Robust Clustering for Tracking Noisy Evolving Data Streams*

Olfa Nasraoui† Carlos Rojas‡

Abstract

We present a new approach for tracking evolving and noisy data streams by estimating clusters based on density, while taking into account the possibility of the presence of an unknown amount of outliers, the emergence of new patterns, and the forgetting of old patterns.

keywords: evolving data streams, robust clustering, dynamic clustering, stream clustering, scalable clustering

1 Introduction

An explosion of applications generating and analyzing *data streams* has recently added new unprecedented challenges for clustering algorithms if they are to be able to track changing clusters in noisy data streams using only the new data points because storing past data is not even an option [1, 2, 3, 4, 5]. Data streams are massive data sets that arrive with a throughput that is so high that the data can only be analyzed sequentially and in a single pass. There have been several clustering algorithms that were designed to achieve scalability [6, 7] by processing the data points in an incremental manner, or by processing the data points in small batches. However these algorithms still treat all the objects of the data set the same way without making any distinction between old data and new data. Therefore, these approaches cannot possibly handle *evolving* data, where new clusters emerge, old clusters die out, and existing clusters change. More recently, several algorithms have been proposed and discussed more specifically within the context of stream mining [1, 4, 5]. STREAMS [4] strives to find an approximate solution that is guaranteed to be no worse than a number of times the optimal. The optimal solution is based on minimizing the Sum of Squared Distances (SSQ), which is the same as the one used in K Means and LS estimates. The ordinary Least Squares (LS) method to estimate parameters is not robust because its objective function, $\sum_{j=1}^{N} d_j^2$, increases indefinitely with the residuals d_j between the j^{th} data point and the estimated fit, with N being the total number of data points in a data set. Hence, extreme outliers with arbitrarily large residuals can have an infinitely large influence on the resulting estimate. All clustering techniques that are based on LS optimization, such as K Means, BIRCH [6], and Scalable K Means [7], inherit LS's sensitivity to noise. Also, STREAMS finds a solution that approximates the entire data stream from beginning to end without any distinction between old data and newer data. Hence it has no provision for tracking *evolving* data streams. Fractal Clustering [1] defines clusters as sets of points with high self-similarity, and assigns new points in clusters in which they have minimal *fractal impact*. Fractal impact is measured in terms of the *fractal dimension*, and this can be shown to be related to the scale parameter or contamination rate. In [8], we presented a recent approach for mining evolving data streams, called TECNO-STREAMS, that is based on artificial immune systems as a type of evolutionary optimization of a criterion function. The most important desirable feature of TECNO-STREAMS was the incorporation of temporal weights that allow gradual forgetting of older portions of the data stream, and better focus on the newer data. In this paper, we present a different approach, based on analytical optimization instead of evolutionary computation, and based on a different criterion function, that explicitly optimizes the stream synopsis in the presence of unknown noise contamination rates. CluStream [5] is a recent stream clustering approach that performs *Micro-Clustering* with the new concept of *pyramidal timeframes*. This framework allows the exploration of a stream over different time windows, hence providing a better understanding of the evolution of a stream. the main idea in CluStream is to divide the clustering process into an online process that periodically stores summary statistics, and an offline process that uses only these summary statistics. Micro-clusters are an extension of BIRCH's Cluster Feature (CF) with temporal statistics, and the incremental updating of the CF is similar to BIRCH. BIRCH, in turn, solves a LS criterion because the first order statistics are nothing more than the mean centroid values. For this reason, CluStream was not designed to handle outliers. However, the attractiveness of CluStream stems from its clear and interpretable ability to track evolving clusters, because all the results can be viewed in terms of snapshots at certain time horizons, and not just as static snapshots. The approach proposed in this paper differs from CluStream in two ways: **(i)** a robust clustering process is used to resist unknown rates of outliers, **(ii)** the goal of stream clustering is to form a continuously (non-stoppable) evolving synopsis or summary of the data stream, while focusing more on the *more recent data*.

