UMD

University of Minnesota Duluth

Driven to Discover

Evaluation of the Orion Outlier Detection Algorithm for Data Streams

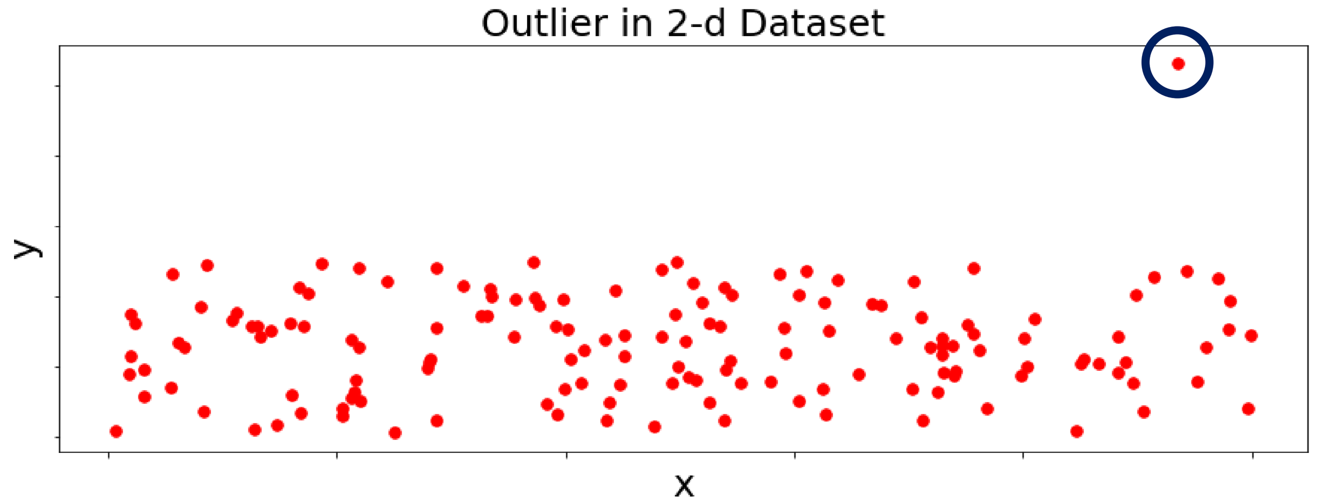
Nam Phung; Eleazar Leal, PhD
University of Minnesota Duluth

Summary: In this research study, we implemented in Java a density-based outlier detection algorithm called Orion, evaluated its performance and compared the results with several other outlier detection algorithms for data streams.

Introduction

An outlier is defined as an observation that has different values compared to other observations in a dataset.

Outlier detection algorithm for data streams has a vital role in Big Data applications. In these applications, data points arrive over time in a window, which makes the problem of detecting outliers becomes more complex due to the special characteristics of data streams.



Orion is a density-based outlier detection algorithm for data streams [1]. Orion detects outliers through three phases:

1. Find a p-dimension Z_a that minimizes the stream density of a data point D_T using an evolutionary algorithm.
2. Compute the stream density and k-integral metrics of D_T projected on Z_a
   i. The stream density of a data point D_T along a p-dimension Z_a is the percentage of neighbors of D_T within the scaled neighbor distance r_{Za}.
   ii. The k-integral of D_T is the integral that includes k percent of the data points along Z_a.
3. Perform co-clustering based on the two metrics to detect the outliers.

Methods

We implemented Orion in Java with JDK 11. We used the Watchmaker framework for evolutionary computation in phase two and the Expectation-Maximization (EM) clustering algorithm provided by WEKA framework in phase three [2].

We used two datasets from the UCI machine learning repository and OpenML Repository for performance analysis:

1. Mulcross dataset (262,144 records, 4 dimensions)
2. Tao dataset (575,468 records, 3 dimensions)

We compared Orion with three existing distance-based outlier detection algorithms: Abstract-C, Micro-Cluster Based Algorithm (MCOD), and Approximate-Storm [3][4].

