Sketching Streaming Histogram Elements Using Multiple Weighted Factors

Quang-Huy Duong*
Norwegian University of Science and Technology, Norway
huydqyb@gmail.com

Heri Ramampiaro
Norwegian University of Science and Technology, Norway
heri@ntnu.no

Kjetil Nørvåg
Norwegian University of Science and Technology, Norway
noervaag@ntnu.no

ABSTRACT

We propose a novel sketching approach for streaming data that, even with limited computing resources, enables processing high volume and high velocity data efficiently. Our approach accounts for the fact that a stream of data is generally dynamic, with the underlying distribution possibly changing all the time. Specifically, we propose a hashing (sketching) technique that is able to automatically estimate a histogram from a stream of data by using a model with adaptive coefficients. Such a model is necessary to enable the preservation of histogram similarities, following the varying weight/importance of the generated histograms. To address the dynamic properties of data streams, we develop a novel algorithm that can sketch the histograms from a data stream using multiple weighted factors. The results from our extensive experiments on both synthetic and real-world datasets show the effectiveness and the efficiency of the proposed method.

CCS CONCEPTS

• **Information systems → Data mining**; • **Theory of computation → Sketching and sampling**;

KEYWORDS

Sketch; Stream; Histogram; Weighted Factors; Concept Drift

ACM Reference Format:
Quang-Huy Duong, Heri Ramampiaro, and Kjetil Nørvåg. 2019. Sketching Streaming Histogram Elements Using Multiple Weighted Factors. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Storing and processing high volume and high velocity streaming data are often infeasible due to the limitation of memory and computation infrastructure. Normally, data in a stream can be accessed only once, thus making efficient processing of streams a challenging but crucial task. A possible solution is sketching of the streaming data. Sketching is an effective approximation method that maps a stream into a maintainable form, while still retaining the characteristics of the stream with high accuracy. Development of efficient sketching techniques has attracted much research during the past decades [24, 25, 27]. In addition to being used to visualize the statistical information of the data, *histogram* estimation is among commonly used techniques to sketch the underlying distribution of data [21]. Still, an important limitation of existing histogram-based methods is that most of these methods were developed with the assumption that histograms are drawn from a non-changing or static data distribution [10, 22]. In practice, streaming data are inherently dynamic and evolve over time, which in turn result in dynamic changes in the underlying data distribution, i.e., concept drift [8].

To cope with the issues related to concept drift, the concept of forgetting factor has been introduced to determine the significance of data according to the time the data occurs in a stream [7, 20, 21, 29]. However, most of the current work, e.g., [21] and [29], assume that a forgetting factor is constant at every point of observation and in the whole process, which is too restrictive. This is because in many real-world applications, we cannot consider the same factor for every past data at a particular single observation point. To illustrate, assume we would like to analyse the behavior of tourists related to point-of-interest (POI) check-ins in several cities at the same time. Here, there would be several underlying aspects that might affect the user check-ins. In addition to types and places of the POIs, changes in the weather conditions, seasons and times of the day could influence changes in the user behaviors. For instance, a sightseeing place would normally get more visitors during the summer than winter, but at the same time, they could be impacted by (possibly sudden) changes of the weather conditions. In conclusion, analyzing the check-in behaviors without considering the changes with respect to the dynamic properties and time-dependent changes would not be enough to cover the whole picture. This example further motivates the need for a model that is capable of dynamically adapting to changes in a data stream. We refer such a model to as an *evolving model*, i.e., a model that can automatically tune its coefficients to follow any occurring changes. Here, assuming a histogram to be formed by a matrix, coefficients are the factors specifying the weight of the values of such a histogram matrix.

To address the challenges described above, we propose an efficient algorithm called "**E**nsemble **R**andomization **S**ketch of **S**treams" (eRSS). The eRSS utilizes an evolving model that automatically updates its coefficients to minimize a proposed objective cost of timestamps, with dynamic constraints. To the best of our knowledge, this is the first work that studies a sketching technique for data streams using a dynamically adapting model, which sketches a

*Corresponding author.

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

stream of data, while taking into account the time-sensitive changes in the stream using different adaptive weighted factors for different groups of data, at every observation point. Hence, the proposed model is specifically useful for histogram estimation for streaming data. In particular, we apply weighted minwise hashing on estimated count-min histograms to sketch the stream.

To explain the basic idea of our approach, Figure 1 shows a block diagram of the important steps of the method. In this figure, referring to our earlier example, we consider the data stream to be a stream of histograms, each of which has elements that are generated from the check-ins at a POI. Hence, since we have several POIs, at a given time, $t-1$, we need to generate several histograms as well. Then, at a time t, to handle the increasing number of check-ins and any occurring changes, we apply an ensemble-based model consisting of several single sketching processes in order to estimate the final sketch of s elements (sc_i). Here, we estimate an adaptive weighted count-min histogram for each POI, and for each histogram, we generate a sketch by utilizing a sampling process on the weighted histogram elements.

An important property of our method is that although it provides an estimated sketch resulting in a compact representation of the stream, as we will explain later, it is able to preserve the characteristics of the data distribution that underlies a stream. In addition, our model is able to maintain histogram similarities, which is not only useful but needed for handling concept drifts. To show its efficiency and effectiveness we fully evaluate our approach with extensive experiments on both synthetic and real-life datasets. For the sake of reproducibility, the source code and data used in our experiments are made publicly available at https://bitbucket.org/duonghuy/sketch.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 briefly introduces the background and formulation of sketching a stream of data. Section 4 presents the proposed model and the adaptive estimation method. Section 5 describes and discusses the results from the experimental evaluations. Finally, Section 6 concludes the paper and outlines possible topics for future work.

2 RELATED WORK

Sketching is a hashing technique [4] that has been extensively studied and has been widely applied to maintain a dataset in an approximating compact representation. Numerous algorithms for sketches have been proposed, including Count-Min sketch (CM) [5], Count-Min sketch with conservative update (CM-CU) [9], and Augmented sketch [18]. However, due to skewness and dynamic property of data in a stream, these sketches are inefficient when applied on real data streams [6, 31], since they often lead to a need to estimate the frequency while considering the weight of data in a stream.