In this paper, we present a new approach for tracking evolving and noisy data streams by estimating clusters based on density, while taking into account the possibility of the presence of an unknown amount of outliers, the emergence of new patterns, and the forgetting of old patterns. Our approach, called *TRAC-STREAMS* (**T***racking* **R***obust* **A***daptive* **C***lusters in evolving data STREAMS*), is a new approach for mining noisy and evolving data streams that is based on a fast iterative optimization approach amounting to robust statistical estimation, and is free of assumptions about the noise contamination rate or scale value. This approach draws its strength from the application of *robust statistics* to the challenging *stream* environment, and to a simple continuous monitoring of the learned information in relation to the incoming data stream.

The rest of the paper is organized as follows. In Section 2, we present a new approach for mining evolving noisy data

*supported by the National Science Foundation CAREER Award IIS-0133948 to O. Nasraoui and NSF grant IIS-0431128, as well as by the National Aeronautics and Space Administration under Grant No. AISR-03-0077-0139 issued through the Office of Space Sciences.

†Department of Computer Engineering and Computer Science, Speed School of Engineering, University of Louisville, Louisville, KY 40292.

‡Department of Computer Engineering and Computer Science, Speed School of Engineering, University of Louisville, Louisville, KY 40292.

streams (TRAC-STREAMS) that achieves robustness in location and scale without any assumptions about the contamination rate or scale value. In Section 3, we present our experimental results. Finally, in Section 4, we present our conclusions.

2 Mining Evolving and Noisy Data Streams with Density Based Cluster Estimation

In a dynamic environment, the data from a data stream $\mathbf{X}_a$ are presented to the cluster model one at a time, with the cluster centroid and scale measures re-updated with each presentation. It is more convenient to think of the data index, j, as monotonically increasing with time. That is, the data points are presented in the following chronological order: $\mathbf{x}_1, \cdots, \mathbf{x}_N$. Hence after encountering J points from the data stream, a cluster is characterized by its location or center $\mathbf{c}_{i,J}$, its scale $\sigma^2_{i,J}$, and its age t_i which, to make independent of any time units, can be set to the number of points that have been streaming since the cluster's conception at $t_i = 0$. The set of clusters and their characteristic parameters define a *synopsis* [9] or good summary representation of the data stream. to summarize more basic statistics of a stream. As we will see below, because the currency of the stream is taken into account to define the influence zone around each cluster, the synopsis will also reflect a more current summary of the data stream, that will evolve with the changes in the data stream. In order to adhere to the memory requirements of a stream scenario, the maximal number of clusters is fixed to an upper bound, C_{max}, that depends on the maximum size allocated for the stream synopsis. Each candidate cluster defines an influence zone over the data space. However, since data is dynamic in nature, and has a temporal aspect, data that is more current will have higher influence compared to data that is less current. The influence zone is defined in terms of a weight function that decreases not only with distance from the data to the cluster prototype, but also with the time since the data has been presented to the cluster model. It is convenient to think of time as an additional dimension to allow the presence of evolving clusters.

Definition 1 (Adaptive Robust Weight): For the i^{th} cluster, $\mathcal{C}_i$, $i = 1, \cdots, C$, we define the robust weight of the j^{th} data point, at the moment when the total size of the data stream accumulated to J inputs: $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_j, \cdots, \mathbf{x}_J$, as

$$w_{ij,J} = w_{i,J}\left(d^2_{ij}\right) = e^{-\left(\frac{d^2_{ij}}{2\sigma^2_{i,J}} + \frac{(J-j)}{\tau}\right)} \tag{2.1}$$

where τ is an application-dependent parameter that controls the time decay rate of the contribution from old data points, and hence how much emphasis is placed on the currency of the cluster model compared to the sequence of data points encountered so far. This robust weight model is similar to the one used with an immune system inspired method proposed earlier [8] to learn a dynamic stream synopsis, however the optimization criterion function as well as the optimization method are radically different. d^2_{ij} is the distance from data point $\mathbf{x}_j$ to cluster location $\mathbf{c}_{i,J}$. $\sigma^2_{i,J}$ is a scale parameter that controls the decay rate of the weights along the spatial dimensions, and hence defines the size of an influence zone around a cluster prototype. Data samples falling far from this zone are considered outliers. At any point J in the stream sequence, the weight function of data point $\mathbf{x}_j$ in cluster $\mathcal{C}_i$ decreases geometrically with the age ($t_j = J - j$) or number of samples encountered since $\mathbf{x}_j$ was introduced. Therefore the weights will favor more current data in the learning process. Note that subsequently to each new point that is read from the stream, and assuming that the parameters do not change significantly as a result of a single point, each old weight would experience a decay as follows to enable the forgetting process:

$$w_{ij,J} = e^{\frac{-1}{\tau}} w_{ij,(J-1)} \tag{2.2}$$

At any point J in the stream (after encountering J data points), we search for the optimal *dense* cluster locations $\mathbf{c}_{i,J}$ and scale values $\sigma^2_{i,J}$,by optimizing the following criterion

$$\min_{\mathbf{c}_{i,J}, \sigma_{i,J}} \left\{ \mathcal{J}_{i,J} = \sum_{j=1}^{J} w_{ij,J} \frac{d^2_{ij}}{\sigma^2_{i,J}} - \alpha \sum_{j=1}^{J} w_{ij,J} \right\}, i = 1, \cdots, C, \tag{2.3}$$

The weight $w_{ij,J}$ can also be considered as the degree of membership of data point $\mathbf{x}_j$ in the *inlier* set or the set of good points. The first term of this objective function tries to achieve robustness by minimizing the scaled distances of the good points to the optimal cluster locations. The second term consists of the negative of a soft estimate of the *cardinality* (sum of weights) of the inlier (non-outliers) set which is approximately proportional to the estimated outlier contamination rate. Hence, by maximizing this cardinality, the objective function tries to use as many good points (inliers) as possible in the estimation process, via their high weights, so that efficiency is least compromised. Thus the combined effect is to optimize the *density*, i. e., the ratio of the total number of good points to the scale. In the first term, the distances are normalized by the scale measure for several reasons. First, this normalization counteracts the tendency of the scale to shrink towards zero. Second, unlike the absolute distance d^2_{ij}, $\frac{d^2_{ij}}{\sigma^2_{i,J}}$ is a relative measure that indicates how close a data point is to the center compared to the inlier bound. Therefore, using this normalized measure is a more sensible way to penalize the inclusion of outliers in the estimation process in a way that is independent of scale. Finally, this normalization makes the two terms of the objective function comparable in magnitude. This relieves us from the problem of estimating a value for α which otherwise would depend on the data set's contamination rate and scale. Hence, the value of α is fixed as follows:

$$\alpha = 1.$$

For the general n-dimensional case, α should be close to n, since the ratio of the first term to the second term approaches the average of a χ^2 distribution for normal distributions (a reasonable approximation, given the huge size of most data streams). Finally, we should note that d^2_{ij} should be a suitable distance measure, tailored to detect desired shapes, such as the Euclidean distance for spherical clusters. Similar to M-estimators [10], the criterion in (2.3) attempts to limit the influence of outliers by replacing the square of the residuals with a less rapidly increasing loss function of the data value. Since the objective function depends on several variables, we can use an alternating optimization technique, where in each iteration a set of variables is optimized while fixing all others. This approach forms the basis of most analytical optimization based clustering algorithms where coupled parameters are optimized alternatively, including Expectation Maximization based algorithms, DENCLUE [11], as well as most K Mean variants such as BIRCH.

THEOREM 2.1. *Optimal Incremental Center Update: Given the previous centers resulting from the past* $(J-1)$ *data points,* $\mathbf{c}_{i,J-1}$*, the new centroids that optimize (2.3) after the* J^{th} *data*

point is given by

$$\mathbf{c}_{i,J} = \frac{e^{-\frac{1}{\tau}}\mathbf{c}_{i,J-1}W_{i,J-1} + w_{iJ,J}\mathbf{x}_J}{\left(e^{-\frac{1}{\tau}}W_{i,J-1} + w_{iJ,J}\right)}. \tag{2.4}$$

where $W_{i,J-1} = \sum_{j=1}^{J-1} w_{ij,(J-1)} = W_{i,J-2} + w_{i(J-1),(J-1)}$ *is the sum of the contributions from previous data points,* $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{J-1}$.

