) Distributed microcluster tracking; and (4) Maintaining global clusters. These are described in detail in the following.

1- Continuous Local Clustering. Every local site clusters its data incrementally with the ClusTree approach. Summaries of data are collected as CFVs and maintained dynamically in an R tree. In this way micro-clusters are maintained in various levels of the tree and in different resolutions (i.e. coarser microclusters are located in higher levels of the tree). Therefore, the root node in the ClusTree contains the broadest view of all microclusters at the current snapshot, while the leaf nodes include all of the fine-grained micro-clusters. Such hierarchically summarized organization is shown in Fig. 1.

2- Extracting Local Representatives. To extract local representatives, we propose two simple but effective and adaptable approaches: Naive-DistClust and DistClust.

Naive-DistClust- Local representatives from different levels of the tree are extracted regularly (i.e. at every ΔT time period) to be communicated to the

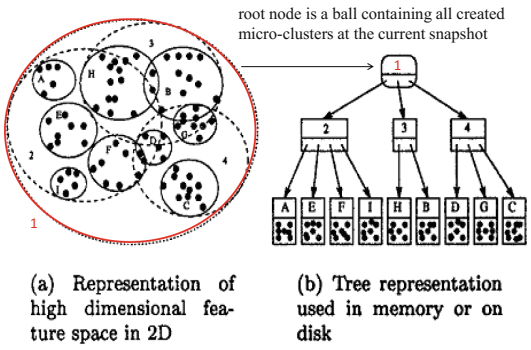


Fig. 1. The local ClusTree summary in DistClusTree (source [10]).

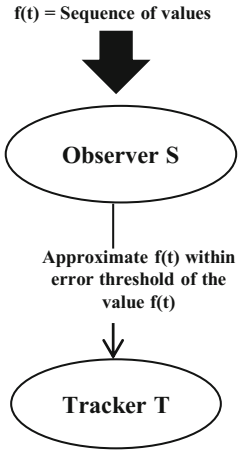


Fig. 2. One-to-one online tracking.

central/global site. This approach is adjustable based on the network traffic (i.e. the frequency of data arrivals), degree of privacy, and required quality of central clustering. Depending on traffic and required quality of clustering, local sites can send created micro-clusters at different levels of the tree to the central site. For maximum quality, local sites should send all created micro-clusters at their leaf level. While in a heavily loaded network (e.g., peak hours), more compressed trees (i.e. at most the root level) with some sacrifice of clustering quality can be sent to the central site. This translates to reducing the overall communication cost by sending more coarse local micro-clusters from higher levels of the tree to the central site.

DistClust- A further way to reduce communication cost is for every tree node only sending statistical summaries of its contained microclusters to the central site. For example, in an R-tree only with 3-fan outs (i.e. the number of entries in each node where each entry represents one summarized micro-cluster), at level 1 (considering level 0 as the root level), we have 3 subtrees each containing 3 micro-clusters. This means we have 9 micro-clusters in total at level 1. Instead of sending all these 9 entries to the central site, we could choose to send only the median of the entries from each node, thereby reducing communication cost to one-third. Next, we discuss in detail how local representatives (i.e. the selected microclusters) are tracked and sent with an on-line tracking algorithm.

3- Distributed Microcluster Tracking. We first give a brief introduction on online tracking and then illustrate how we formulate the global clustering in DistClusTree as an online tracking problem. In the conventional online tracking, a pair of observer and tracker communicates with each other. Observer observes values of a function f over time and keeps inform the tracker these values time to time as shown in Fig. 2. However, determination of a strategy that minimizes communication cost is the main issue in online tracking problems. A naive solu-

tion is that observer sends every observed value to the tracker. This leads to a heavy communication. To minimize the communication cost, an error threshold is generally introduced. This means observers only communicate with the tracker whenever a value of $f(t_{now})$ exceeds a predefined error threshold Δ from the last communicated value $f(t_{last})$. We extend the multidimensional one-to-one online tracking framework presented in [11] that only works when there is an observer and a tracker. It is not designed for the distributed m-to-one communication where there are multiple observers and a central tracker. The one-to-one online tracking algorithm divides the entire tracking period into rounds and denotes A_{opt} as the offline optimal algorithm. A round is started by initializing a set $S = S_0$ which contains all possible points that might be sent by A_{opt} in its last communication. In a while loop, a median of S is calculated and sent to the tracker. If $\|f(t_{now}) - f(t_{last})\| \geq \beta\Delta$, where $\beta = 1/(1+\epsilon)$ and Δ represents the threshold error, then S is updated as $S \leftarrow S \cap Ball(f(t_{now})), \Delta)$, where $f(t_{now})$ is the center of Ball and Δ is its radius in d-dimensional space. A round is terminated when S becomes empty. The online tracking algorithm is represented in Algorithm 1.