In order to preserve the characteristics of data, a sketch needs to simulate the data distributions of a given dataset into a form that is closest possible to the real distributions. Several algorithms have been proposed to sketch a given dataset [3, 11, 12, 19]. One of the most well-known algorithms is locality-sensitive hashing (LSH) [11], which is a data-independent method that can guarantee the collision probability between similar data points. Examples of LSH-based methods are SimHash [19] and MinHash [3] which differ on the ways they compute similarities. SimHash [19] uses cosine distance to measure the similarity between data files, while in MinHash (or minwise hashing) [3], Jaccard similarity is used to estimate the similarity between two sets. MinHash-based algorithms are generally efficient and are more efficient than SimHash when using cosine similarity, but one of the drawbacks is that they consume much space. Further, minwise hashing uses large number of permutations on the data, which, in turn, degrades the performance of the algorithms since minwise hashing is normally adopted in the context of binary or high-dimensional data. To cope with the costly processing of data in minwise hashing, Weighted Minwise Hashing (WMH) [12], inspired the idea of using densification and one permutation [14]. WMH uses a “rotation" scheme for densifying the estimated sparse sketches of one permutation hashing that assigns new values to all the empty buckets. As a result, WMH greatly reduces computation cost compared to MinHash-based hashing. The WMH is much simpler, significantly faster, and more memory efficient than the original ones, especially in very sparse datasets [4].

Histograms has long been seen as one of the most important data statistical tools for providing information about a data distribution [17]. There exist various algorithms for estimation of histograms generated from streaming data. A few of these algorithms consider dynamic forgetting factor, including the most recent one, HistoSketch [29]. HistoSketch adopts Weighted Sampling method [12, 13] and Hamming distance, which is comparable with our approach. However, every time a new data arrives in the stream, to cope with concept drift, HistoSketch uses a constant decay factor to decrease the importance of outdated data. Several adaptive estimation approaches have been proposed to efficiently detect concept drifts [1, 21]. An important difference with our approach is that all of these algorithms compute a decay factor at every observation. This means that they decrease the importance of all outdated data by that factor, and use the same factor on all data. As mentioned above, the mechanism of applying the same forgetting factor to all data at different timestamps in the stream is too restrictive and could be impractical. To be effective, it is necessary that a sketching approach considers a model that can automatically adapt its decay factor following the changes in the stream. Moreover, sketching the histogram of a stream is an approximate solution and is based on randomization processes and randomization hashing functions. Single randomization is usually a weak process, which in general yields poor performance [20]. Our hypothesis is that combining several weak single processes to form an ensemble component can obtain better performance. Particularly, each single component in the ensemble model estimates a sample of the histogram of the data in a stream, in a sampled sketch. Then, the final sketch is determined using a scoring function on these estimated sketches in a final component.

3 PRELIMINARIES

In the following, we present fundamental definitions about consistent weighted sampling [12, 13, 15] that we use in our solution on sketching of a streaming data containing concept drifts. Given two data vectors (weighted set) of k elements $X, Y \in \mathbb{R}^k$,

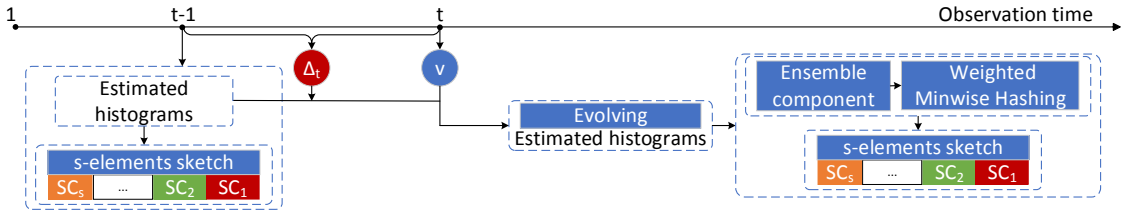


Figure 1: Overview of the proposed method. The processing is performed for each histogram of the stream.

Definition 3.1 (Uniformity [12, 13, 15]). Let (i, S_i) be a sample of X, where $1 \leq i \leq k$ and $0 \leq S_i \leq X_i$. We call a process uniformity sampling if its sample is distributed uniformly over the pairs.

Definition 3.2 (Consistency [12, 13, 15]). Assume that X dominates Y, i.e., $Y_i \leq X_i$, for all i and $1 \leq i \leq k$. If a sample (i, S_i) is sampled from X and satisfies $S_i \leq Y_i$, then (i, S_i) is also sampled from Y.

Definition 3.3 (Consistent weighted sampling [12, 13, 15]). Consistent weighted sampling is a sampling process that satisfies both the uniformity and consistency properties.

The min-max similarity between X, Y is defined by [12, 13]:

$$SIM_{min-max}(X, Y) = \frac{\sum \min(X_i, Y_i)}{\sum \max(X_i, Y_i)}$$

Let $SE(X)$ and $SE(Y)$ denote the samples of data vectors X and Y, respectively, using a consistent weighted sampling schema method. The collision probability between the two vectors X, Y is exactly its min-max similarity [12, 13]:

$$Pr[SE(X) = SE(Y)] = SIM_{min-max}(X, Y).$$

Given a data stream S, let t be the current time point, v_t be the current incoming item at time point (order) t with timestamp $\mathcal{T}_t$, and $V_t = (v_1, v_2, \ldots, v_t)$ be the current sub-stream. The task of sketching a stream S at a point of time t is to maintain a parameter sketch SK (s elements) so that SK is a compact representation of the current sub-stream V_t, and SK preserves the interesting (similarity) measure of V_t. In particular, in this paper, the data stream is a stream of histogram elements of many POIs (point of interest). We estimate an adaptive weighted count-min histogram for each POI in the stream. Then, a sample is generated by utilizing a consistent weighted sampling schema [12, 13, 15] on each weighted histogram. To get s samples, we repeat the process s times using an independent set of random numbers, and the similarity of two samples is preserved based on the theoretical guarantee of consistent weighted sampling.

4 ENSEMBLE HISTOGRAM SKETCHING

This section presents the proposed model and the adaptive estimation method to sketch the streaming data.

4.1 Histogram

For a data stream, a histogram is a data summarization method, with which data are hashed into a set of buckets. Each bucket maintains the frequency of the hashed data. In order to estimate the histogram of a data stream, we adopt the count-min sketching method originally proposed by Cormode and Muthukrishnan [5] because of its efficiency and simplicity. A histogram of a data stream using a count-min sketch is maintained by a matrix $\Theta \in \mathbb{R}^{n \times m}$, where n is the number of random hash functions h_i, $i = 1, 2, \ldots, n$, and m is the number of ranges such that each hash function h_k will map an incoming data v of the stream to each range from 1 to m. When a new data v arrives in the stream, n hash functions h_i, $i = 1, 2, \ldots, n$, will hash the value v to a corresponding range (a column of m columns of the matrix Θ), then the value of the corresponding cell of the matrix Θ is increased by 1, i.e., $\Theta(i, h_i(v)) = \Theta(i, h_i(v))+1$.

4.2 Weighted Sampling

This subsection presents how the proposed method weighs the importance of histogram element frequencies in a stream. The weighted cumulative frequencies of histogram elements are used to estimate a compact sketch of that stream.