Proof. Since the time dependency has been absorbed into the weight function, and by fixing the previous centroids, $\mathbf{c}_{i,J-1}$, scales $\sigma^2_{i,J-1}$, and weights $w_{i,j}$, the equations for centroid updates are found by solving

$$\frac{\partial \mathcal{J}_{i,J}}{\partial \mathbf{c}_{i,J}} = \frac{1}{\sigma^2_{i,J}} \quad \sum_{j=1}^{J} w_{ij,J}\frac{\partial d^2_{ij}}{\partial \mathbf{c}_{i,J}} = \mathbf{0}.$$

Each term that takes part in the computation of $\mathbf{c}_{i,J}$ is updated individually with the arrival of each new data point using the old values, and adding the contribution of the new data sample. For instance, if d^2_{ij} is the squared Euclidean distance $d^2_{ij} = \|\mathbf{x}_j - \mathbf{c}_{i,J}\|^2$, then we need to solve the following system of equations for the center components.

$$\frac{-2}{\sigma^2_{i,J}} \quad \sum_{j=1}^{J} w_{ij,J}\left(\mathbf{x}_j - \mathbf{c_{i,J}}\right) = \mathbf{0}.$$

This results in the center $\mathbf{c}_{i,J}$ given by

$$\mathbf{c}_{i,J} = \frac{\sum_{j=1}^{J} w_{ij,J}\mathbf{x}_j}{\sum_{j=1}^{J} w_{ij,J}} = \frac{\sum_{j=1}^{J-1} w_{ij,J}\mathbf{x}_j + w_{iJ,J}\mathbf{x}_J}{\sum_{j=1}^{J-1} w_{ij,J} + w_{iJ,J}}.$$

The above equation can be rewritten in the incremental form of (2.4) by substituting (2.2) into the contributions of the previous $J-1$ points in the first term of the sum in the above numerator.

THEOREM 2.2. *Optimal Incremental Scale Update: Given the previous scales resulting from the past* $(J-1)$ *data points,* $\sigma^2_{i,J-1}$, *the new scales that optimize (2.3) after the* J^{th} *data point is given by*

$$\sigma^2_{i,J} = \frac{(2+\alpha)\, e^{-\frac{1}{\tau}}\sigma^2_{i,J-1}\mathrm{WD}_{i,J-1} + w_{ij,J}d^4_{iJ}}{(2+\alpha)\left(e^{-\frac{1}{\tau}}\mathrm{WD}_{i,J-1} + w_{iJ,J}d^2_{iJ}\right)}. \tag{2.5}$$

$\mathrm{WD}_{i,J-1} = \sum_{j=1}^{J-1} w_{ij,(J-1)}d^2_{ij} = \mathrm{WD}_{i,J-2} + w_{i(J-1),(J-1)}d^2_{i(J-1)}$ *is the sum of the contributions from previous data points,* $\mathbf{x}_1, \cdots, \mathbf{x}_{J-1}$.

Proof. By fixing the previous values of the centroids, $\mathbf{c}_{i,J-1}$, scales $\sigma^2_{i,J-1}$ and weights $w_{i,j}$, the equations for scale updates are found by solving $\frac{\partial \mathcal{J}_{i,J}}{\partial \sigma^2_{i,J}} = 0$ and substituting $\frac{\partial w_{ij,J}}{\partial \sigma^2_{i,J}} = \frac{1}{2\sigma^2_{i,J}}w_{ij,J}$. The scale parameter of the ith cluster is given by $\sigma^2_{i,J} = \frac{1}{(2+\alpha)}\frac{\sum_{j=1}^{J} w_{ij,J}d^4_{ij}}{\sum_{j=1}^{J} w_{ij,J}d^2_{ij}}$. the rest of the proof proceeds similarly to the proof for the incremental centers.

Therefore, the algorithm for incrementally adjusting the optimal cluster locations and scales will consist of updates of the prototype parameters, followed by updates of the scale parameter and the weights in an iterative fashion, with the arrival of each new data point in the stream. Note that only the cumulative statistics need to be stored for each cluster at any given instant. Hence the synopsis consists of a summary of all the clusters, where each cluster is represented by the tuple consisting of $\mathbf{c}_{i,J}$, $\sigma^2_{i,J}$, $W_{i,J}$, and $\mathrm{WD}_{i,J}$.

