Robust Clustering of Data Streams using Incremental Optimization

Basheer Hawwash and Olfa Nasraoui

Knowledge Discovery and Web Mining Lab
Computer Engineering and Computer Science Department
University of Louisville, Louisville, KY, USA

Abstract. In many applications, it is useful to detect the evolving patterns in a data stream, and be able to capture them accurately (e.g. detecting the purchasing trends of customers over time on an e-commerce website). Data stream mining is challenging because of harsh constraints due to the continuous arrival of huge amounts of data that prevent unlimited storage and processing in memory, and the lack of control over the data arrival pattern. In this paper, we present a new approach to discover the evolving dense clusters in a dynamic data stream by incrementally updating the cluster parameters using a method based on robust statistics. Our approach exhibits robustness toward an unknown number of outliers, with no assumptions about the number of clusters. Moreover, it can adapt to the evolution of the clusters in the input data stream.

1 Introduction

Data Stream Mining has attracted increasing attention in recent years [2,3,8], due to the proliferation of applications that generate massive streams of data. Data streams are characterized by a huge throughput which makes it hard if not impossible to store every single data point and process it, therefore each new data point can be processed only once. Moreover, there is no control or expectation on the order of the data arrival, which adds more difficulties when trying to extract useful information from the data streams. An example of a data stream are the transactions performed on a busy e-commerce website. The output of the data stream mining, in this case, would be the purchasing patterns of the customers.

To meet the challenge of mining data streams, clustering algorithms should be scalable enough to cope with huge throughputs of data [1]. Several clustering algorithms have been designed to deal with large amounts of data by updating the clustering model incrementally or by processing data points in small batches. For example, STREAMS [2] first processes data streams in batches which generates a set of weighted cluster centers for each batch; then the set of cluster centers for all batches are clustered. However, it is not suitable to handle or detect outliers because it is based on minimizing the sum of squared distances which is very sensitive to extreme outliers [7]. BIRCH [3] is another incremental

clustering algorithm which uses a set of statistical measurements to capture the closeness of data points. It updates these measurements incrementally and stores the clusters in a tree of Cluster Features (CF). One of BIRCH's limitations is that it treats all the data points in the same way without considering the time when the data points arrived. Hence, it cannot handle the *evolution* of data, where a cluster can disappear, change or merge with other clusters, which is one of the inherent characteristics of a data stream. In [8], an algorithm for clustering data streams, known as TRAC-Streams, was presented. It relied on incremental processing and robust statistics for learning an evolving model of the clusters in a data stream. Its objective function consists of two terms: the first term minimizes the scaled distances of the good points to the optimal cluster locations to achieve robustness, while the second term maximizes a soft estimate of the cluster cardinality (sum of weights) which ensures that as many good points as possible are used in the estimation process. However, a constant factor α is used to balance the two terms, and had to be chosen carefully and depends on the data's dimensionality. DBSCAN [9] is a density based clustering technique that relies on labeling each data point as a core point, border point or noise point based on the number of points within a predefined distance. This labeling relies on two important parameters, MinPts (a threshold on density around a core point) and Eps (a measure of scale used as a threshold on the size of a dense region). Its incremental version [10] (Inc-DBSCAN) achieves the same goal in an online manner by attempting to label each new data point into one of the above labels and updating (merging, elimination, etc) the clusters accordingly. It was proved that Incremental DBSCAN results in clusters that are equivalent to the original DBSCAN, however it has the same quadratic complexity in terms of the data size.

In this paper, we present a novel algorithm for extracting evolving clusters from a massive data stream in one pass, while being able to resist and detect the presence of outliers in the data stream. Our approach is rooted in robust statistics [7] since it uses a robust estimation of centroids (location) as well as a robust and dynamic estimation of the cluster scales. It is also based on a robust distribution independent statistical test (Chebyshev) for the detection of outliers and for the mergal of compatible clusters, thus assuring robustness and compactness of the extracted cluster model. The statistically based robustness to noise, dynamic estimation of scales, and fast analytical optimization constitute the distinguishing contributions of our approach compared to existing stream clustering methods. The main distinction between RINO-STREAMS and TRAC-STREAMS is in the simpler objective function which consists of only one term (the density) for the former, thus eliminating the need to balance two terms in the later. This results in a simpler and faster optimization. Moreover, RINO-STREAMS has a better cluster merging technique.