We measured the outlier detection performance in terms of precision, recall, and Jaccard coefficient (JC). In addition, we also studied the execution time of Orion to gain more insight into its scalability.

Comparison of Orion and Other Algorithms

Table 1. Performance Results

Algorithm	Dataset	Precision	Recall	Jaccard Coefficient	F-1 Score
Orion	mulcross	**0.411145**	**0.670213**	**0.920538**	**0.509645**
AbstractC		0.001907	0.084746	0.898132	0.003731
Approximate-Storm		0.058747	0.099883	0.852936	0.073982
Micro-Cluster		0.004005	0.088832	0.896294	0.007665
Orion	tao	**0.860670**	0.016143	0.612727	0.031692
AbstractC		0.517928	**0.653266**	0.973538	**0.577778**
Approximate-Storm		0.446454	0.177866	0.970825	0.254386
Micro-Cluster		0.303663	0.229582	**0.980876**	0.261477

Figure 1. Precision of Outlier Detection Algorithms

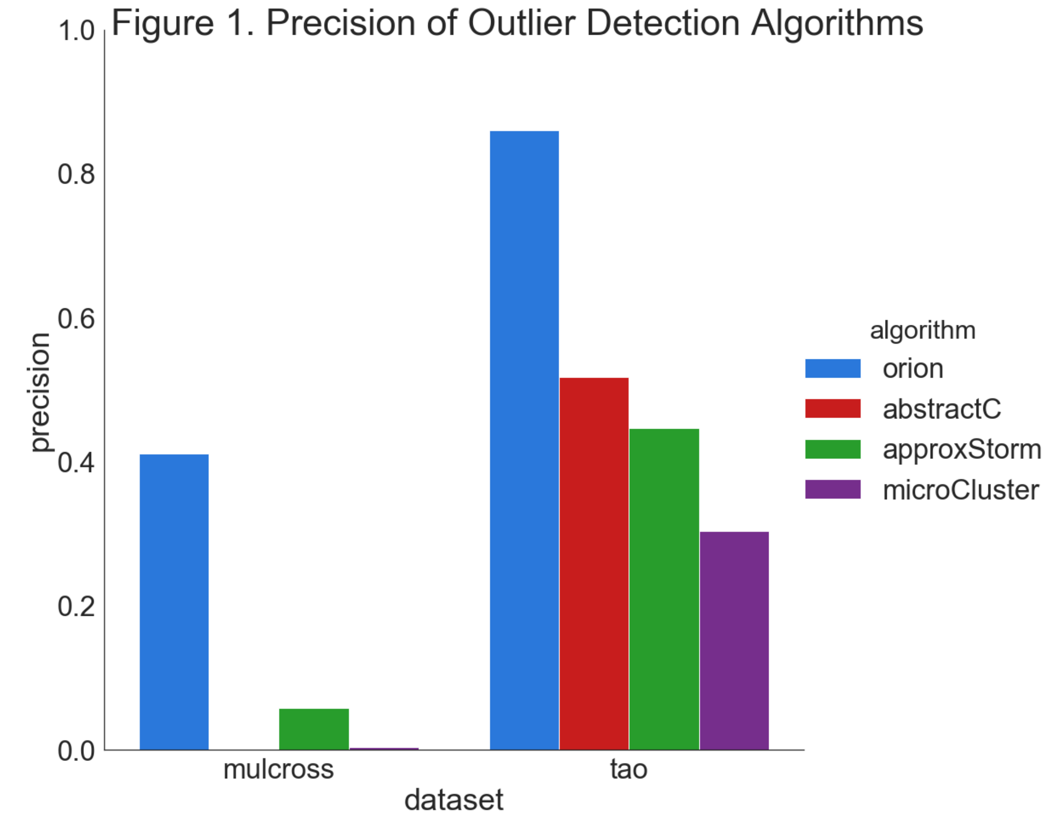


