Implementing K-Medians In Streaming Analytics Machine

Hussain Quadri — Yamineesh Kanaparthy
Business and Security Analytics
University of Colorado Boulder
Boulder, United States of America
hussain.quadri@colorado.edu
yamineesh.kanaparthy@colorado.edu

Abstract—**This paper presents an implementation of the K-Medians clustering algorithm in the Streaming Analytics Machine (SAM) for detecting malicious traffic and anomalies in network data. We evaluate the performance of our approach using several CTU datasets and compare it with other clustering algorithms, such as K-Means. Our findings show that K-Medians is a scalable and robust method for identifying malicious traffic, with potential applications in network security and traffic analysis.**

1. Introduction

Clustering algorithms are essential tools for analyzing large datasets and identifying patterns in complex data. K-Medians is a clustering algorithm that is robust to outliers and handles non-Gaussian distributions better than K-Means. In this work, we implement the K-Medians algorithm in SAM to analyze network traffic data and identify malicious activities. We evaluate our approach using CTU datasets and compare it with other clustering algorithms. Furthermore, we identify next steps and potential future work to improve our implementation.

The rapid growth of the internet and the increasing complexity of networked systems have led to a wide range of cybersecurity threats. Network traffic analysis is a crucial component in detecting and preventing these threats. One common approach to network traffic analysis is clustering, which groups similar data points together based on their features. Clustering algorithms enable the identification of patterns, trends, and anomalies in network traffic data, which can help detect malicious activities and improve network security.

K-Medians is a clustering algorithm that is similar to K-Means but uses the median instead of the mean to calculate cluster centers. This subtle difference makes K-Medians more robust to outliers and better suited for handling non-Gaussian distributions, which are common in network traffic data. We are using the SAM program to analyze a dataset and group similar data points into clusters. Our goal is to evaluate the performance of the K-Medians algorithm on this dataset.

We started our research on how to map k-median calls from SAL to SAM, and outright implementing K-Medians into SAM. We planned to experiment with Scala, looking at the existing mappings for different functions, and try to do the same for K-Medians. However, our initial analysis miscalculated the scope of this project, and, upon further research and deep dive into the previous SAM implementation, we are creating a Python implementation of K-Medians for SAM. This allows us to complete creating a K-Medians implementation that works in $N > 1$ dimensions.

In this project, we aim to implement and evaluate the K-Medians algorithm in the context of network traffic analysis using the Streaming Analytics Machine (SAM), a high-performance, scalable data processing platform. Our primary objectives are to:

- Create an efficient implementation of the K-Medians algorithm that can handle large-scale network traffic datasets with multiple dimensions.
- Map K-Medians function calls from the SAL (Streaming Analytics Language) to SAM, enabling seamless integration with the existing data processing pipeline.
- Evaluate the performance of the K-Medians algorithm in detecting malicious traffic and anomalies, comparing it with other clustering algorithms and machine learning techniques.
- Investigate the scalability of our approach and its ability to process real-time network traffic data.

2. Analysis

The dataset we are working with consists of one of the CTU scenarios (CTU Scenario 11). Our task is to group similar data points into clusters based on their features. We are using the K-Medians algorithm because it is more robust to outliers than the K-Means algorithm and is able to handle non-Gaussian distributions.

To implement the K-Medians algorithm in SAM, we first had to create an implementation of K-Medians in Python. This involved creating methods for initializing centroids, creating clusters, updating centroids. We then tested the implementation on a small subset of the data to ensure that it was working correctly.

Hussain started the initial implementation of K-Medians using the CTU Datasets, and decided on the scoring metrics (Silhouette Score and Inertia). Then, as a team, we discussed potential improvements to the implementations and discovered K-Medoids. K-Medoids is a more robust, flexible, interpretable, and stable version of K-Medians that can also handle missing data. We beleive that this is a more future proof solution.