The rest of the paper is organized as follows. In Section 2, we describe the proposed approach (RINO-STREAMS), while in Section 3, we present the experimental results that validate our algorithm. Finally, we conclude in Section 4.

2 RINO-STREAMS

The proposed algorithm, RINO-STREAMS, is based on incrementally updating the clustering model using each newly arrived data point, and thus maintaining the model summaries (clusters) over time. The data stream X consists of a set of data instances or points that are indexed based on the order of their arrival, and presented as: $x_1, x_2 ...., x_N$, where N is the size of the data stream. Each cluster i at time n (i.e. after receiving n points) is defined using its centroid $c_{i,n}$, its scale $\sigma^2_{i,n}$ and its age t_i. The centroid represents the location of the cluster center with respect to other clusters at any time, while the scale represents the influence zone around the centroid. The data points are weighted using a weight function that decreases with the distance from the cluster prototype/centroid as well as with the time at which the data arrived. Hence, newer data points would have more effect on the model than older ones, thus capturing the evolution of clusters over time. Using an exponentially decaying forgetting factor was proved to be successful in modeling evolution as was done in previous work [4,5,6]. The age is the difference between the current time/iteration or step (n) and the time when the cluster was first created.

Definition 1. ***Adaptive Robust Weight:*** *For the i^{th} cluster, $C_i, i = 1, ..., K$, the robust weight of the j^{th} data point at time n (i.e. $x_1, x_2, ..., x_n$ points are encountered so far) is defined as:*

$$w_{ij,n} = e^{-\left(\frac{d^2_{ij}}{2\sigma^2_{i,n}} + \frac{n-j}{\tau}\right)} \tag{1}$$

where τ is an optional application-dependent parameter that controls the time decay rate of old data points (when needed), and how much importance is given to newer data points. d^2_{ij} is the Euclidean distance from the j^{th} data point to the i^{th} cluster's centroid c_i, and $\sigma^2_{i,n}$ is the scale of cluster i at time/step n and relates to the size of the influence zone of the cluster where every point outside this zone is considered an outlier with respect to cluster c_i. The second term represents a forgetting factor, where the weight of the data point j decreases geometrically by the value of $n - j$, hence a new data would have a higher weight since the forgetting factor would be close to zero, while an older point would have a large value for the forgetting factor which results in a smaller weight. Assuming that the parameters of the model don't change significantly with every new point, then each old point's weight, and thus its influence on the cluster parameter estimates, would decrease as follows:

$$w_{ij,n} = e^{\frac{-1}{\tau}} w_{ij,n-1} \tag{2}$$

After encountering n data points, we search for the cluster centroids $c_{i,n}$ and scales $\sigma^2_{i,n}$ by optimizing the density objective function $\delta_{i,n}$ as follows:

$$\underset{c_{i,n}, \sigma^2_{i,n}}{max} \left\{ \delta_{i,n} = \sum_{i=1}^{K} \sum_{j=1}^{n} \frac{w_{ij,n}}{\sigma^2_{i,n}} \right\} i = 1, ..., K \tag{3}$$

The robust weight $w_{ij,n}$ can also be considered as a typicality or possibilistic degree of membership [13] of the point j in the cluster i after encountering n points, hence the sum of the weights for each cluster represents a soft cardinality of that cluster. A high cardinality is desirable because it means that this cluster represents enough points to justify its existence. A small scale means that it is a good and compact cluster. The density of the cluster, $\delta_{i,n}$, combines the previous two metrics of the cluster, and hence it increases as the cardinality increases and the scale decreases. The advantage of optimizing the density, which combines the two metrics, is that judging the quality of the cluster using only the sum of weights (the numerator) is not enough, because a cluster with a large number of similar points which are not confined within a small zone is not desirable from the point of view of density.

Optimizing the density objective function is done using alternating optimization, where finding the optimal centroid is done by fixing all other variables, then the same is done when optimizing the scale. This optimization technique is also used in the EM algorithm [11]. Below we present the optimal update theorems for the cluster centroids and scale, while omitting the proofs due to the paucity of space. The proofs are similar to the approach followed in [8].