1. Let $P = Ball(f(t_{now}), \beta\Delta)$;
2. **while** $(\omega_{max}(p)) \geq \epsilon\Delta$ **do**
   Let $g(t_{now})$ be the centroid of P;
   send $g(t_{now})$ to tracker;
   wait until $\|f(t_{now}) - g(t_{last})\| \geq \beta\Delta$
   $S \leftarrow S \cap Ball(f(t_{now}), \Delta)$
   end

Algorithm 1. *One round of d-dimensional tracking*

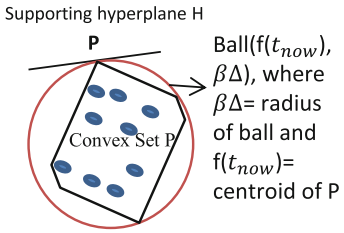


Fig. 3. Convex set P is covered by $Ball(f(t_{now}, \beta\Delta)$.

We model our clustering algorithm based on the above online tracking algorithm. First we show how we can keep track of micro-clusters assuming there is one local site and one central site, and then we extend our algorithm from one local site to multiple local sites as shown in Fig. 4 (i.e., distributed) that they communicate with a central site in a synchronous mode. We explain with the Definition 2 and Lemma 4 from [11] below to illustrate our tracking model.

Definition 2 *(Directional Width).* For a set P of points in R^d, and a unit direction μ, the directional widths of P in direction μ is $\omega_\mu(P) = max_{p \in P} \langle \mu, p \rangle - min_{p \in P} \langle \mu, p \rangle$, where $\langle \mu, p \rangle$ is the standard inner product.

For simplicity, suppose a given set of points form a convex set P, and the centroid of P is the intersection of hyperplanes that divide P into two equal parts. This convex set has minimum and maximum directional width as $\omega_{min}(p)$, $\omega_{max}(p)$, respectively.

Lemma 4. For any convex set P, if H is any supporting hyperplane of P at $p \in \partial P$, that is, H contains p and P is contained in one of the two halfspaces bounded by H. Then there is a ball B with radius $\beta\Delta$ such that H is tangent to B at p and B contains P as shown in Fig. 3.

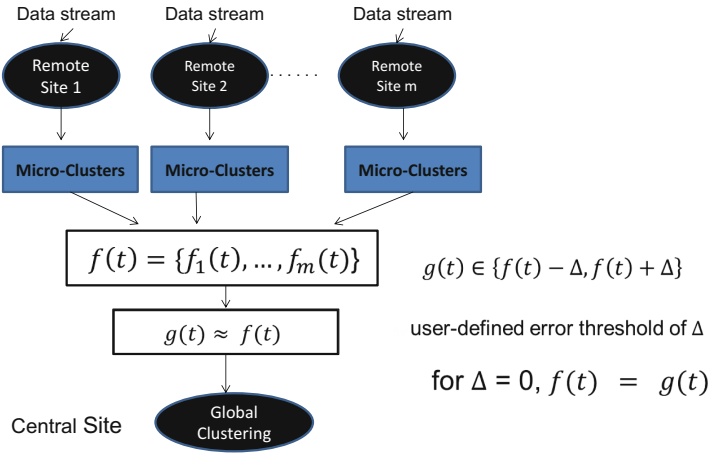


Fig. 4. The global DistClusTree framework.

Local site acts as an observer that sends an approximation of created micro-clusters to a central site at different time snapshots. Assume that error threshold Δ is determined based on the maximum distance between centroids of two clusters at two consecutive snapshots of t_i and t_{i+1}, where $C_{t_i} = f(t_{last})$ and $C_{t_{i+1}} = f(t_{now})$ are the centroids of previous and current root node. According to [11], a convex set P in our local ClusTree is a set of microclusters taken at snapshot t_{now}. Based on Lemma 4, a ball containing P is the root node at snapshot t_{now}. Following this, we initialize P with the root node, and then centroid of the current root (i.e. as a representative of all microclusters) is computed as $g(t_{now})$ and sent to the tracker. In the next snapshot, if the absolute euclidean distance between the centroid of the previous root node and the current root node is within the predefined error threshold, then there is no communication. Otherwise, the intersection of two roots (two balls) is computed. If the maximal directional width of this intersection is greater than $\beta\Delta$, then a communication between local site and the central site is triggered and the centroid of the intersection is sent to the central site. Otherwise, this round is finished and the next new round is triggered. Different scenarios that may happen between the last communicated root node and the new root node at two different snapshots of t_1 and t_2 are presented in Fig. 5 (a–c).