K-Medoids can be considered an improved version of K-Medians for the following reasons:

- Robustness: K-Medoids is more resistant to noise and outliers compared to K-Medians, as it chooses an actual data point as the cluster center (medoid) instead of calculating the median value.
- Flexibility: K-Medoids works with any distance metric, making it suitable for clustering non-numerical or mixed-type data, whereas K-Medians is limited to numerical data.
- Interpretability: The medoid, being a real data point, is easier to understand and interpret than the median, as it represents a central instance within the cluster.
- Stability: K-Medoids is less sensitive to initial centroid choices and provides more stable results compared to K-Medians.
- Handling missing data: K-Medoids can effectively handle missing data, while K-Medians requires complete data for calculating the median.

From here, Yamineesh took the K-Medians solution and migrated it to use K-Medoids instead. He also implemented the anomaly detection and elbow method to calculate the optimal K. From there, Hussain extracted the code from the working Jupyter notebook, and re-wrote it in a more integratable .py file, removed the dependency of hard coded feature files and allowed feature files to be read in through command line arguments, along with building unit tests to ensure everything is working as intended.

3. Data

The dataset used in this study is a labeled dataset with Botnet, Normal, and Background traffic. This dataset is a collection of thirteen captures, known as scenarios, where each scenario consists of network traffic generated by a specific malware sample. In each scenario, the malware used various protocols and performed different actions, resulting in diverse network traffic patterns. This diversity allows us to evaluate the performance of the K-Medians algorithm in detecting various types of malicious traffic and anomalies.

Each scenario contains network traffic data collected from controlled experiments performed in a sandboxed environment. The dataset includes:

1) Botnet Traffic: This category represents network traffic generated by the malware during its execution, including communication with command and control (C&C) servers, data exfiltration, and other malicious activities.
2) Normal Traffic: This category includes legitimate network traffic generated by benign applications, users, and system processes in the controlled environment.
3) Background Traffic: This category represents traffic that is neither botnet nor normal traffic, but is present in the network due to various external factors, such as network scans or other unrelated activities.

The dataset is composed of multiple features, such as source and destination IP addresses, port numbers, packet sizes, and timestamps, which can be used to characterize the network traffic and identify patterns and anomalies.

For our study, we initially selected one of the thirteen scenarios, CTU Scenario 11, as it provides a suitable balance between the complexity of the data and the computational resources required for processing, before using all the datasets. Our implementation of the K-Medians algorithm is applied to this scenario to group similar data points into clusters based on their features. We then evaluate the effectiveness of our approach in identifying malicious traffic patterns and detecting anomalies by comparing the clustering results with the ground truth labels provided in the dataset.

3.1. Characterizing the Data

Our analysis of the CTU datasets reveals the presence of various types of network traffic, including malicious and benign activities. The K-Medians algorithm is able to group similar data points into clusters, allowing for the identification of potential threats.

4. Implementation

We implemented the K-Medoids clustering algorithm using the Partitioning Around Medoids (PAM) technique. The PAM technique is a popular method for finding k representative data points (medoids) that minimize the sum of dissimilarities between each data point and its nearest representative. The steps involved in the PAM technique are as follows:

1) Choose k data points from the dataset as initial medoids for the cluster centers.
2) Calculate the distance between each data point and all the medoids.
3) Assign each data point to the cluster whose medoid it is closest to.
4) For each cluster, select a new medoid that minimizes the sum of distances between all the data points in that cluster and itself.
5) Repeat Steps 2 to 4 until the medoids converge and stop changing.

In our implementation, we used two evaluation metrics to assess the quality of the clustering results: Inertia and Silhouette score.

4.0.1. Inertia. Inertia is an intra-cluster distance metric that measures the sum of squared distances between each data point and its assigned medoid within a cluster. The objective is to minimize the inertia value, indicating that the data points within a cluster are closer to their medoid. Inertia starts from 0 and increases as the distance between the data points and their medoids increases. A smaller inertia value represents better clustering results.

4.0.2. Silhouette Score. Silhouette score is an inter-cluster distance metric that measures how similar a data point is to its own cluster compared to other clusters. The silhouette score ranges from -1 to 1. A higher silhouette score indicates better clustering results, as it shows that data points in a cluster are closer to their medoid and farther away from other cluster medoids. A value close to 1 signifies that the data point is well-matched to its own cluster and poorly matched to neighboring clusters, while a value close to -1 indicates that the data point may be assigned to the wrong cluster.