2.1 Learning New data points and Relation to Outlier Detection

Definition 2 (Potential Outlier): A *potential outlier* is a data point that fails the outlyingness test for the entire cluster model. The outlier is termed *potential* because, initially, it may either be an outlier or a new emerging pattern. It is only through the continuous learning process that lies ahead, that the fate of this outlier will be decided. If it is indeed a true outlier, then it will form no mature clusters in the cluster model. Several upper tail bounds exist in statistics that bound the total probability that some random variable is in the tail of the distribution, i.e., far from the mean. Markov bounds apply to any non-negative random variable, and hence do not depend on any knowledge of the distribution. However, a *tighter bound can be obtained using Chebyshev bounds* if a reliable estimate of scale is available. Again, *no assumptions are made about the distribution* of the data, other than scale. Because TRAC-STREAMS provides robust scale estimates, we will use Chebyshev bounds to test whether a data point is an outlier.

Chebyshev Bounds: The Chebyshev bound for a random variable X with standard deviation σ is:

$$Pr\left\{|X-\mu| \geq t\sigma\right\} \leq \frac{1}{t^2} \tag{2.6}$$

Testing a Data Point or a New Cluster for Outlyingness with respect to cluster $\mathcal{C}_i$ using Chebyshev Bound with Significance Probability $1/t^2$: (2.6) can be rearranged in the form
$Pr\left\{|X-\mu|^2 \geq t^2\sigma^2\right\} \leq \frac{1}{t^2}$, or equivalently
$Pr\left\{e^{\frac{-|X-\mu|^2}{2\sigma^2}} \leq e^{-t^2/2}\right\} \leq \frac{1}{t^2}$
This results in the following test for outlier detection:

IF $w_{ij,J} < e^{\left(-t^2/2\right)}$ THEN
  $\mathbf{x}_j$ is an outlier with respect to cluster $\mathcal{C}_i$

The same test is used in the algorithm to decide when a new cluster is created, since points from new clusters can be considered as outliers with respect to old clusters. Note that the quantity $t^2\sigma^2$ is referred as the *Chebyshev Bound* from now on, and it is also used to plot the contours of the learned clusters later on, since it is considered as the distance from the centers that includes all the inliers with probability $1 - \frac{1}{t^2}$. We also use the same test to estimate the cardinality N_i for each cluster centroid as the number of points from the input data stream that fail the outlier test.

Testing the compatibility of clusters $\mathcal{C}_i$ and $\mathcal{C}_k$ with scales $\sigma^2_{i,J}$ and $\sigma^2_{k,J}$ using Mutual Chebyshev Bounds with Significance Probability $1/t^2$: Given the distance between these two clusters, d^2_{ik}, if the clusters are compatible, then (2.6) can be rearranged in the form
$Pr\left\{e^{\frac{-d^2_{ik}}{2\sigma^2_i}} \leq e^{-t^2/2}\right\} \leq \frac{1}{t^2}$ and $Pr\left\{e^{\frac{-d^2_{ik}}{2\sigma^2_k}} \leq e^{-t^2/2}\right\} \leq \frac{1}{t^2}$
This results in the following test for testing cluster compatibility:

IF $\left(dist\left(\mathcal{C}_i, \mathcal{C}_k\right) < t^2\sigma^2_i \text{ AND } dist\left(\mathcal{C}_i, \mathcal{C}_k\right) < t^2\sigma^2_k\right)$ THEN
  Merge $\mathcal{C}_i$ and $\mathcal{C}_k$

The cluster with higher density defined in (2.8), is the one that remains. Its scale is inherited, while the its new center becomes

$$\mathbf{c}_{new,J} = \frac{\mathbf{c}_{i,J}W_{i,J} + \mathbf{c}_{k,J}W_{k,J}}{W_{i,J} + W_{k,J}}. \tag{2.7}$$

Cluster Testing Based on Density The density of the i^{th} cluster after presenting J data points from the stream is defined as follows:

$$\delta_i = \frac{\sum_{j=1}^{J} w_{ij,J}}{\sigma^2_{i,J}}. \tag{2.8}$$

Clusters (i) with low density (δ_i) or zero cardinality (N_i) are eliminated as follows:

IF $\left(\delta_{(i)} < \delta_{min}\right)$ OR $N_i = 0$ THEN
$\mathcal{B} \leftarrow \mathcal{B} - \mathcal{C}_{(i)}$
}

Finally we give the steps of the TRAC-STREAMS algorithm below:

TRAC-STREAMS:
Tracking Robust Adaptive Clusters in evolving data STREAMS

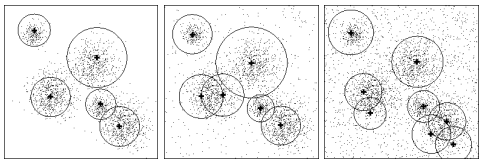
Fix the maximal number of cluster prototypes, C_{max}*;*
Initialize centroids $\mathbf{c}_{i,J} = \mathbf{x}_i$*,* $i = 1, \cdots, C_{max}$*;*
Initialize scale $\sigma_{i,J} = \sigma_0$*,* $i = 1, \cdots, C_{max}$*;*
Initialize age $t_i = 0$*,* $i = 1, \cdots, C_{max}$*;*
Initialize the sums $W_{i,J} = WD_{i,J} = 0$*,* $i = 1, \cdots, C_{max}$*;*
Initialize $C = C_{max}$*;*
Let cluster model $\mathcal{B} = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \cdots \cup \mathcal{C}_C$*, and cluster representatives after* J *data points, be* $\mathcal{C}_i = (\mathbf{c}_{i,J}, \sigma_{i,J}, t_i, W_{i,J}, WD_{i,J})$*;*
FOR $J = 1$ **TO** N **DO** { *// single pass over the data stream* $\mathbf{x}_J$
  FOR $i = 1$ **TO** C **DO**
    Compute distance, d^2_{iJ}*, and robust weight,* $w_{iJ,J}$*;*
  IF $C < C_{max}$ *AND* $\mathbf{x}_J$ *is an outlier in all clusters* $\mathcal{C}_i$ *THEN* {
    Create new cluster $\mathcal{C}_k$*:* $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{C}_k$ *with* $k = C + 1$*;*
    $\mathbf{c}_k = \mathbf{x}_J$*;*
    $\sigma_k = \sigma_0$*;*
    Initialize $t_k = W_{k,J} = WD_{k,J} = 0$*;*
    $C = C + 1$
  }
  FOR $i = 1$ **TO** C **DO** {
    Update $\sigma^2_{i,J}$ *using (2.5);*
    Update $\mathbf{c}_{i,J}$ *using (2.4);*
    Update $W_{i,J} = W_{i,J-1} + w_{ij,J}$ *;*
    Update $WD_{i,J} = WD_{i,J-1} + w_{ij,J} d^2_{iJ}$ *;*
    Update age $t_i = t_i + 1$ *;*
  }
  FOR $i = 1$ **TO** C **DO**
    IF $\left(\delta_{(i)} < \delta_{min}\right)$ *OR* $N_i = 0$ *THEN* {
      $\mathcal{B} \leftarrow \mathcal{B} - \mathcal{C}_i$*;*
      $C = C - 1$*;*
    }
  Test Clusters for Compatibility and Merge Compatible Clusters;
}

2.2 Computational Complexity

TRAC-STREAMS necessitates the iterative computations of distances and weights for each data vector, followed by the center and scale parameter. These are all linear in the number of data vectors. Hence the computational complexity of TRAC-STREAMS is $\mathcal{O}(N)$. At any point in time throughout the stream sequence, only the most recent data point $\mathbf{x}_J$ is needed to incrementally update all the cluster components. Therefore, the memory requirements are obviously linear with respect to the maximal number of clusters, C_{max}, which is a negligible fraction of the size of the data set.