Theorem 1. ***Optimal Incremental Centroid Update****: Given the previous centroids, $c_{i,n-1}$, and assuming that the scales do not change much relative to the scale that resulted from the previous point, the new centroid that optimize* (3) *after the n^{th} point is given by:*

$$c_{i,n} = \frac{e^{\frac{-1}{\tau}} \left(\sum_{j=1}^{n-1} w_{ij,n-1} x_j \right) + w_{in,n} x_n}{e^{\frac{-1}{\tau}} \left(\sum_{j=1}^{n-1} w_{ij,n-1} \right) + w_{in,n} x_n} \tag{4}$$

The first term in the numerator (and denominator) represents the previous knowledge about the location of the centroid obtained from the points $(x_1, ..., x_{n-1})$, and this term is multiplied by the forgetting factor to reduce its effect on the new updated centroid and give more emphasis to the new data point. This comes directly from the fact that the weight of each point is reduced by $e^{\frac{-1}{\tau}}$ as given by equation (2). The second term represents the weighted effect of the new data point on the location of the cluster.

Theorem 2. ***Optimal Incremental Scale Update****: Given the previous scales, $\sigma^2_{i,n-1}$, the new scale that optimize* (3) *after the arrival of the n^{th} point is given by:*

$$\sigma_{i,n} = \frac{e^{\frac{-1}{\tau}} \left(\sum_{j=1}^{n-1} w_{ij,n-1} d^2_{ij} \right) + w_{in,n} d^2_{in}}{2e^{\frac{-1}{\tau}} \left(\sum_{j=1}^{n-1} w_{ij,n-1} \right) + 2w_{in,n}} \tag{5}$$

Similar to the centroid update equation, the first term in the numerator represents the sum of the contributions of all previous points $(x_1, .., x_{n-1})$, and this value is also penalized by the forgetting factor. The second term represents the new information obtained from the new data point.

Following these two update equations, the algorithm would update the cluster parameters with the arrival of a new data point incrementally, and it would keep as a summary of the data stream only the previous centroids ($c_{i,n}$), scales ($\sigma^2_{i,n}$) and the sums of weights ($W_{i,n} = \sum_{j=1}^{n} w_{ij,n}$) for each cluster.

2.1 Detecting outliers

An outlier is defined as a data point that doesn't belong to any of the existing clusters (i.e. not in their influence zone). If a point is determined to be an outlier with respect to all existing clusters, then a new cluster will be created with the point itself being its centroid. This newly created cluster will be allowed a grace period, t_{mature}, and if after this threshold, it is still weak (it has a density less than a threshold δ_{min}), then it will be considered an outlying cluster and will be deleted. To detect outliers we are going to use the Chebyshev bound, an upper tail bound that bounds the total probability that some random variable is in the tail of the distribution, i.e. far from the mean. One important property of Chebyshev bounds is that they rely on no assumptions about the distribution of the data, rather they assume that a reliable scale estimate is available, which is the case using RINO-STREAMS by virtue of the robust density and scale optimization.

Chebyshev Bounds: The Chebyshev bound for a random variable Y with standard deviation σ for any real number $t > 0$ is given by:

$$Pr\{|Y - \mu| \geq t\sigma\} \leq \frac{1}{t^2} \tag{6}$$

This bound will allow us to design an outlyingness test for any new data point with respect to cluster C_i with significance probability $1/t^2$. The Chebyshev inequality can be rearranged as follows:

$$Pr\left\{e^{\frac{-|Y-\mu|^2}{2\sigma^2}} \geq e^{\frac{-t^2}{2}}\right\} \leq \frac{1}{t^2} \;\; Or \;\; Pr\left\{w_{ij} \geq e^{\frac{-t^2}{2}}\right\} \leq \frac{1}{t^2} \tag{7}$$

which means that if the robust weight w_{ij} of the point j with respect to cluster C_i is less than the constant value of $e^{\frac{-t^2}{2}}$, then point j is considered to be an outlier with a significance probability of $\frac{1}{t^2}$. Moreover, the Chebyshev Bound is used when finding the cardinality of the cluster when doing the evaluation (i.e. all points that pass the Chebyshev test are considered part of the cluster with a significance probability of $1 - \frac{1}{t^2}$).