Evolving data and weight of histogram: Fading factor, which is used to weigh the elements within a streaming context, was proposed to reduce the impact of noise in streaming data and weigh the importance of data in the stream [20, 21, 29]. In these works, at each single observation time, the weight of all the old data is divided by the same constant factor. In a data stream, however, data evolves over time, and the evolvement of data might be different with respect to different group of elements. Some previously generated elements or groups of elements might have more importance than other elements. Therefore, it is crucial not to treat the importance of all past data equally by the dividing with the same constant factor. On top of this, a data point could be correlated with other data points, which makes this even more important.

Moreover, there might be strong correlation between features of data during the time they are evolving, such that a feature evolves, this might affect or impact the evolvement of other features, as well. Intuitively, the idea of the modeling procedure is as follows. At each observation time, groups of elements might have different weights and there might exist correlations between some features. We need to model the evolvement data features by estimating the variables that determine the weights of the histogram elements. These variables are unknown in advance, and they are continuously updated and estimated from observed data. The model then automatically performs the variable selection to minimize a loss function, while considering the variance and sparsity factors of the variables.

Here, to address the correlation of data, we utilize the idea of the elastic net [32] to simulate the evolvement of data in our proposed method. In our model, we define an objective function at an observation time t, having a form of an elastic net, as follows:

$$\mathcal{L}(\Theta_i) = \Omega_1(\Theta_i^t) + \Omega_2(\Theta_i^t) + \Delta(\Theta_i^t, \Theta_i^{t-1}), \quad (1)$$

where Θ_i are our predicting estimators. The meaning of penalties in the objective function are the same as in elastic net regularization. The first term, a penalty term, $\Omega_1(\Theta_i^t) = \alpha\beta\Lambda(\Theta_i^t)$ is to control sparsity in the single solution. To estimate the sparsity, we choose $\Lambda(\Theta_i^t)$ as ℓ_1 column norm of vector at row i of the matrix Θ at time t, $\Lambda(\Theta_i^t) = \| \Theta_i^t \|_1$, because it minimizes the less squares loss function [23]. However, using only ℓ_1 norm tends to select one variable from a group of highly correlated variables and set less important coefficients to zero. To overcome this drawback, the second term $\Omega_2(\Theta_i^t)$ is used to enforce the hierarchy constraint, and it is set to a form of 2-norm to force the coefficients close to the average value rather than zero, $\Omega_2(\Theta_i^t) = (1-\alpha)\beta \| \Theta_i^t \|_2^2$. The parameter α, β in the first two penalties are tuning parameters, $\beta \in [0, \infty)$, and $\alpha \in [0, 1]$. Here, α controls the relative weight on the first two penalties and is a relaxation parameter; while parameter β controls the selection of parameters.

The last term $\Delta(\Theta_i^t, \Theta_i^{t-1})$ is a penalty to minimize the deviation across different timestamps. The last term in Eq. 1 controls this deviation, on which we target the similarity of Θ at different observation times t and (t-1). The deviation is estimated by a robust ℓ_2 penalty that is useful in concept drift detection. Δ has a form of $\Delta(\cdot) = \| \cdot \|_2$. The objective function in Eq. 1 now can be rewritten as:

$$\mathcal{L}(\Theta_i) = \alpha\beta \| \Theta_i^t \|_1 + (1-\alpha)\beta \| \Theta_i^t \|_2^2 + \| (\Theta_i^t, \Theta_i^{t-1}) \|_2^2 \quad (2)$$

Our evolving model minimizes objective function $\mathcal{L}(\Theta_i)$ to obtain its coefficients at observation time t. Figure 2 shows the weighting process of histogram elements of the current technique and our proposed method.

Time constraints: Studying current approaches, it seems that most of them assume that the observations are captured sequentially in time. They only consider the order of observations, and do not include the time interval in the study but consider the timespan between observations to be constant. This means that $\Delta(t) = \mathcal{T}_t - \mathcal{T}_{t-1}$ is constant, i.e., intervals between sampling are discretely framed. However, such an assumption usually does not hold in practice. To cope with this, we re-define the third term of $\mathcal{L}(\Theta_i)$ when considering the real time of observations of data sampling. We let $\mathcal{T}^t$ be the time between two consecutive observations, i.e., the timespan between current and previous time within which the data are sampled. The intuition is that we split the frame into $\mathcal{T}^t$ equal segments. Further, we assume that there are virtual data sampling observations at each splitting point. Therefore, the deviation penalty is cumulative deviation via a solution path including $\mathcal{T}^t$ penalties:

$$\Omega_3(\Theta_i^t, \Theta_i^{t-1}) = f(\mathcal{T}^t) \| (\frac{\Theta_i^t - \Theta_i^{t-1}}{\mathcal{T}^t}) \|_2^2,$$

where $f(.)$ is an interest function of time. The objective function corresponding to each row of histogram matrix Θ now becomes:

$$\mathcal{L}(\Theta_i) = \alpha\beta \| \Theta_i^t \|_1 + (1-\alpha)\beta \| \Theta_i^t \|_2^2 + f(\mathcal{T}^t) \| (\frac{\Theta_i^t - \Theta_i^{t-1}}{\mathcal{T}^t}) \|_2^2$$

Given a histogram matrix Θ of a data stream, the value of Θ at observation time (t-1) is denoted as Θ^{t-1}. Given that Θ evolves over time, at time point t, Θ becomes Θ^t such that the objective function $\mathcal{L}(\Theta_i)$ is minimum. Because Θ^{t-1} is known, in the following we use notation A regrading Θ^{t-1} to indicate that it is constant. Our problem is an optimization problem to minimize the objective function as follows:

$$\begin{aligned}\mathcal{L}(Z, t) = \alpha\beta \| Z^t \|_1 + (1-\alpha)\beta \| Z^t \|_2^2 \\ + f(\mathcal{T}^t) \| (\frac{Z^t - A_i}{\mathcal{T}^t}) \|_2^2,\end{aligned} \quad (3)$$

where Z is used to denote Θ_i. For the sake of brevity, in the rest of the paper, we remove subscript of A and drop the argument t. The proposed objective function in (3) has a form of an elastic net regularization [32]. Numerous works have been extensively studied elastic net regularization in many applications, such as machine learning and neural networks, but to the best of our knowledge, using elastic net like in the model in Eq. 3 for sketching a data stream with concept drift is new.

Optimization solver: To minimize the objective function in Eq. 3, we employ the Alternating Direction Method of Multipliers (ADMM) [2] by controlling each penalty separately. Firstly, the ADMM breaks the objective function in Eq. 3 into three smaller parts, such that each part is easier to handle. Thereafter, it solves each part separately while considering the rest as constant. Finally, the global convex optimization problem is solved when all parts are handled.