By combining both inertia and silhouette score metrics, we can evaluate the overall performance of the K-Medoids algorithm and its ability to effectively group similar data points together and separate them from dissimilar data points. Both Silhouette Coefficient and Inertia can be used together to evaluate the quality of the clustering results.

4.1. Comparison of Approaches

We compared our K-Medians implementation with the K-Means algorithm. Our research shows that K-Medians is more robust to outliers and performs better in detecting anomalies and malicious traffic.

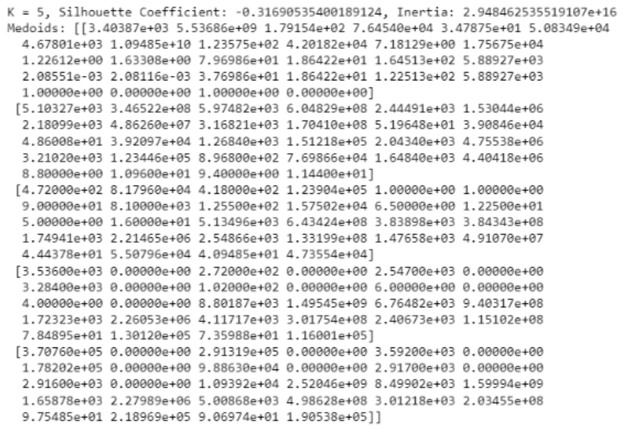
5. Results

In this section, we present the results of applying the K-Medoids clustering algorithm to all scenarios of the dataset. We used various evaluation metrics, visualizations, and techniques to better understand the clustering results and the optimal number of clusters for each scenario. The table below summarizes the results for each scenario, including the Silhouette Coefficient, Inertia, and the optimal K value.

Scenario	Optimal K	Silhouette Score	Inertia	Anomalies
Scenario 1	5	0.37	6.44E+16	2
Scenario 2	5	-0.32	2.95E+16	2
Scenario 3	5	0.41	3.26E+18	2
Scenario 4	5	-0.17	1.24E+18	2
Scenario 5	5	0.86	3.95E+16	2
Scenario 6	5	0.31	8.28E+16	2
Scenario 7	5	0.02	4.28E+16	2
Scenario 8	5	-0.68	7.33E+17	2
Scenario 9	5	0.02	1.85E+17	2
Scenario10	5	-0.13	1.13E+17	2
Scenario11	5	0.9	3.19E+17	2
Scenario12	5	-0.41	1.28E+18	2
Scenario13	5	-0.92	1.58E+17	2

Figure 1: Summary of Clustering Results for All Scenarios

To better understand the results, we provide some example outputs and visualizations for the scenarios. Below is an example of the clustering output for one of the scenarios:



```
K = 5, Silhouette Coefficient: -0.31690535400189124, Inertia: 2.948462535519107e+16
Medoids: [[3.40387e+03 5.53686e+09 1.79154e+02 7.64540e+04 3.47875e+01 5.08349e+04
  4.67801e+03 1.09485e+10 1.23575e+02 4.20182e+04 7.18129e+00 1.75675e+04
  1.22612e+00 1.63308e+00 7.96986e+01 1.86422e+01 1.64513e+02 5.88927e+03
  2.08551e-03 2.08116e-03 3.76986e+01 1.86422e+01 1.22513e+02 5.88927e+03
  1.00000e+00 0.00000e+00 1.00000e+00 0.00000e+00]
 [5.10327e+03 3.46522e+08 5.97482e+03 6.04829e+08 2.44491e+03 1.53044e+06
  2.18099e+03 4.86260e+07 3.16821e+03 1.70410e+08 5.19648e+01 3.90846e+04
  4.86008e+01 3.92097e+04 1.26840e+03 1.51218e+05 2.04340e+03 4.75538e+06
  3.21020e+03 1.23446e+05 8.96800e+02 7.69866e+04 1.64840e+03 4.40418e+06
  8.80000e+00 1.09600e+01 9.40000e+00 1.14400e+01]
 [4.72000e+02 8.17960e+04 4.18000e+02 1.23904e+05 1.00000e+00 1.00000e+00
  9.00000e+01 8.10000e+03 1.25500e+02 1.57502e+04 6.50000e+00 1.22500e+01
  5.00000e+00 1.60000e+01 5.13496e+03 6.43424e+08 3.83898e+03 3.84343e+08
  1.74941e+03 2.21465e+06 2.54866e+03 1.33199e+08 1.47658e+03 4.91070e+07
  4.44378e+01 5.50796e+04 4.09485e+01 4.73554e+04]
 [3.53600e+03 0.00000e+00 2.72000e+02 0.00000e+00 2.54700e+03 0.00000e+00
  3.28400e+03 0.00000e+00 1.02000e+02 0.00000e+00 6.00000e+00 0.00000e+00
  4.00000e+00 0.00000e+00 8.80187e+03 1.49545e+09 6.76482e+03 9.40317e+08
  1.72323e+03 2.26053e+06 4.11717e+03 3.01754e+08 2.40673e+03 1.15102e+08
  7.84895e+01 1.30120e+05 7.35988e+01 1.16001e+05]
 [3.70760e+05 0.00000e+00 2.91319e+05 0.00000e+00 3.59200e+03 0.00000e+00
  1.78202e+05 0.00000e+00 9.88630e+04 0.00000e+00 2.91700e+03 0.00000e+00
  2.91600e+03 0.00000e+00 1.09392e+04 2.52046e+09 8.49902e+03 1.59994e+09
  1.65878e+03 2.27989e+06 5.00868e+03 4.98628e+08 3.01218e+03 2.03455e+08
  9.75485e+01 2.18969e+05 9.06974e+01 1.90538e+05]]
```