Definition 2. ***Outlier with Chebyshev probability of*** $\frac{1}{t^2}$*:* *x_j is an outlier with respect to cluster C_i at time n with a significance probability of $\frac{1}{t^2}$ if:*

$$w_{ij,n} \geq e^{\frac{-t^2}{2}} \tag{8}$$

2.2 Cluster mergal and splitting

We further use the Chebyshev bound to design a compatibility test for merging clusters C_i and C_k. This is done by checking their mutual Chebyshev bounds with significance probability $1/t^2$: Given the distance d_{ik} between the centroids c_i and c_k, then using (6), the clusters are merged if:

$$d_{ik}^2 < t^2\sigma_i^2 \ \& \ d_{ik}^2 < t^2\sigma_k^2 \tag{9}$$

When the clusters C_i and C_k are merged, the centroid of the new cluster is a weighted centroid as follows

$$c_{new,n} = \frac{c_{i,n}\sum_{j=1}^{n} w_{ij,n} + c_{k,n}\sum_{j=1}^{n} w_{kj,n}}{\sum_{j=1}^{n} w_{ij,n} + \sum_{j=1}^{n} w_{kj,n}} \tag{10}$$

and the scale is weighted as follows:

$$\sigma_{new,n}^2 = \frac{\sigma_{i,n}^2\sum_{j=1}^{n} w_{ij,n} + \sigma_{k,n}^2\sum_{j=1}^{n} w_{kj,n}}{\sum_{j=1}^{n} w_{ij,n} + \sum_{j=1}^{n} w_{kj,n}} \tag{11}$$

and the age t_{new} is $max(t_i, t_k)$. Moreover a test is done to eliminate bad clusters whose density is less than a threshold (δ_{min}) and is mature enough (i.e. $t_i > t_{mature}$). Splitting clusters in RINO-STREAMS occurs naturally and does not require a special treatment (see experiments in Section 3.3). A cluster split occurs when points from a cluster evolve in two or more different directions. The complete steps of RINO-STREAMS are listed in Algorithm 1. The input parameters to RINO-STREAMS include the maximum number of clusters K_{max} which is an upper bound of the allowed number of clusters which constraints the maximal amount of memory devoted to storing the cluster model, the initial scale σ_0^2 which is assigned to the newly created cluster, the density threshold δ_{min} which is used to ensure that only good clusters with high density are kept, the maturity age t_{mature} which provides the newly created clusters with grace period before testing its density quality, the forgetting factor τ which controls the decay in the data point weights over time and the Chebyshev bound constant t that is used in equations (7) and (9).

2.3 Complexity

For each new data point, RINO-STREAMS computes the distance and the weights with respect to all the clusters in ζ, which is done in linear steps. Since the clustering model is updated incrementally, then nothing is recomputed from scratch, and hence the computational complexity of RINO-STREAMS is $O(NK^2)$ where N is the size of the data stream and K is the number of clusters discovered in the data set ($0 \leq K \leq K_{max}$) which is a very small value compared to N. Moreover, the memory requirements of RINO-STREAM are linear with the number of clusters, because at any point of time only the clusters properties (c_i, σ_i^2, W_i) are kept besides the new data point. This memory is constrained

Algorithm 1 RINO-STREAMS

```
Input: Maximum number of clusters (K_max), Initial scale (σ_0^2), density threshold (δ_min),
maturity age (t_mature), forgetting factor (τ), Chebyshev Bound constant (t)
Output: Cluster model after n points ζ = C_1 ∪ C_2 .... ∪ C_K where C_i = (c_{i,n}, σ^2_{i,n}, t_i, W_{i,n}), where
W_i = Σ_{j=1}^{n} w_{ij,n}
FOR n = 1 TO N DO
    //single pass over the data stream
    Compute the distances, d_in, and robust weights w_{in,n} between x_n and clusters
C_i ∀i = 1, .., K
    IF K < K_max And x_n is an outlier with respect to all clusters in ζ
        // Create a new cluster centered on x_n
        K = K + 1
        c_K = x_n //centroid
        σ^2_K = σ^2_0 //initial scale
        t_K = 0 //initial age
        W_K = 1 //initial sum of robust weights
        δ_K = 1/σ^2_0 // initial density
    END IF
    FOR each cluster C_i ∀i = 1, .., K
        //Update the compatible clusters if x_n is not an outlier
        IF x_n is NOT an outlier with respect to cluster i (see Def. 2)
            Update c_{i,n} using (4)
            Update σ^2_{i,n} using (5)
            Update sum of weights: W_{i,n} = e^{-1/τ} W_{i,n-1} + w_{in}
            Update density δ_{i,n} = W_{i,n} / σ^2_{i,n}
            Update age t_i = t_i + 1
        END IF
    END FOR
    FOR each pair of clusters C_i & C_k ∀i, k = 1, .., K
        IF C_i and C_k are Chebyshev-compatible using equation (9)
            Merge clusters C_i and C_k using equations (10) and (11)
        END IF
    END FOR
    FOR each cluster C_i ∀i = 1, .., K
        IF t_i > t_mature & δ_i < δ_min //remove mature clusters with low density
            ζ = ζ − C_i
        END IF
    END FOR
END FOR
```