We use a consensus variable $U = \{U_1, U_2\}$ and scaled dual form variable $V = \{V_1, V_2\}$. We form the augmented Lagrangian with respect to the augmented penalty, $\rho > 0$, which is a penalty parameter of the ADMM. The augmented Lagrangian is as follows:

$$\begin{aligned}\mathcal{L}^\rho(Z, U, V) = (1-\alpha)\beta \| Z \|_2^2 + f(\mathcal{T}^t) \| (\frac{U_2 - A}{\mathcal{T}^t}) \|_2^2 + \\ \alpha\beta \| U_1 \|_1 + \frac{\rho}{2} (\| Z - U_1 + V_1 \|_2^2 + \| Z - U_2 + V_2 \|_2^2),\end{aligned}$$

where $U_1 = Z, U_2 = Z,$ and $Z_j \geq 0, j \in [1, \ldots, m]$.

The objective has a form of mixed norm regularization with a non-negative constraints on element values of vector Z (weighted histogram sampling). We apply ADMM to our optimization problem. The ADMM alternately estimates (Z, U, V) which results in iteration steps. The iteration of individual penalty solving steps in scaled form is as follows:

$$Z^{(k+1)} = \underset{Z}{\operatorname{argmin}}\ \mathcal{L}^\rho(Z,\ U^{(k)},\ V^{(k)}) \quad (4)$$

$$U^{(k+1)} = \underset{U_1, U_2}{\operatorname{argmin}}\ \mathcal{L}^\rho(Z^{(k+1)},\ U,\ V^{(k)}) \quad (5)$$

$$V^{(k+1)} = V^{(k)} + Z^{(k+1)} - U^{(k+1)} \quad (6)$$

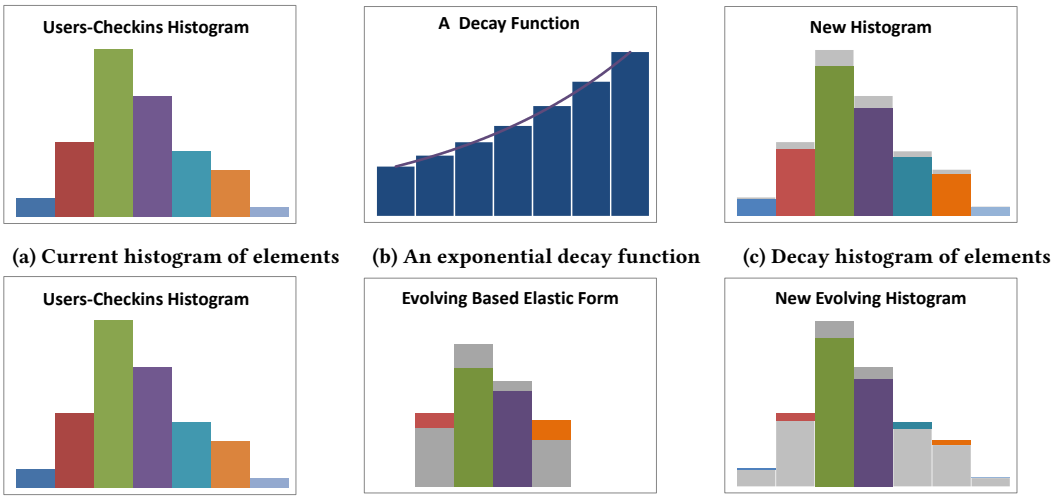
Let C be the constraint set of non-negative, $C = \{Z : Z_j \geq 0, \text{for } j = 1, \ldots, m\}$, and $\prod_C$ be a projection onto C space. The update U step in Eq. 5 involves a projection $\prod_C$ onto set C, rewritten as:

$$U^{(k+1)} = \prod_C (Z^{(k+1)} + V^{(k)})$$

A non-negative projection on C is chosen as: $\prod_C(x) = \max(x, 0)$. The primal and dual residuals which control the convergence of the process are determined at iteration step k as $p^{(k)}$ and $d^{(k)}$ and are computed by:

$$\begin{cases} p^{(k+1)} = Z^{(k+1)} - U^{(k+1)} \\ d^{(k+1)} = \rho_i^{(k)} \times (U^{(k)} - U^{(k+1)}) \end{cases} \quad (7)$$

If the dual residual $d^{(k)}$ approaches zero, then the obtained value of the objective is nearly close to the optimal solution. Meanwhile, the $p^{(k)}$ approaches zero when conditional constraints in the objective function are enforced accurately. We use a maximum number of



(a) Current histogram of elements (b) An exponential decay function (c) Decay histogram of elements

(d) Current histogram of elements (e) An evolving process bases on an elastic net model (f) Evolving histogram of elements

Figure 2: How a histogram of elements evolves over time. From the original histogram (a), the current method applies an exponential decay function (b), a constant decay, on all the data equally to get a new weighted histogram of elements (c). Our proposed method utilizes the idea of elastic net (e) in order to allow us to study the correlation of different groups of elements in the original histogram (d), and finally get an evolving histogram (f).

the iteration steps, l, and a tolerance value, ϵ, on the primal and dual residuals as stopping criteria, given by:

$$\begin{cases} \| p^{(k)} \|^2 & \leq \epsilon \times \max(\| Z^{(k)} \|_2^2, \| U^{(k)} \|_2^2) \\ \| d^{(k)} \|^2 & \leq \epsilon \times \| V^{(k)} \|_2^2 \end{cases} \tag{8}$$

An improvement of ADMM in ρ penalty is a relaxed ADMM approach. Relaxed ADMM algorithms have a relaxation parameter γ. After Z-update step in Eq. 4, they continue to update Z by: $\gamma Z^{(k+1)} + (1-\gamma)U^{(k)}$. Like ADMM, the convergence rate of the relaxed ADMM algorithms depends on the choice of the relaxation parameter.

To address the drawback of ADMM and the relaxed variants, an adaptive method for automatically tuning ρ and γ, called Adaptive Relaxed ADMM (ARADMM) [26], was proposed. The penalty and relaxation parameters of ARADMM are continuously updated at each iteration k based on its safeguarding values. (For details on how to update ρ and γ at every iteration step k, please refer to [26]). In addition to the ability to automatically tune its parameters, ARADMM converges fast and is robust. Since it is also suitable for streaming data, we utilize ARADMM in the solver for our optimization problem. The optimal values of the histogram Θ are obtained when the optimization reaches convergence. Thereafter, the value of the corresponding cell of Θ is increased by 1, i.e., $\Theta_{i,h_i(v)} = \Theta_{i,h_i(v)} + 1$, where 1 is used to indicate that it is the weight of the current data in the stream. Θ is continuously estimated and updated following the changes in the stream.