3 TRAC-STREAMS Experimental Results

We applied TRAC-STREAMS in a single pass for several clean and noisy data sets (all 256 X 256 binary images), with $\sigma_0 = 100$, significance level $1/t^2 = 0.075$ for all Chebyshev bounds. To make the order of cluster presentations easy to follow, the data points are presented in row major format, i.e. in increasing order of their y-values. Hence newer clusters are towards the bottom of the images. Fig. 1 illustrates the results of TRAC-STREAMS for a data set with varying noise contamination rates, showing robustness in both location and scale, achieved in a single pass over the data set, with the number of clusters limited to a modest $C_{max} = 10$ clusters. Fig. 2 illustrates the effect of the upper limit on the number of clusters C_{max} on TRAC-STREAMS for a noisy data set (65% noise), showing stability and robustness in both location and scale. When C_{max} is less than 8 for this data set, one or more clusters will be missed depending on their density and age. Fig. 3 illustrates the effect of the time constant τ that affects the speed of forgetting older clusters (hence, the ability to track evolving clusters), showing robustness and a continuous adaptation to new clusters. When τ is less than 3000, one or more older clusters will be forgotten by the time all the data points have been presented. As expected, the smaller the value of τ, the faster is the forgetting. This desirable property illustrates how the proposed approach can be tuned to provide slow or fast evolvability with the input data stream. Again, this time dependent adaptation is optional since it is easy to set $\tau = \infty$ (i.e., infinite memory, hence $e^{-1/\tau} = 1$) if *all* clusters are to be maintained, of course subject to the maximal limit C_{max}. Finally Fig. 4 illustrates the performance for noisy data sets with varying cluster configurations, densities, sizes, and shapes. Even though we do not discuss the details of how TRAC-STREAMS can be generalized to approximate *arbitrary* shapes, showing robustness and reasonable approximation. We repeat the same validation experiment with the BIRCH data set consisting of 100,000 data points presented as a stream in the order of columns. Fig. 5 shows the learned synopsis (cluster contours corresponding to a Chebyshev inlier bound/diameter of $t^2\sigma^2$, learned at several points in the stream, superimposed on the data stream seen so far. Notice how the learned synopsis closely approximates the last $\tau = 5000$ data points, showing ability to evolve with new concepts, and to forget old concepts. Finally, to show the linear scalability, we plot the total number of distance computations (the most expensive computation/step of the algorithm, independently of time unit) in Fig. 6.

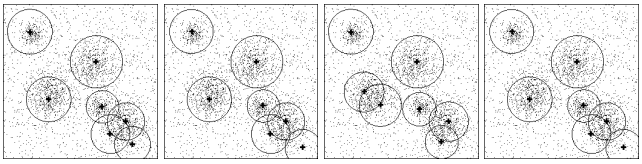


(a) (b) (c)

Figure 1. Results of TRAC-STREAMS for varying noise contamination rates ($C_{max} = 10$): (a) 0% noise, (b) 15% noise, (c) 65% noise

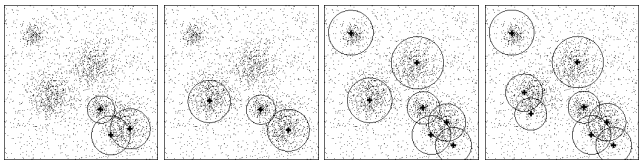
4 Conclusions

We presented a new approach for tracking evolving and noisy data streams by estimating clusters based on density, while taking into account the possibility of the presence of an unknown amount of outliers, the emergence of new patterns, and the forgetting of old patterns. As expected, the smaller the value of the application dependent time constant τ, the faster the forgetting of old clusters. Hence, TRAC-STREAMS can be tuned



(a) (b) (c) (d)

Figure 2. Results of TRAC-STREAMS for varying Maximal number of clusters allowed C_{max} ($\tau = 4000$): (a) $C_{max} = 10$, (b) $C_{max} = 20$, (c) $C_{max} = 30$, (d) $C_{max} = 40$



(a) (b) (c) (d)

Figure 3. Results of TRAC-STREAMS with varying time constants τ that affects the speed of forgetting older clusters (ability to track evolving clusters) ($C_{max} = 10$): a) $\tau = 1000$, (b) $\tau = 2000$, (c) $\tau = 4000$, (d) $\tau = \infty$ (infinite memory, hence $e^{-1/\tau} = 1$)

to provide slow or fast *evolvability* with the input data stream. Even though we did not discuss the fate of older clusters, it is clear that a policy, such as delegating older clusters to secondary storage (*archiving*) before they die and wither away, is easy to implement. The maximal limit on the number of clusters, C_{max}, can be set according to the memory constraints, and even if it is too small, the densest clusters will be estimated correctly.