by $K_{max} \times B_f \times (Dim + 3)$ where B_f is the number of bytes needed to store one floating point number, and Dim is the dimensionality (number of data attributes) and refers to the size of the centroid c_i and the number 3 refers to the scale σ_i^2, soft cardinality W_i and age t_i.

3 Experimental Results

RINO-STREAMS was evaluated using three synthetic datasets generated using a random Gaussian generator: **DS5** has five clusters, **DS8** has eight clusters and **DS16** has 16 clusters. For each dataset, nine stream variations were created by adding different percentages of noise and by changing the order of data arrival. We will be using a brief code that describes the experiments obtained with these dataset variations. The first part of the code is the name of the algorithm used, the second letter reflects the number of true clusters, C_x where x is the number of true clusters, then followed by the order of the points arrival (O: ordered one

cluster at a time then followed by the noise if any, R_2: random points from two clusters at a time followed by noise, R: completely random from all clusters and noise). The final part of the code describes the percentage of noise added as N_y where y is the percentage of noise relative to all the data.

3.1 Experimental Setup:

We compared RINO-STREAMS against two density based online clustering algorithms: TRAC-STREAMS [8] and Inc-DBSCAN [10]. Since both RINO-STREAMS and TRAC-STREAMS have similar parameters, we compared the results between these two algorithms by changing the values of three parameters while fixing the other parameters. These parameters and their values are listed in Table 1. The first parameter is the optional forgetting lifetime (τ) where we assigned a value as a percentage of the data stream length ($|X|$) with infinity for the case of no forgetting; the second parameter is the minimum sum of weights (W_{min}) which affects the minimum density threshold value ($\delta_{min} = \frac{W_{min}}{\sigma_0^2}$), and finally the maximum number of clusters allowed (K_{max}) as a percentage of the real number of clusters in the ground truth (K_G); $K_G = 5, 8, or\, 16$ depending on the data set. Some non-critical parameters were fixed, which are the initial scale value $\sigma_0^2 = 0.01$, maturity age $t_{mature} = 100$, and Chebyshev constant $\frac{1}{t^2} = 0.075$. When comparing against Inc-DBSCAN, we picked the best results for both methods and compared using different evaluation metrics discussed below, because both algorithms don't share similar parameters. To optimize the performance for Inc-DBSCAN, we tried different values for $MinPts$ from 1 to 10, and chose the best value, then we plotted the sorted k-dist graph for each run and chose the best Eps value as recommended in [9]. Table 2 shows Inc-DBSCAN's optimal values of $MinPts$ and Eps found manually for the different datasets. We chose Inc-DBSCAN because it is a leading density-based algorithm and has been widely studied in the literature.

To evaluate the results we used four evaluation metrics. Two internal metrics: Silhouette index [12] (a value close to 1 means that data points were well clustered) and the similarity matrix [12] containing the similarity values between every two data points, after ordering the data by their cluster label. The rest of the metrics are external validation metrics: error in number of clusters detected and the average centroid error relative to the ground truth. A cluster is considered to be correctly discovered if its centroid is close enough to one of the found centroids based on the Chebyshev test (7), and the difference between the two scales is less than a threshold value. In the case of Inc-DBSCAN, the centroid and scale were calculated from the points themselves. Because Inc-DBSCAN doesn't have the notion of centroid and scale, we cannot compare these values with the ground truth. Table 3 explains the x-axis index values for figure 3. All the experiments against TRAC-STREAMS showed that RINO-STREAMS performed as good or better than TRAC-STREAMS, hence, and due to paucity of space, we will only present the comparison against Inc-DBSCAN.