4- Maintaining Global Clusters. For simplicity, we only enhance the tracking framework from 1-to-1 to m-to-1 sites in a synchronous manner. Each local site

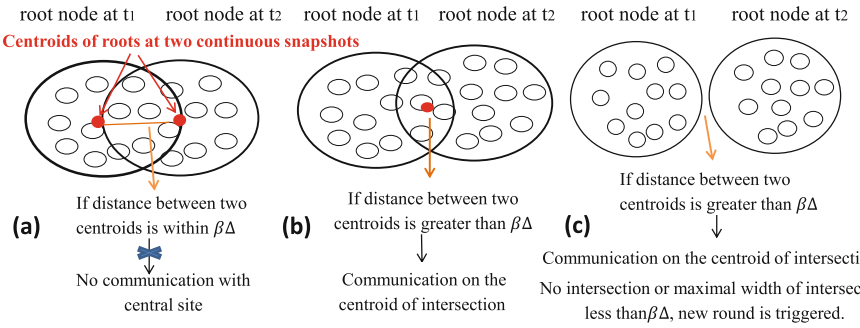


Fig. 5. Different scenarios to trigger a communication between a local site and a central site to update the global clustering.

keeps track of its local representatives in periodical time snapshots and if any threshold breaking occurs at a local site, then this site simply sends its updates to the central site along with all other updates from other local sites. As depicted in Fig. 4, local sites communicate with a central site if $f_1(t_1) - f_1(t_2)$ are within some error threshold. After sending updates to the central site, central site does a global clustering using k-means over the union of all received microclusters from local sites. Local sites incrementally send their updates to the central site to keep global clusters updated. By receiving regular updates, the central site incrementally keep global clusters updated.

5 Experiments

We implement DistClusTree under Massive Online Analysis (MOA) [1] and evaluated the distributed algorithms based on a synthetic dataset. The dataset is generated using Gaussian distribution with varying number of attributes and classes. Data points are randomly and equally divided among sites and for the central clustering, we use the union of the local points. Our experiments focus on clustering quality and the communication cost of distributed clustering considering their dependency on different parameters such as number of sites, the accuracy ϵ, granularity of local representatives and runtime of distributed clustering in comparison with centralized clustering. To assess our framework, we ran ClusTree on each local site and then collected all representatives of local sites w.r.t the demanded granularity of local clusterings (i.e., different levels of local trees) and then performed a global clustering on these representatives. We executed all the experiments on the same machine and reported all the results as average of 10 runs of our algorithms. We compare clustering quality of our distributed clustering (i.e., DistClusTree) against its centralized counterpart, i.e. ClusTree. As mentioned in [5], different studies have evaluated their algorithms based on characteristics of their distributed clustering algorithm in variety of ways and the majority of studies compares their proposed distributed clustering algorithm against their centralized counterpart [9]. Therefore, we compared the

result of our distributed clustering algorithms to a central clustering of the n data points when all n data points are clustered using ClusTree in local sites and applying k-means on top of the microclusters created at leaf level of the tree.

Clustering quality - We measure quality of clustering by defining Mean Squared Error (MSE), and also using within-cluster sum of squares error. As the baseline, firstly we sent all microclusters created at the leaf level to the central site and applied k-means to calculate cluster centroids. Secondly we sent microclusters of each level of the local trees and find cluster centroids for each level using k-means. To calculate MSE, we take the average of euclidean distances between cluster centroids obtained from the last level of tree in ClusTree (i.e., centralized model) and every level of the local trees (DistClust/Naive-DistClust). As it can be seen in Fig. 6(A), MSE is reduced by descending tree since microclusters with smaller within cluster sum squared error are located at the lower level of the tree which impacts on quality of clustering in the central site. We compared MSE of $k = 5$ centroids at different levels of the tree for both DistClust and Naive-DistClust. In the latter, we send all created microclusters from different levels of trees while in DistClust we only send a mean of microclusters of each node of tree. That is the reason MSE between ClusTree and NaiveClust is less than MSE between ClusTree and DistClust. The MSE difference between both distributed algorithms is getting higher at the lower levels of tree since granularity increases at the bottom levels and sending less microclusters impacts on calculating right centroids and consequently on clustering quality. We ran the experiments with 3 number of fans out at each node of R-tree as referring to [8], 3 fans out is the best number of entries (#microclusters at each node) in terms of space and distance computation.