4.3 Ensemble Sampling

The histogram of a stream is estimated by a matrix $\Theta \in \mathbb{R}^{n \times m}$. The stream is sketched by a vector SK of s values (s elements) based on the weighted values of the histogram elements in Θ. The value of sketch element SK_i ($1 \leq i \leq s$) is computed using a process with three random variables as in [12, 13]. The process of choosing the sketch values of the stream is done as follows:

$$SK_i = \underset{j \in (1;m)}{\operatorname{argmin}} a_{i,j}, \tag{9}$$

where

$$a_{i,j} = \frac{c_j}{y_{i,j} \exp(r_j)},$$

$$y_{i,j} = \exp(r_j(\lfloor \frac{\ln(H_j)}{r_j} + \beta_j \rfloor - \beta_j))$$

r_j, c_j, and β_j are the three random variables that are generated as:

$$r_j \sim Gamma\ (2,1),$$
$$c_j \sim Gamma\ (2,1),$$
$$\beta_j \sim Uniform\ (0,1).$$

H_j is the weighted histogram of element range j, which is estimated as a count-min sketch-modeled histogram Θ. It is given by $H_j = min_i(\Theta_{i,h_i(j)})$. In order to get s samples of SK, we perform this process s times with different values of the three random variables r_j, c_j, and β_j. This process corresponds to a single component, and can be seen as a weak process [20]. We investigate combining several single processes to improve the performance. The ensemble-centric method is used by w components such that each single component C^j sketches Θ^j with a corresponding sketch SK^j. Hence, the estimated sketch will be a $w \times s$ matrix: $SK = [SK^1 \quad SK^2 \quad \dots \quad SK^w]^T$. The ensemble-centric SC of SK is computed based on a scoring function: $SC = Scoring(SK)$, where $Scoring$ is a metric summarizing a sketch from the matrix SK, which

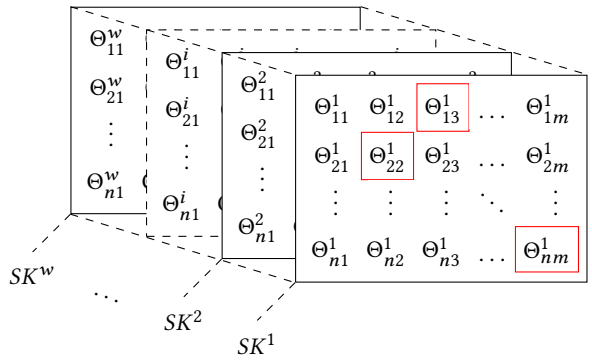


Figure 3: Ensemble w components of histograms.

can be set as the average, min, max, or a random selection. Figure 3 shows an ensemble of w components of histograms of a stream, where each red square in each row i of the matrix is the corresponding histogram of the incoming stream value v_t under hash function h_i. At every observation, a sketch SC of the stream is estimated. Given two sketches SC_1 and SC_2, then the similarity of two sketches is computed using a min-max similarity. The similarity value between the two sketches is used to classify the histograms. For the classification, we use kNN classifier.

4.4 Implementation Details

Figure 4 shows a flow diagram of the proposed method. The flow diagram is processed as follows.

(1) Whenever there is a histogram element arriving in the stream, the corresponding count-min histogram of the element is updated (step $S1$).
(2) In the eRSS, the coefficients of the histogram matrix evolve to new optimal values such that they satisfy an objective constraint (Eq. 3) corresponding to each individual row of the matrix (step $S2$).
(3) After the evolving step, step $S3$ is processed to update the histogram with the current element.
(4) An ensemble w components is used in step $S4$.
(5) Each single component uses a weighted minwise hashing method with a different set of random variables to sketch the corresponding histogram to a vector of s elements in step $S5$.
(6) The final sketch of the histogram is obtained in step $S6$ by using a random selection.

These individual processes are independent. Thus, they can be implemented in a parallel setting. At the same time, we propose an ensemble method which combines several single processes to obtain a more efficient result. The single processes can also be scheduled into tasks in a parallel configuration. The implementation detail of the proposed algorithm is shown in Algorithm 1.

Complexity note: For each incoming element in the stream, the major time-consuming task is the *UpdateTheta* step. The execution time of optimization solver for matrix Θ is bounded by $O(nl)$, where l is the maximum number of iterations of ARADMM. In the worst case, the running time to get the count-min sketch of Θ (n rows) and to produce a k element sketch is $O(n + kw)$. In summary, the runtime complexity of the proposed framework for processing each incoming element is $O(nl + n + wk)$.

In terms of space, a count-min sketch Θ of size $n \times m$ takes $O(nm)$ space, while using k elements for sketching streaming histogram in an ensemble w components takes $O(wk)$ space. Overall, the space complexity is $O(nm + wk)$.

Algorithm 1 eRSS Algorithm

Input: A stream data S
Output: A sketch SK of S, and change alarm

Initialization (α, β, Θ, SK)
for each data point v_t arriving on a stream S **do**
  UpdateTheta(Θ)
  for i from 1 to n **do**
    Increase cell value: $\Theta_{i,h_i(v_t)} \leftarrow \Theta_{i,h_i(v_t)} + 1$
  end for
  for i from 1 to w **do**
    $SK^i \leftarrow SingleRandomization$
  end for
  $SC \leftarrow Scoring(SK)$
  if change detected (labeling using kNN) **then**
    Report alarm
  end if
end for
procedure UpdateTheta(Θ)
  while not converge by Eq. 8 And $++k \leq maxiter$ l **do**
    Update Θ, U, V by Eqs. 4-6
    Update ρ, and γ (refer to [26])
    Update residuals p and d.
  end while
end procedure
function SingleRandomization
  for k from 1 to s **do**
    for l from 1 to m **do**
      Random sampling three variables r, c, β
      $r \sim Gamma\ (2, 1)$,
      $c \sim Gamma\ (2, 1)$,
      $\beta \sim Uniform\ (0, 1)$
      $y_l \leftarrow \exp(r(\lfloor \frac{\ln(H_l)}{r} + \beta \rfloor - \beta))$
      $a_l \leftarrow \frac{c}{y_l \exp(r)}$
    end for
    $SK_k \leftarrow \underset{l}{\text{argmin}}\ a_l$
  end for
  return SK
end function

5 EXPERIMENTS

We evaluated the performance of our eRSS algorithm and compared it with the current state-of-the-art methods, HistoSketch[29] and the POISketch[28] algorithms. All the experiments were carried out on a personal computer running the Windows 10, having a 64 bit

