SECLEDS: Sequence Clustering in Evolving Data Streams via Multiple Medoids and Medoid Voting (Encore abstract)

Azqa Nadeem and Sicco Verwer

Delft University of Technology, The Netherlands
{azqa.nadeem,s.e.verwer}@tudelft.nl

Abstract

Stream clustering is the problem of clustering an unbounded stream of data items in a single pass. Sequence clustering in a streaming setting is challenging because i) the data stream might contain concept drift, and ii) the sequences themselves might be out-of-sync, making them expensive to cluster. K-medoids or Partitioning Around Medoids (PAM) is commonly used for sequence clustering because it supports arbitrary distance measures, *e.g.,* the alignment-based dynamic time warping (DTW) distance that is especially designed for sequences [5,6]. The clusters generated by k-medoids are also interpretable since the k-centers are represented by actual data items. However, even the most optimized offline variants of k-medoids with a runtime of $\mathcal{O}(nlogn)$ [3,4] cannot be adopted for streaming settings because they are prohibitively expensive and cannot handle concept drift.

In the accepted paper [2], we propose SECLEDS – a lightweight streaming version of the k-medoids algorithm. In order to support high bandwidth data streams, SECLEDS does not store any part of the stream: it receives an item, assigns it to one of the k-clusters, and then discards it. This way, SECLEDS has a guaranteed constant memory footprint.

SECLEDS also has two unique properties: i) it produces stable and high-quality clusters using *multiple medoids per cluster*, and ii) it minimizes the required number of distance computations by introducing an intuitive *medoid voting scheme* to estimate a cluster's center of mass. SECLEDS maintains votes for each medoid, estimating how representative it is based on the data seen so far. A user-supplied decay factor allows SECLEDS to slowly forget the votes regarding older data. The least representative medoids are then replaced with new data items at each time step. This way, the clusters evolve with an evolving data stream, without having to perform additional distance computations. This allows SECLEDS to support computationally expensive distance measures, such as DTW. The medoid voting scheme also enables SECLEDS to handle concept drift using k-evolving clusters, instead of having to create new clusters for new concepts like typical adaptive algorithms. SECLEDS is implemented in Python, and is available open-source[1].

[1] SECLEDS: `https://github.com/tudelft-cda-lab/SECLEDS`

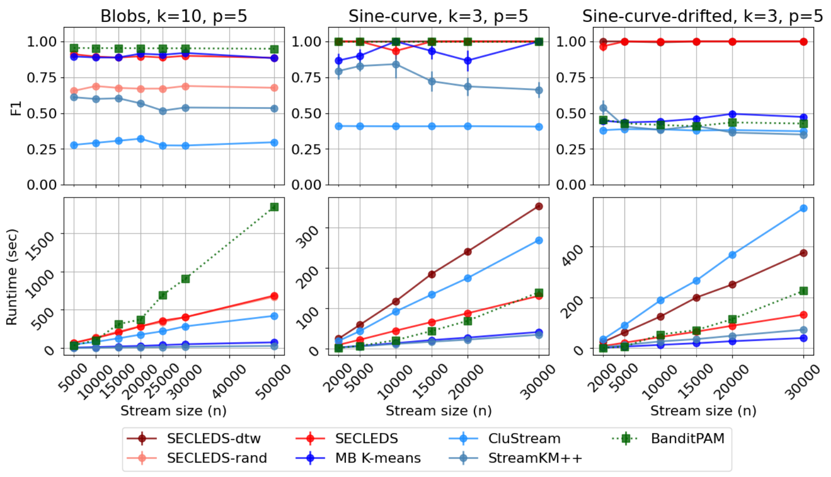

Fig. 1. The performance of SECLEDS on synthetic point (blobs) and univariate sequence (sine-curve) clustering, given k clusters and p medoids per cluster. Sine-curve-drifted is a drifted variant of the since-curve dataset (containing a univariate stream).

Testing on real and synthetic datasets, we empirically demonstrate that SECLEDS produces stable and high-quality clusters regardless of stream size, data dimensionality, concept drift, and number of clusters. We compare against three popular stream and batch clustering algorithms, *i.e.,* CluStream, StreamKM++, and Minibatch k-means. We treat BanditPAM as a benchmark for the best achievable clustering on an offline dataset. The results (given in Figure 1) show that SECLEDS achieves comparable F1 score[2] to BanditPAM, while reducing the required number of distance computations by 83.7%. The runtime of SECLEDS grows approximately linearly with stream size, while the F1 score remains competitive with the best performing baselines. For a stream with concept drift (sine-curve-drifted), SECLEDS outperforms all baselines by 138.7%, while being faster than BanditPAM and CluStream.

As a use case, we employ SECLEDS for temporal pattern-preserving network traffic summarization, *i.e.,* preserving temporal patterns while network traffic sampling so that we can use them for downstream tasks, *e.g.,* behavior analytics. To this aim, we cluster sequences of network traffic using SECLEDS and periodically store the cluster medoids. On a real-world data stream, we show that SECLEDS-dtw achieves an F1 score of 0.88 for $k = 5$ while BanditPAM only gets 0.38. SECLEDS-dtw clusters the stream collected in ∼5.5h in less than 27 minutes, implying that it can support network bandwidths of up to 1.08 Gbps, which is significantly higher than the needs of a typical enterprise network.

For future work, we are investigating how to make the medoids aware of each other to improve the cluster coverage in a computationally efficient way.

[2] F1 score is computed by comparing the pairwise co-occurrence of true labels [1].

References

1. Manning, C., Raghavan, P., Schütze, H.: Introduction to information retrieval. Natural Language Engineering **16**(1), 100–103 (2010)
2. Nadeem, A., Verwer, S.: Secleds: Sequence clustering in evolving data streams via multiple medoids and medoid voting. Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (2022)
3. Schubert, E., Rousseeuw, P.J.: Faster k-medoids clustering: improving the pam, clara, and clarans algorithms. In: SISAP. pp. 171–187. Springer (2019)
4. Tiwari, M., Zhang, M.J., Mayclin, J., Thrun, S., Piech, C., Shomorony, I.: Banditpam: Almost linear time k-medoids clustering via multi-armed bandits. NeurIPS **33**, 10211–10222 (2020)
5. Ushakov, A.V., Vasilyev, I.: Near-optimal large-scale k-medoids clustering. Information Sciences **545**, 344–362 (2021)
6. Wang, T., Li, Q., Bucci, D.J., Liang, Y., Chen, B., Varshney, P.K.: K-medoids clustering of data sequences with composite distributions. IEEE Transactions on Signal Processing **67**(8), 2093–2106 (2019)