The approach proposed in this paper differs from existing methods in two ways: **(i)** a robust clustering process is used to resist unknown rates of outliers, **(ii)** the goal of stream clustering is to form a continuously evolving synopsis or summary of the data stream, while focusing more on the *more recent data*. This is essentially a different framework from all existing approaches, because the stream mining is to run without any stoppages or reconfigurations. This framework is useful in cases like monitoring live data such as network activity data, newsfeeds and Web clickstreams, because in most of these applications, the goal is to use the mined patterns for a subsequent stage (e.g. intrusion detection, information filtering, or Web personalization) that depends on the accuracy of the summary, with an emphasis on the most recent state of the system.

References

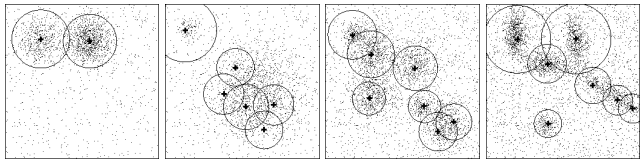
[1] Daniel Barbara, "Requirements for clustering data streams," *ACM SIGKDD Explorations Newsletter*, vol. 3, no. 2, pp. 23–27, 2002.

[2] S. Babu and J. Widom, "Continuous queries over data streams," in *SIGMOD Record'01*, 2001, pp. 109–120.

[3] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang, "Multi-dimensional regression analysis of time-series data streams," in *2002 Int. Conf. on Very Large Data Bases (VLDB'02)*, Hong Kong, China, 2002.

[4] 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.

[5] C. Aggarwal, J. Han, J. Wang, and P. Yu, "A framework for clustering evolving data streams," in *29th VLDB conference*, 2003.



(a) (b) (c) (d)

Figure 4. Results of TRAC-STREAMS for different data sets ($C_{max} = 10, \tau = \infty$): (a) 2 clusters with different density, (b) 2 clusters with different size, (c) 6 clusters with different sizes and densities, (d) data with a mixture of spherical and elongated clusters (note how different shapes are approximated by several spherical clusters)

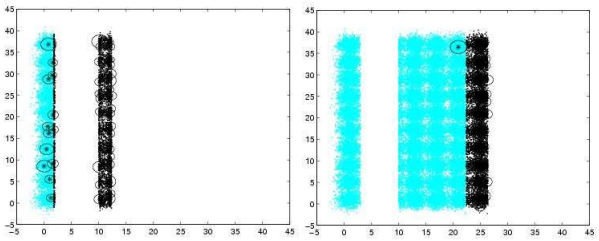


Figure 5. BIRCH data - Learned Synopsis in one pass at 13,000 and 51,000 points respectively. Only the data points that have been seen so far are shown in light color. Since $\tau = 5000$ only the last 10,000 data points (shown in black) are summarized by a synopsis limited not to exceed $C_{max} = 25$ clusters.

[6] 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.

[7] P. Bradley, U. Fayyad, and C. Reina, "Scaling clustering algorithms to large databases," in *Proceedings of the 4th international conf. on Knowledge Discovery and Data Mining (KDD98)*, 1998.

[8] O. Nasraoui, C. Cardona, C. Rojas, and F. Gonzalez, "Tecno-streams: 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.

[9] M. Henzinger, P. Raghavan, and S. Rajagopalan, "Computing on data streams," 1998.

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

[11] Alexander Hinneburg and Daniel A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *Knowledge Discovery and Data Mining*, 1998, pp. 58–65.

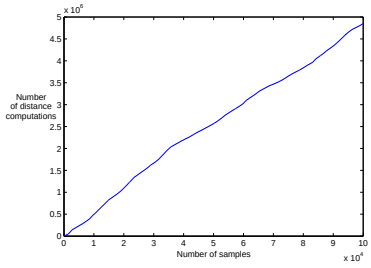


Figure 6. Linearity of the algorithm in terms of total number of distance computations (BIRCH data set)