Figure 2. Recall of Outlier Detection Algorithms

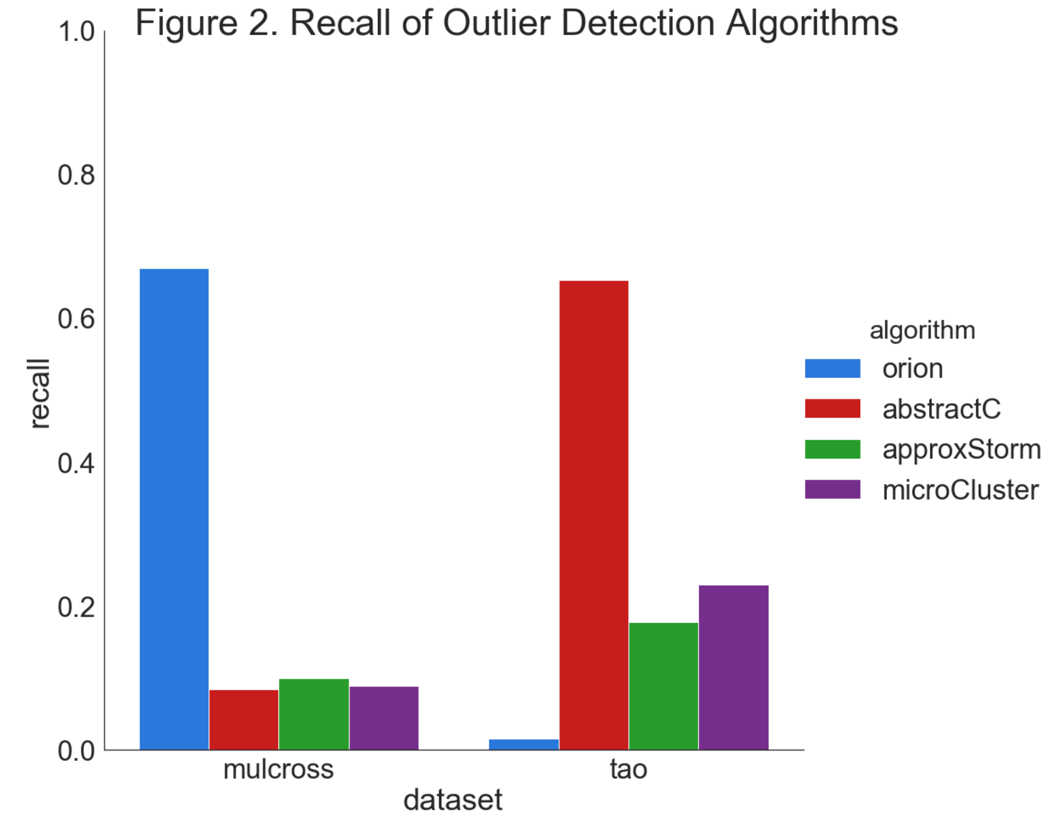


Figure 3. Jaccard Coefficient of Outlier Detection Algorithms

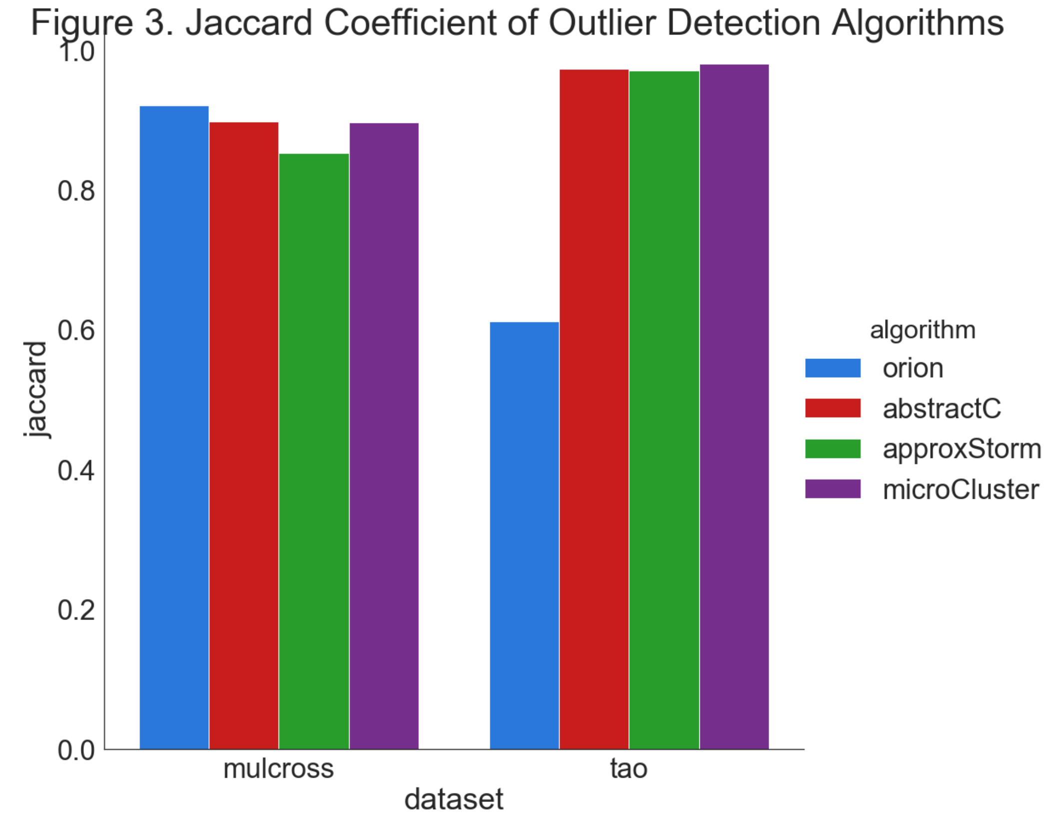
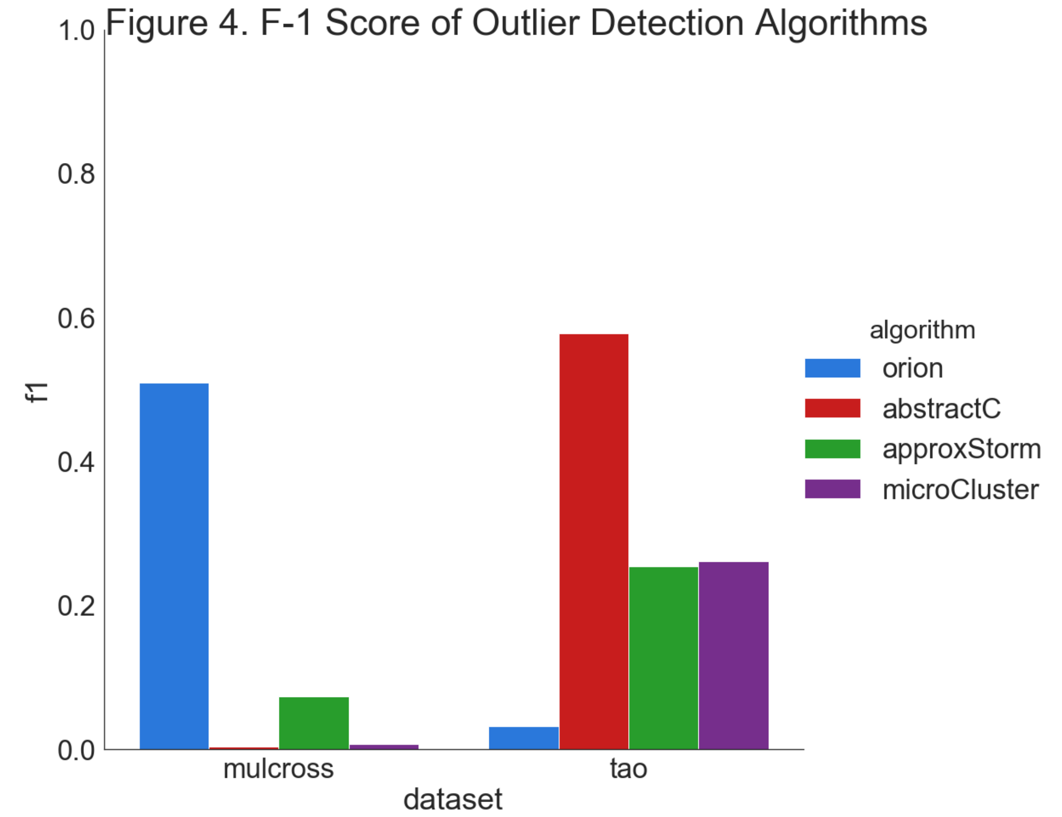
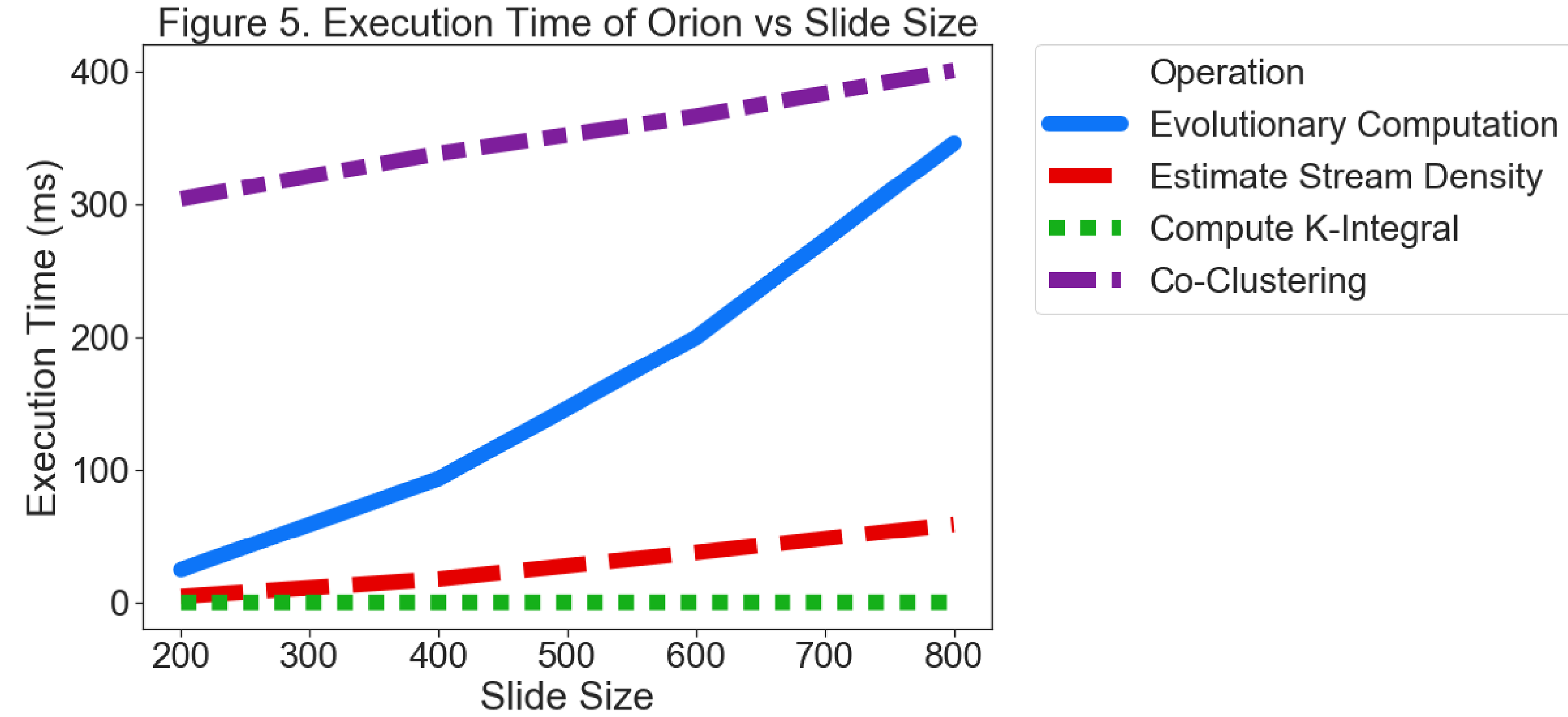