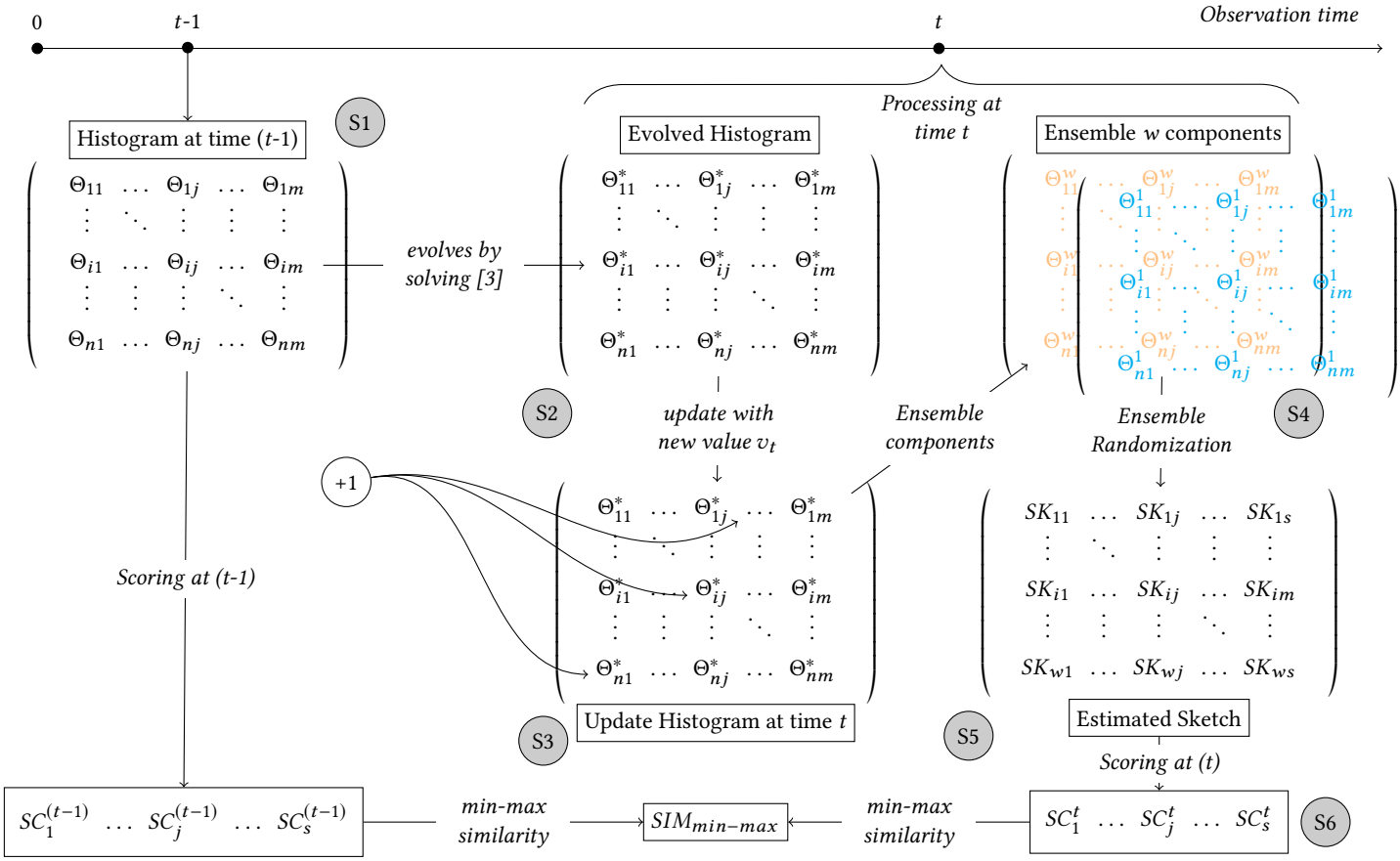


Figure 4: Flow diagram of the proposed method.

Intel i7 2.6 GHz processor and 16 GB of RAM, and the algorithms were implemented in Matlab[1] version R2017b.

5.1 Datasets

We used both synthetic and real-world datasets. The datasets are user check-in activity on locations and are simulated as stream of histogram elements, with the following characteristics:

Synthetic dataset: We created a synthetic data set that contains 1,000,000 check-ins. We simulated the dataset as a stream of histogram elements of 1,000 histograms. Each histogram has 1,000 elements. The histograms are classified into two distribution classes, and each class has 500 histograms. The classes are generated using Gaussian distributions such that the mean and the variance of the distribution are (10,2) and (11,2), respectively.

Real-world dataset: We used *POI datasets* consisting of different user check-in data from Foursquare[2]. Specifically, we perform our experiments on user check-ins in America, Japan, and Turkey, because these places are the most checked places by Foursquare users. The datasets are based on 18 months of user check-ins from April 2012 to September 2013, and were provided by [30]. Each POI in the datasets is classified into hierarchical categories by Foursquare[3]. We pre-processed the datasets to remove all POIs that have a small number of check-ins to prevent skewness and sparsity of data. Particularly, we kept POIs that have number of check-ins greater than 100, 200 and 400 times with America, Japan and Turkey dataset, respectively. Considering the visiting time behaviors for POIs and POI types, previous works [28, 29] have shown that the visiting time behaviors for different types of POIs are different. Each kind of POIs might have a specific check-in time. For example, the number of check-ins for a bar is highest during the night, while the check-ins for a park is highest during daytime. Hence, for a fair comparison, we use fine-grained element as a pair of user and check-in time in a week in the HistoSketch method [28] as a histogram element to make it get the best results. In particular, each week is first split into 168 hours (each day has 24 hours, 7 days a week × 24 = 168 hours). Second, each check-in time is mapped into a range of 168 hours. A pair of user and time range will form an element of a histogram. Since the model used in the eRSS algorithm evolves with respect to time, we use coarse-grained elements of user check-ins. The characteristics of the datasets are summarized in Table 1.

[1] http://mathworks.com/products/matlab/
[2] https://developer.foursquare.com/
[3] https://developer.foursquare.com/docs/resources/categories

Table 1: Dataset characteristics.