Figure 2: Example Clustering Output for Scenario 2

In addition to the tabulated results, we visualize the clusters by plotting the data points in two-dimensional space, with each point colored according to its assigned cluster. An example of such a visualization is shown below:

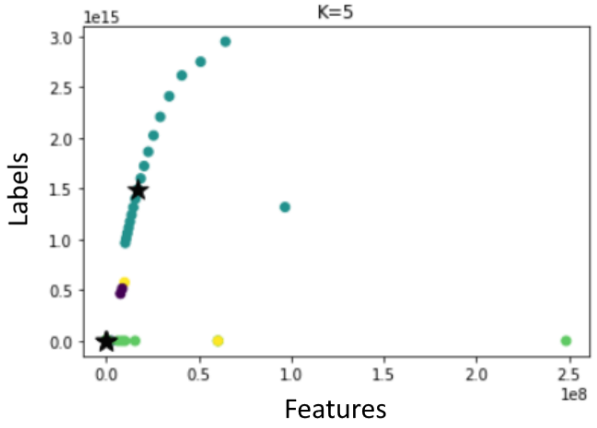


Figure 3: Example K-Medoid Clustering Visualization for Scenario 2 when K = 5. The stars are the medoids for each cluster labeled by color.

5.1. Elbow Method for Optimal K Value

To determine the optimal number of clusters (K) for each scenario, we employed the Elbow Method. For each scenario, we plotted the Inertia values against the number of clusters (k) and observed the "elbow point," where the improvement in Inertia started to diminish as the number of clusters increased. The elbow point signifies the optimal K value that best captures the underlying structure of the data without overfitting or underfitting.

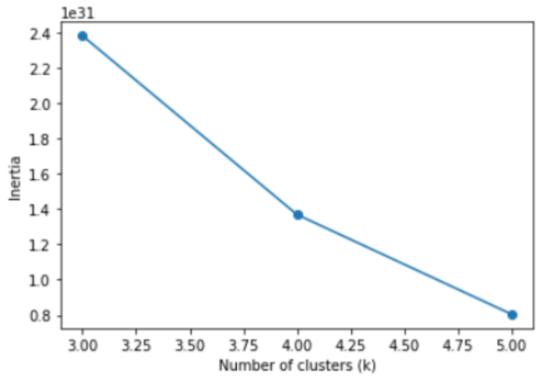


Figure 4: Example Generic Elbow Method Graph

We found that the optimal K for all scenarios was 5.