Figure 4. F-1 Score of Outlier Detection Algorithms



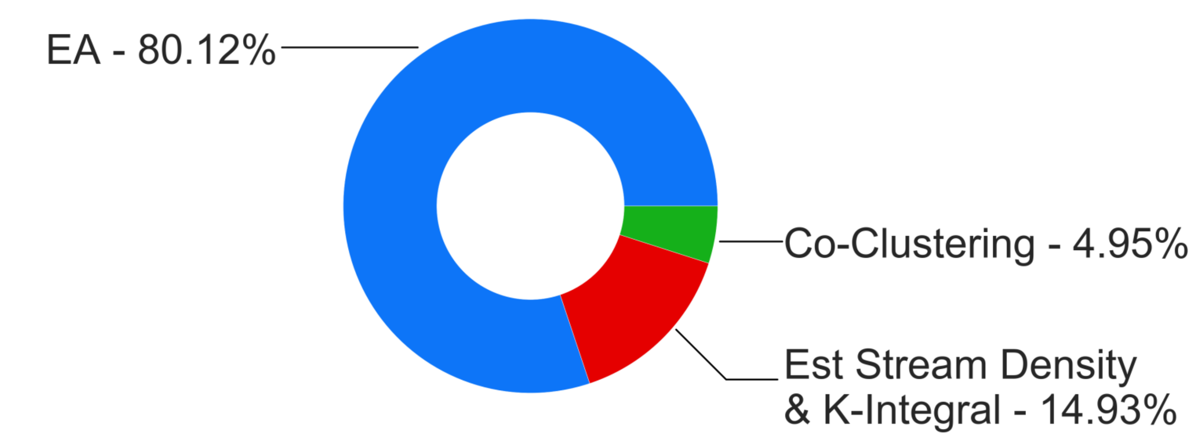
Performance Evaluation of Orion

Figure 5. Execution Time of Orion vs Slide Size



Performance Evaluation of Orion

Figure 6. Orion Algorithm Cost Breakdown



Although Co-Clustering is shown as the most expensive operation in figure 5, the operation only runs once for each window. On the other hand, evolutionary computation, stream density estimator, and k-integral estimator are executed for each data point in a window.

This makes evolutionary algorithm the most expensive operation inside Orion, which contributes to 80.12% of the total execution time for each window on average.

Discussions & Conclusions

- The results indicate that Orion has better performance than AbstractC, Approximate-Storm, and Micro-Cluster, especially in the case of the Mulcross dataset.
- Although Orion has a very low recall rate on the Tao dataset, it has a significantly higher precision rate than other algorithms, which means that the amount of normal data points falsely predicted as anomaly by Orion is extremely low. On the other hand, low recall rate suggests that Orion failed to detect most of the outliers in the dataset.
- The performance evaluation indicates that Orion is not scalable. The evolutionary computation (phase 1) is the most expensive phase and it seems to have a non-linear growth rate.
- Any further work on optimizing Orion need to focus on reducing the growth rate of the first phase. Solving the non-scalable issue of Orion could use GPU parallel computing to increase performance.

References

[1] S. Sadik, L. Gruenwald and E. Leal, "In pursuit of outliers in multi-dimensional data streams," 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, 2016, pp. 512-521.doi: 10.1109/BigData.2016.7840642

[2] Eibe Frank, Mark A. Hall, and Ian H. Witten (2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.

[3] Luan Tran, Liyue Fan, and Cyrus Shahabi. 2016. Distance-based outlier detection in data streams. Proceedings of the VLDB Endowment9, 12 (2016), 1089–1100.

[4] Distance-based outlier detection in data streams repository. http://infolab.usc.edu/Luan/Outlier/.