Datasets	Check-ins	No of POIs	No of Users
America	303,647	2,555	16,531
Japan	871,646	1,340	14,052
Turkey	780,657	1,783	15,059
Synthetic	1,000,000	1,000	1,000

5.2 Baseline Algorithms

We evaluated the performance of the proposed method, eRSS, and compared the classification accuracy and running time against current state-of-the-art algorithms, POISketch [28] and HistoSketch [29], using the source code provided by the authors. The POISketch algorithm maintains the frequency of histogram elements using Count-Min sketch. It does not consider the weight of the elements in streams. Instead, the POISketch treats the importance of elements in the stream equally. The HistoSketch algorithm considers the weight of the elements by using a constant fading factor. The HistoSketch uses fine-grained elements for similarity preservation. The eRSS uses an automatically tuning coefficient model, combining a random ensemble process to adapt to changes in a stream. We prepared two variants of the eRSS, namely *eRSS-deviation* and *eRSS-evolving*. *eRSS-deviation* is a version where selection parameter β is set to 0, so that it only considers the optimization objective function in the deviation of Θ, and thus fading is utilized on Θ. *eRSS-evolving*, on the other hand, considers both tuning variables α and β.

5.3 Configuration Setting of Parameters

We conducted the experiments by varying the number of items in the sketch, s = 20, 50, 100, 150, and 200. The number of components w was 4, and we used a random selection in the ensemble components. We used the same configuration setting of the histogram as in [29], n = 10 hash functions of range m = 50. The decay factor in the HistoSketch was alternately set to 0.02, and 0.01, as done in the original paper. To classify histograms, we used a k-nearest neighbors (kNN) algorithm [16] because of its implementation simplicity, robustness to noise, and immediate adaptation with new training data. We empirically set k = 5 (i.e., five nearest neighbors) to test the labels for the streaming histograms, where a weight of a hit is given on its position in the k-nearest neighbors having the testing label. Specifically, if a nearest neighbor at p-th position order in the k-nearest neighbors has a label equal to the testing label, then the weight of the hit is set to $1/p$. Otherwise, the hit is set to 0.

We split the datasets into 2 subsets, testing set and training set. Then, we recorded the accuracy of the algorithms when labeling the POIs. The testing-training sets are randomly selected as 50%-50% and 10%-90% of POIs, with the synthetic and real-world datasets, respectively. In the experiments with the synthetic dataset, a drift was simulated at point 300K of the stream, and we evolved 25% of 1000 histograms of the stream randomly during the evaluation. Further, we varied the parameters k, α, and β to study the impacts of variables α and β on the sparsity and hierarchy constraint on the optimization solution of the objective function. On real-world datasets, we performed the classification at the last check-in time every month, and the time function in the objective was chosen as square root of timespan in hour as a unit.

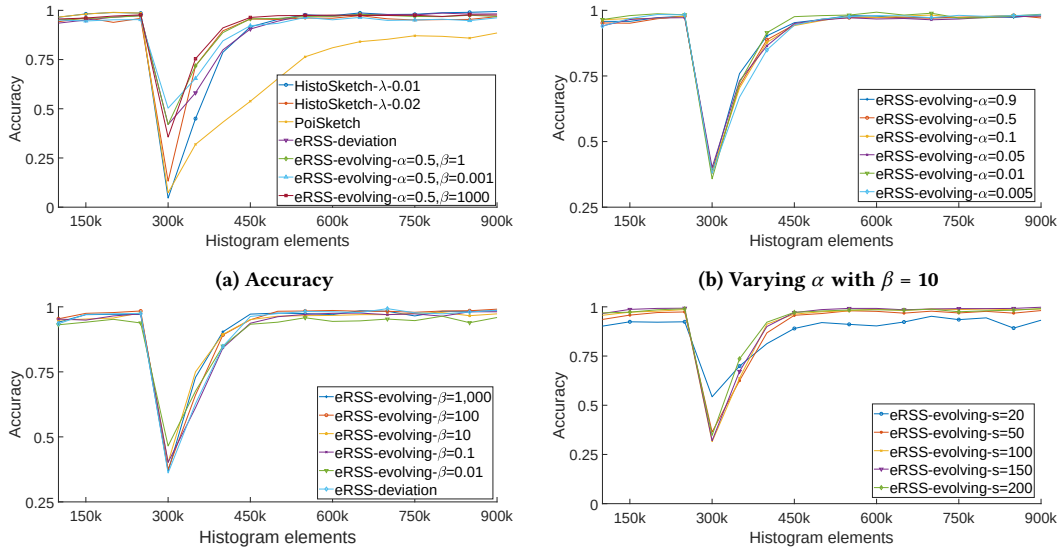
5.4 Experimental Results

Synthetic dataset: Figure 5 shows the classification accuracy results on the synthetic dataset when we set the sketch length to 50. As shown in Fig. 5a, POISketch adapts to a concept drift slowly. It has the worst performance and very low accuracy at the drift point. The reason is that POISketch treats all data in the past equally. Conversely, HistoSketch and eRSS quickly adapt to a concept drift. The classification accuracy of both HistoSketch and eRSS are high, with being the best. We observe that the accuracy values of the algorithms are slightly different at stable points, while the speed of concept drift adaptations are similar. In summary, the average accuracy of eRSS is 91.87%, while HistoSketch and POISketch obtain 89.04%, and 74.64%, respectively.

Furthermore, the results show a large difference in concept drift detection. At a drift point, HistoSketch classifies histogram with a very high error rate. The error rates are 96% and 87% with the values of the forgetting factor λ = 0.01 and 0.02, respectively. In contrast, with eRSS the error rate is much lower. Specifically, with the *eRSS-evolving* variant with β=0.001, the rate is 50%, and 58% with *eRSS-deviation*. In general, the accuracy obtained with eRSS is much better than the previous methods. eRSS is an order of magnitude accurate higher than the baseline algorithms with concept drift. Specifically, our experimental results show that at drift points, eRSS has 12.5x times higher accuracy value than the baseline algorithms (50% versus 4%). This can be explained as follows. In streams, when using the same factor for the whole data as HistoSketch does, it might be efficient with stable data, and it can quickly adapt to changes after changes have occurred. However, around drift points, the data distribution changes dramatically, in terms of both internal characteristics and the relationship with other features. This is exactly why considering evolving models with different factors is useful for detecting concept drifts in streams.

Figures 5b-5d show the classification accuracy of the *eRSS* when varying the different parameters, and the timespan was set to a unit. The experimental results when varying α, β are shown in Figs. 5b-5c. In the first test, β was set to 10, and we varied the value of α within [0.9, 0.5, 0.1, 0.05, 0.01, 0.005]. In the second test, α was set to 0.5, and then we varied the value of β within [1,000, 100, 10, 0.1, 0.01, 0]. We observe that when testing on synthetic dataset, with time being generated sequentially and equally, the impact of relaxation parameter is low, and that selecting a small value of β has significant effect on concept drifts. Further, Figure 5d plots the classification accuracy when we vary the sketch elements s within [20, 50, 100, 150, 200] while α, β are set to 0.5, 1.0. We observe that the adaptive speed of eRSS with concept drift is fast. The result shows that, when the value of s increases, the accuracy at stable points also increases. The lowest value is obtained when s = 20, and the highest accuracy is obtained when s = 200. However, we need to make a trade-off between accuracy and space complexity, i.e., the larger s is, the larger space is consumed.