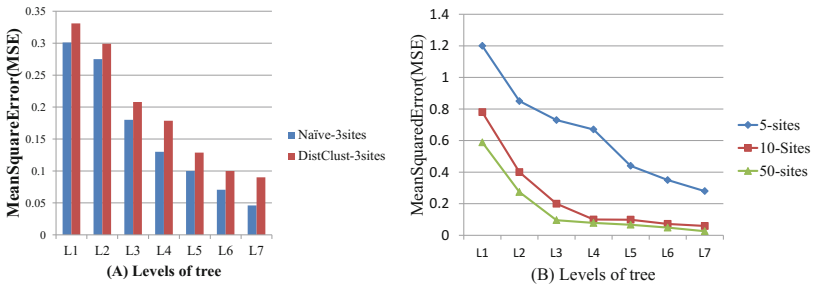


Fig. 6. (A) Comparison of clustering quality of Naive-DistClust and DisClust with ClusTree on different levels of tree, when number of sites = 6 and number of entries = 3; (B) MSE comparison of varying number of sites: 5, 10, 50.

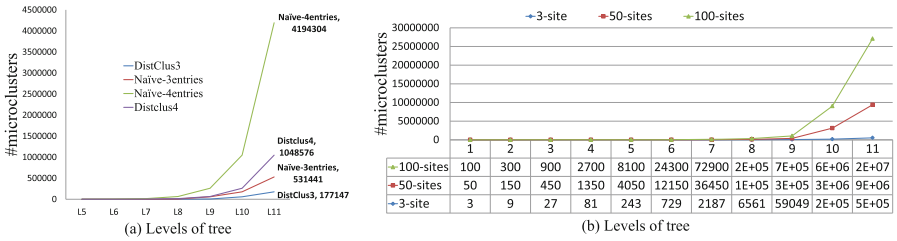
On the other hand, as you can see in Fig. 6, MSE is reduced by descending the tree. The reason is that purity of microclusters at lower levels of tree is increased which causes to reduce MSE between cluster centroids obtained from upper and lower levels of the tree. We test our framework for different numbers of sites 5,

10, and 50. Although in all plots in Fig. 6(B) MSE is reduced by descending the tree, reducing number of sites reduces quality of clustering because we send less micro-clusters which impacts on final quality of clustering at the central site.

Communication cost - We calculated communication cost in terms of number of transfered microclusters from each level of the tree as *communication ratio* $= (compressedtree/uncompressedtree) \times numberofsites$.

In our formula, communication ratio is calculated as the ratio of compressed tree to uncompressed tree. Uncompressed tree is a full multi-way tree with maximum number of levels. Maximum number of level is predefined by the user or local memory limit. Compressed tree is a full multi-way tree with less number of levels compared to the uncompressed one. The lowest communication ratio is the ratio of the minimum number of levels (maximum compression, i.e., root level of tree) to the maximum number of levels. Communication ratio also depends on the number of entries in each node. For instance, in a 3-way full tree, level 0 which is root level has 1 node, level 1 has 3 nodes, level 2 has 9 nodes and in general for n-multi-way tree, the number of node in level n is calculated as m^n, where m represents number of ways in the tree.

We compare communication cost of different levels of the tree for different number of entries at each node of the R-tree. In Fig. 7, we compare the communication cost for two different number of entries, 3 and 4 in our two proposed distributed clustering algorithms. By sending median of microclusters at each node in DistClust, we reduce communication cost to $1/k$, where k is the number of entries. We reduce communication cost to one third with Distclust for the choice of three entries in all levels of tree. This reduction is obvious in the lower levels of the tree where more granular microclusters are required.



	1	2	3	4	5	6	7	8	9	10	11
100-sites	100	300	900	2700	8100	24300	72900	2E+05	7E+05	6E+06	2E+07
50-sites	50	150	450	1350	4050	12150	36450	1E+05	3E+05	3E+06	9E+06
3-site	3	9	27	81	243	729	2187	6561	59049	2E+05	5E+05

Fig. 7. (a) Effect of number of entries on communication cost; (b) Communication cost of DistClust for different number of sites when number of entries = 3.