Table 1: Experimental Parameters

$\frac{\tau}{\lvert X\rvert}$	15%	40%	∞
W_{min}	5	20	50
$\frac{K_{max}}{K_G}$	50%	100%	200%

Table 2: Inc-DBSCAN's Optimal Parameter Values (found manually)

	MinPts	Eps
DS5	6	0.029
DS8	3	0.02
DS16	4	0.01

Table 3: The meaning of the X-axis index values for figure 3

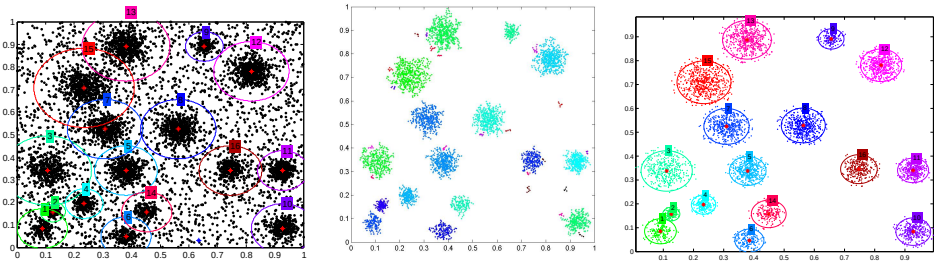
X-axis Index	1	2	3	4	5	6	7	8	9
Dataset Variations	$C_{16}ON_0$	$C_{16}ON_9$	$C_{16}ON_{18}$	$C_{16}R_2N_0$	$C_{16}R_2N_9$	$C_{16}R_2N_{18}$	$C_{16}RN_0$	$C_{16}RN_9$	$C_{16}RN_{18}$

3.2 RINO-STREAMS vs Inc-DBSCAN

In this section, we compare the quality of clustering of the proposed algorithm with that of Inc-DBSCAN. The Inc-DBSCAN optimal parameter values are listed in Table 2. For the proposed algorithm, we set the reasonable value of $K_{max} = 200\% * K_G$, $\tau = 40\% * |X|$ and $W_{min} = 20$. Figure 1 shows the clustering output for the experiment $C_{16}RN_{18}$, where figure 1(a) shows the ground-truth, figures 1(b) and 1(c) show the clustering output using Inc-DBSCAN and RINO-STREAMS respectively. Inc-DBSCAN falsely detected more clusters because it depends on the notion of density only, so outliers which are close to each other are considered valid clusters, whereas RINO-STREAMS uses a robust estimation of scale that makes it more resistant to outliers, and it also uses some quality tests (using δ_{min}) to ensure only high quality clusters are maintained. Figure 2 shows the similarity matrices of the clustering outputs for Inc-DBSCAN and RINO-STREAMS that serves to validate both cluster outputs. However Inc-DBSCAN tends to fragment some clusters (e.g.the blocks toward the bottom right corner). Figure 3 shows the results for DS16, where the x-axis corresponds to the different dataset variations (i.e. different noise percentages and order of data arrival) as explained in Table 3. DBSCAN always overestimates the actual number of clusters, and it labels small groups of noise as clusters thus finding more false clusters as the noise increases, whereas RINO-STREAMS has correctly detected the right number of clusters in most cases. The silhouette index for RINO-STREAMS is always better than Inc-DBSCAN in all the configurations, which means a higher quality clustering.

Fig. 1: DS16: Final output for Experiment $C_{16}RN_{18}$

(a) Ground Truth (b) Inc-DBSCAN (c) RINO-STREAMS (noise points detected and removed)



Note 1. Only the points that were not labeled as noise are shown. Contours denote the Chebyshev bound resulting from the estimated scales