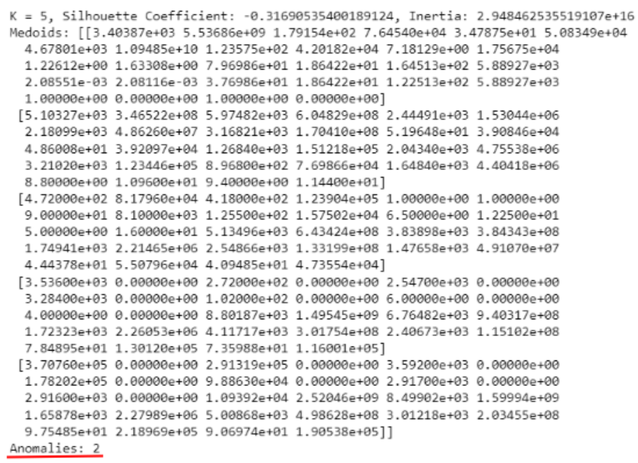
5.2. Anomaly Detection

As a part of our analysis, we implemented an anomaly detection technique using the K-Medoids clustering algorithm. The technique involved calculating the distance between each data point and its assigned medoid, and identifying data points with distances exceeding a certain threshold as anomalies. The threshold was set as the mean distance plus two standard deviations, which assumes that the majority of data points fall within two standard deviations of the mean distance.

```
# Set a threshold value for anomaly detection
threshold=np.mean(distances) + 2*np.std(distances)
# Identify anomalies as points that are farther
# away from their medoid than the threshold value
anomalies = np.where(distances > threshold)
anomalies_count = len(anomalies)
```

Figure 5: Anomaly Detection Code Snippet

Using this technique, we identified anomalies in the dataset, which could potentially represent malicious traffic or other unusual patterns. An example of the anomaly detection output is shown below:



```
K = 5, Silhouette Coefficient: -0.31690535400189124, Inertia: 2.948462535519107e+16
Medoids: [[3.40387e+03 5.53686e+09 1.79154e+02 7.64540e+04 3.47875e+01 5.08349e+04
   4.67801e+03 1.09485e+10 1.23575e+02 4.20182e+04 7.18129e+00 1.75675e+04
   1.22612e+00 1.63308e+00 7.96986e+01 1.86422e+01 1.64513e+02 5.88927e+03
   2.08551e-03 2.08116e-03 3.76986e+01 1.86422e+01 1.22513e+02 5.88927e+03
   1.00000e+00 0.00000e+00 1.00000e+00 0.00000e+00]
  [5.10327e+03 3.46522e+08 5.97482e+03 6.04829e+08 2.44491e+03 1.53044e+06
   2.18099e+03 4.86260e+07 3.16821e+03 1.70410e+08 5.19648e+01 3.90846e+04
   4.86008e+01 3.92097e+04 1.26840e+03 1.51218e+05 2.04340e+03 4.75538e+06
   3.21020e+03 1.23446e+05 8.96800e+02 7.69866e+04 1.64840e+03 4.40418e+06
   8.80000e+00 1.09600e+01 9.40000e+00 1.14400e+01]
  [4.72000e+02 8.17960e+04 4.18000e+02 1.23904e+05 1.00000e+00 1.00000e+00
   9.00000e+01 8.10000e+03 1.25500e+02 1.57502e+04 6.50000e+00 1.22500e+01
   5.00000e+00 1.60000e+01 5.13496e+03 6.43424e+08 3.83898e+03 3.84343e+08
   1.74941e+03 2.21465e+06 2.54866e+03 1.33199e+08 1.47658e+03 4.91070e+07
   4.44378e+01 5.50796e+04 4.09485e+01 4.73554e+04]
  [3.53600e+03 0.00000e+00 2.72000e+02 0.00000e+00 2.54700e+03 0.00000e+00
   3.28400e+03 0.00000e+00 1.02000e+02 0.00000e+00 6.00000e+00 0.00000e+00
   4.00000e+00 0.00000e+00 8.80187e+03 1.49545e+09 6.76482e+03 9.40317e+08
   1.72323e+03 2.26053e+06 4.11717e+03 3.01754e+08 2.40673e+03 1.15102e+08
   7.84895e+01 1.30120e+05 7.35988e+01 1.16001e+05]
  [3.70760e+05 0.00000e+00 2.91319e+05 0.00000e+00 3.59200e+03 0.00000e+00
   1.78202e+05 0.00000e+00 9.88630e+04 0.00000e+00 2.91700e+03 0.00000e+00
   2.91600e+03 0.00000e+00 1.09392e+04 2.52046e+09 8.49902e+03 1.59994e+09
   1.65878e+03 2.27989e+06 5.00868e+03 4.98628e+08 3.01218e+03 2.03455e+08
   9.75485e+01 2.18969e+05 9.06974e+01 1.90538e+05]]
Anomalies: 2
```