Figure 7(b) represents the communication cost in terms of number of microclusters for 3,50 and 100 sites. The number of entries in all 3 experiments has been set to 3 and the height of tree is 11. As it can be clearly seen that communication cost depends on the number of sites and different levels of the tree. To have more granular clusters we need to send microclusters at the lower level of tree causing to increase communication cost exponentially. While by sending

representatives from upper levels of trees we reduce communication cost significantly and still have good quality clustering as demonstrated in the above experiments.

Effect of error threshold Δ - We evaluated the effect of varying error threshold on communication cost. Error threshold is the difference between centroid of the new microcluster at snapshot t_i and the previous transmitted at snapshot t_{i-1}. As error threshold is increased the communication cost is decreased since we send less number of updates to the central site by increasing euclidean difference between centroids of previous and current snapshots. The communication cost at the lower levels of the tree is higher than the upper levels of tree as shown in Fig. 8 for L1 as root level and level 6. However, increasing error threshold reduces quality of clustering.

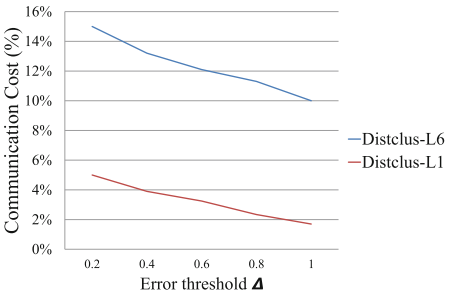


Fig. 8. Effect of different Δ values on communication cost, level 1 and 6, 10 sites.

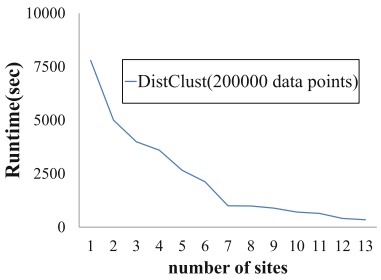


Fig. 9. Runtime for central and distributed clustering with varying number of sites.

Runtime - In Fig. 9, the runtime of DistClusTree is shown. As number of sites are increasing, the distributed approach performs much better than a single clustering algorithm applied to the complete data set of 200k data points.

6 Conclusion

We extended ClusTree into DistClusTree, a comprehensive distributed framework for stream clustering. The framework leverages both spatial index summaries and online tracking for balancing communication cost and clustering quality. We demonstrated in experiments that DistClusTree efficiently produces clusters as good as its centralized version. DistClusTree is able to reduce communication cost significantly and it is easily configurable in practice according to the requested clustering quality. For future work, we plan to carry out more insightful theoretical analysis and justification of DistClusTree.

References

1. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: massive online analysis. J. Mach. Learn. Res. **11**, 1601–1604 (2010)
2. Cormode, G., Muthukrishnan, S., Zhuang, W.: Conquering the divide: continuous clustering of distributed data streams. In: 2007 IEEE 23rd International Conference on Data Engineering, pp. 1036–1045, April 2007
3. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD 1996, pp. 226–231. AAAI Press (1996)
4. Guttman, A.: R-trees: a dynamic index structure for spatial searching, vol. 14. ACM (1984)
5. Januzaj, E., Kriegel, H.-P., Pfeifle, M.: Towards effective and efficient distributed clustering. In: Workshop on Clustering Large Data Sets ICDM, pp. 49–58 (2003)
6. Kargupta, H., Huang, W., Sivakumar, K., Johnson, E.: Distributed clustering using collective principal component analysis. Knowl. Inf. Syst. **3**, 2001 (1999)
7. Klusch, M., Lodi, S., Moro, G.: Distributed clustering based on sampling local density estimates. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI 2003, pp. 485–490. Morgan Kaufmann Publishers Inc., San Francisco (2003)
8. Kranen, P., Assent, I., Baldauf, C., Seidl, T.: The clustree: indexing micro-clusters for anytime stream mining. Knowl. Inf. Syst. **29**(2), 249–272 (2011)
9. Rodrigues, P.P., Gama, J.: Distributed clustering of ubiquitous data streams. Wiley Interdisc. Rev. Data Mining Knowl. Disc. **4**(01), 38–54 (2014)
10. White, D.A., Jain, R.: Similarity indexing with the SS-tree. In: Proceedings of the Twelfth International Conference on Data Engineering, pp. 516–523, February 1996
11. Yi, K., Zhang, Q.: Multidimensional online tracking. ACM Trans. Algorithms (TALG) **8**(2), 12 (2012)
12. Zhou, A., Cao, F., Yan, Y., Sha, C., He, X.: Distributed data stream clustering: a fast EM-based approach. In: 2007 IEEE 23rd International Conference on Data Engineering, pp. 736–745, April 2007