Fig. 2: DS16: Similarity Matrix for Experiment $C_{16}RN_{18}$

(a) Inc-DBSCAN (b) RINO-STREAMS

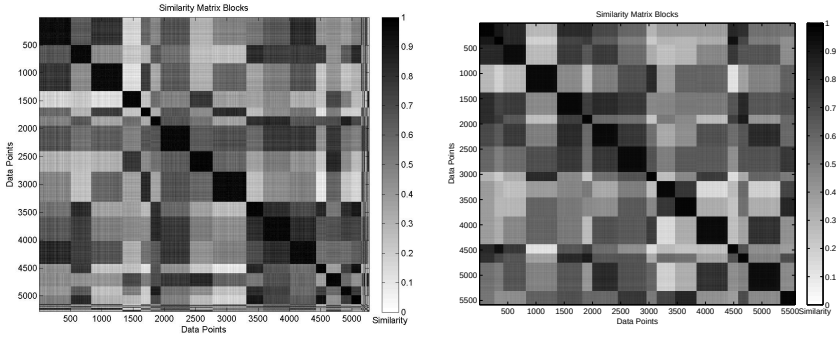
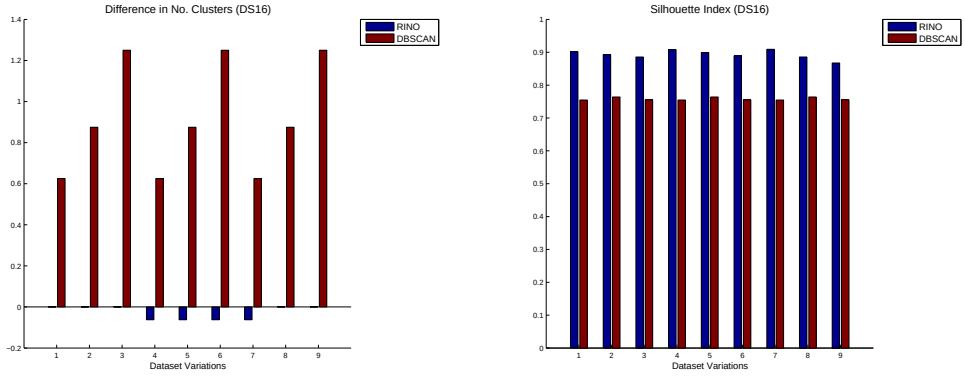


Fig. 3: DS16: RINO vs DBSCAN (See Table 3 for the x-axis legend)

(a) Error in Number of Clusters Detected ($\frac{K-K_G}{K_G}$) (b) Silhouette Coefficient



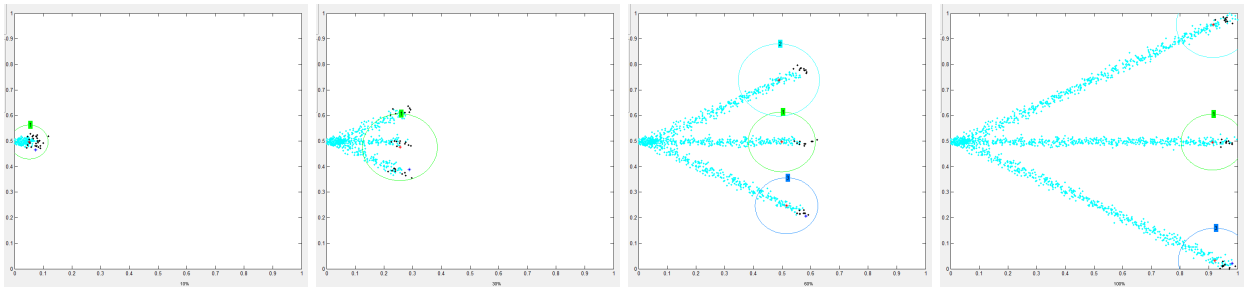
Note 2. The absence of a bar means no error

3.3 Cluster splitting and mergal

To illustrate how clusters merge and split in RINO-STREAMS, we designed two experiments where one cluster evolves into three different clusters to show cluster splitting, and one where three clusters evolve into one cluster to show cluster mergal. Figure 4 shows the cluster output evolution at five different time periods, where time is measured in terms of the number of data points that arrived relative to the data stream size ($|X|$). It can be seen that one cluster (cluster number 1) is detected at the beginning, and then, as the cluster splits, two more clusters are detected. Figure 5 illustrates the gradual mergal of three different clusters, over five different time periods, into one cluster.