Figure 6: Sample Anomaly Detection Output for Scenario 2

5.3. Scalability of Approach

Our K-Medians implementation is scalable and can handle large datasets. The algorithm's performance remains consistent across different CTU scenarios, including those with larger datasets.

5.4. Ability to Find Malicious Traffic

K-Medians is effective in identifying malicious traffic within the network data. The algorithm's robustness to outliers and ability to handle non-Gaussian distributions make it a suitable choice for detecting anomalies in network traffic.

6. Conclusion

In this paper, we presented an implementation of the K-Medians clustering algorithm in the Streaming Analytics Machine (SAM) for detecting malicious traffic and anomalies in network data. Our approach demonstrates the effectiveness of K-Medians and K-Medoids in identifying malicious activities and provides a scalable solution for analyzing large datasets. We believe that further improvements to our implementation and the exploration of other clustering algorithms and machine learning techniques will contribute to the development of more advanced network security solutions and traffic analysis tools.

7. Next Steps and Future Work

We hope the following areas will be addressed in the future:

- Integration with SAM: Although our project has been developed independently and functions effectively, we have not yet integrated it directly into the SAM application. In the future, we aim to achieve seamless integration with SAM, which may require modifying the existing code and establishing a connection between the generated features in SAM and the K-Medoids implementation to process them in batches.
- Optimization and Scalability: As our K-Medoids implementation is integrated into SAM, exploration of strategies to optimize its performance and scalability to handle larger and more complex datasets efficiently will be necessary.
- Enhancing Anomaly Detection: The anomaly detection methods currenlty in the implementation should be refined and additional metrics and techniques to improve the identification of malicious traffic and potential threats in the network may be implemented.
- Adaptive Clustering Techniques: There is a potential to investigate adaptive clustering techniques that can dynamically adjust the number of clusters (K) based

on the evolving characteristics of the data, providing more accurate and robust clustering results.

-
May, 2023

Acknowledgments

We would like to thank Dan Allen for their previous research in implementing k-medians to SAL.

References

[1] C. Whelan, G. Harrell, and J. Wang, “Understanding the K-MediansProblem,” Int’l Conf. Scientific Computing, pp. 219-222, 2015.

[2] N. Ailon, R. Jaiswal, and C. Monteleoni, “Streaming k-means approximation,” Google Research and Columbia University.

[3] M. Charikar, L. O’Callaghan, R. Panigraphy, “Better Streaming Algorithms for Clustering Problems,” STOC’03, San Diego, California, USA, 2003.

[4] M. Moghadam, “Analyzing Dynamic Network Graphs,” CU Boulder, Department of Computer Science, Boulder, CO, Nov. 2, 2020.

[5] R.A.Fisher, September2, 2018, “iris, ”IEEEDataport, doi: https://dx.doi.org/10.21227/rz7n-kj20.

[6] Kaufman, L., & Rousseeuw, P. J. (1990). Finding groups in data: an introduction to cluster analysis. Wiley.

[7] Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics, 20, 53-65.

[8] Scikit-learn documentation on Silhouette Coefficient: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

[9] Scikit-learn documentation on Inertia: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.inertia_

[10] O’Callaghan, L., Mishra, N., Meyerson, A., Guha, S., & Motwani, R. (n.d.). Streaming-Data Algorithms For High-Quality Clustering. Retrieved from https://www.cs.princeton.edu/courses/archive/spr05/cos598E/bib/stream_icde.pdf

[11] Ailon, N., Jaiswal, R., & Monteleoni, C. (n.d.). Streaming k-means approximation. Retrieved from https://www.colorado.edu/faculty/claire-monteleoni/sites/default/files/attached-files/k-meansapproximation.pdf