Fig. 4: A cluster gradually splits into three clusters over time

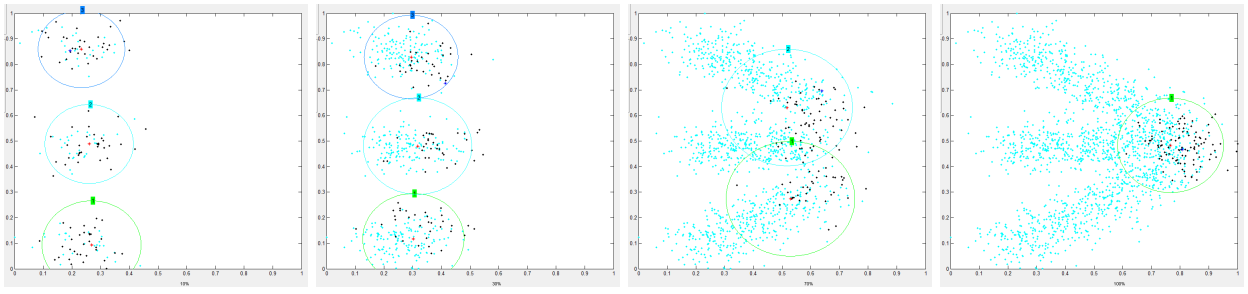
(a) At time=10% of $|X|$ (b) At time=30% of $|X|$ (c) At time=60% of $|X|$ (d) At time=100% of $|X|$



Note 3. Points in turquoise are old points (their time of arrival $> \tau$)

Fig. 5: Three clusters gradually merge into one cluster over time

(a) At time=10% of |X| (b) At time=30% of |X| (c) At time=70% of |X| (d) At time=100% of |X|



Note 4. Points in turquoise are old points (their time of arrival $> \tau$)

4 Conclusion

We presented RINO-STREAMS, a novel algorithm for mining evolving clusters from a dynamic data stream in one pass, while being able to resist and detect

the presence of outliers in the data stream. Our approach is rooted in robust statistics since it uses a robust estimation of centroids (location) as well as a robust and dynamic estimation of the cluster scales. It is also based on a robust distribution independent statistical test (Chebyshev) for the detection of outliers and for the mergal of compatible clusters, thus ensuring the robustness and compactness of the extracted evolving cluster model. The statistically based robustness, dynamic estimation of scales, and fast analytical optimization distinguish our approach from existing stream clustering methods. Our experiments validated the robustness properties of the proposed algorithm and the accuracy of its clustering model obtained in a single pass compared to two competing density based clustering algorithms, TRAC-STREAMS and Inc-DBSCAN.

Acknowledgment

This work was supported by US National Science Foundation Grant IIS-0916489

References

[1] Daniel Barbara, "Requirements for clustering data streams," ACM SIGKDD Explorations Newsletter, vol. 3, no. 2, pp. 23–27, 2002.

[2] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams," in IEEE Symposium on Foundations of Computer Science (FOCS'00), Redondo Beach, CA, 2000.

[3] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: An efficient data clustering method for large databases," in ACM SIGMOD International Conference on Management of Data, New York, NY, 1996, pp. 103–114, ACM Press.

[4] Cao, F.; Ester, M.; Qian, W. & Zhou, A. Density-based clustering over an evolving data stream with noise In 2006 SIAM Conference on Data Mining, 2006, 328-339

[5] O. Nasraoui, C. Cardona, C. Rojas, and F. Gonzalez, "Tecnostreams: Tracking evolving clusters in noisy data streams with a scalable immune system learning model," in Third IEEE International Conference on Data Mining (ICDM'03), Melbourne, FL, November 2003.

[6] Chen, Y. & Tu, L. Density-based clustering for real-time stream data KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2007, 133-142

[7] P. J. Huber, Robust Statistics, John Wiley & Sons, New York, 1981.

[8] Nasraoui, O. ,Rojas, C. Robust Clustering for Tracking Noisy Evolving Data Streams SDM, 2006

[9] Ester, M.; peter Kriegel, H.; S, J. & Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise AAAI Press, 1996, 226-231

[10] Ester, M.; Kriegel, H.-P.; Sander, J.; Wimmer, M. & Xu, X. Incremental Clustering for Mining in a Data Warehousing Environment PROC. 24TH INT. CONF. Very Large Data Bases, VLDB, 1998, 323-333

[11] Borman, S. The expectation maximization algorithm: A short tutorial. 2004

[12] Tan, P.N., Steinbach, M. , Kumar, V., Introduction to Data Mining, Addison Wesley, 2005

[13] Krishnapuram, R. and J. M. Keller (1996). The possibilistic c-means algorithm: insights and recommendations. IEEE Transactions on Fuzzy Systems 4 (3), 385-393.