Real-world POI datasets: Figure 6 shows the classification accuracy of the algorithms on real-world POI datasets. We set the sketch length to 50 and the forgetting factor in HistoSketch to 0.02



(a) Accuracy **(b) Varying α with β = 10**

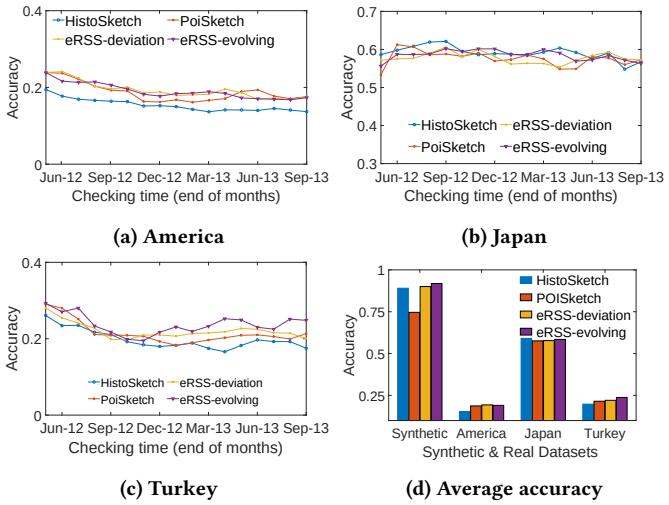
(c) Varying β with α = 0.5 **(d) Varying sketch length**

Figure 5: Classification accuracy on synthetic dataset.

Table 2: Average runtime (second) and velocity (number of elements per ms).

Datasets	America	Japan	Turkey	Synthetic
Histo	24.513	59.512	60.228	60.459
Sketch	12.38	14.64	12.96	16.54
POI	71.329	205.240	168.807	62.711
Sketch	4.26	4.24	4.62	15.95
eRSS	71.411	196.167	169.635	46.815
Deviation	4.25	4.44	4.60	21.36
eRSS	72.816	188.441	175.084	49.055
Evolving	4.17	4.63	4.46	20.39

as suggested in the original paper, which give the best performance for the HistoSketch algorithm. We empirically set the value of (α, β) in evolving variant of eRSS to (0.5, 100). We observe that on the Japan dataset, POISketch performs the worst, while HistoSketch has the best performance, and both variants of the proposed method have similar results as HistoSketch. Nevertheless, the gap among all the results is very small, around 0.6% between HistoSketch and eRSS. On America and Turkey datasets, on the other hand, the worst accuracy is obtained by HistoSketch, and the eRSS is the most accurate algorithm. These results can be analyzed as follows. The Japan dataset is the most dense dataset with largest number of check-ins and smallest number of users and POIs. This increases collaboration and feature selectivity between check-ins, users and POIs. Therefore, it might be sufficient to apply the same factor for all data. However, the America and Turkey datasets are very sparse. Hence, the correlation between check-ins, users and POIs is loose



(a) America **(b) Japan**

(c) Turkey **(d) Average accuracy**

Figure 6: Classification accuracy on real datasets.

or even independent, which, in turn, forms very different check-in behaviors. In the HistoSketch paper, the authors did not seem to have investigated the correlation of different characteristics of data.

Although HistoSketch used fine-grained elements, it applied a constant factor on all the data at different observations. This calls for an evolving model that takes such correlations in a stream into account. The eRSS utilizes an auto-tuning model with respect to different kinds/groups of data to explore the relations between individual check-in behaviors. This's why the eRSS is able to obtain good performance on sparse datasets. Figure 6d plots the average accuracy of the algorithms on the experimental datasets. We learn

that the eRSS has a good performance and has the best average accuracy on three of four test datasets. Importantly, the eRSS quickly adapts to concept drifts but also preserves the high classification accuracy when drift occurs.

In terms of running time, Table 2 displays the execution time and the number of elements can be processed in the streams per millisecond (ms). Interestingly, it reveals that in [29], HistoSketch traded off accuracy (3.5%) for speed (7500x speedup as compared to Histogram-Fine-Forgetting, a variant of POISketch). Note, however, that the efficiency of HistoSketch in term of running time benefits from kNN classification. The result shows that eRSS is a bit slower than HistoSketch, around 2.9 times, which is as we expected. Nevertheless, it is a trade-off between running time and accuracy and concept drift adaptation speed. In addition, the proposed method is not only efficient with high accuracy classification (3.99% higher than HistoSketch for the Turkey and America datasets) and quickly adapts to changes, it also has a fast execution time (2500x speed up, compared to Histogram-Fine-Forgetting according to [29]). With our experimental setup, eRSS is capable of processing streams at high velocity, up to 4000 check-ins per second.

6 CONCLUSION

In this paper, we proposed a novel robust method for sketching streaming histograms based on an ensemble randomization method. To obtain the histogram elements, we developed an algorithm called eRSS, which uses an evolving model with adaptive coefficients. To obtain the values of the coefficients, the eRSS algorithm considers the timestamps of different observations in each coefficient and solves an optimization problem. Here, we studied applying Adaptive Relaxed Alternating Direction Method of Multipliers (ARADMM) as a solver for the optimization problem. To evaluate our approach, we performed extensive experiments on both real-world datasets and synthetic dataset. The results from this evaluation showed the effectiveness and the efficiency of the proposed method. More specifically, our algorithm was able to achieve up to 3.99% higher overall classification accuracy than the baseline algorithms. Overall, our evaluation demonstrated our method's ability to preserve the similarity of generated sketches, with the capability to adapt to concept drifts in data streams, in an online fashion.

For future work, we plan to explore applying our method on high dimensional streaming data, and apply it to solve event detection problems, including anomaly and outlier detection.

REFERENCES

[1] Dean A. Bodenham and Niall M. Adams. 2017. Continuous monitoring for changepoints in data streams using adaptive estimation. *Statistics and Computing* 27, 5 (2017), 1257–1270.

[2] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning* 3, 1 (2011), 1–122.

[3] A. Broder. 1997. On the Resemblance and Containment of Documents. In *Proceedings of the 1997 Compression and Complexity of Sequences (SEQUENCES)*. 21–29.

[4] Lianhua Chi and Xingquan Zhu. 2017. Hashing Techniques: A Survey and Taxonomy. *Comput. Surveys* 50 (2017), 11:1–11:36.

[5] Graham Cormode and S. Muthukrishnan. 2005. An Improved Data Stream Summary: The Count-Min Sketch and its Applications. *Journal of Algorithms* 55, 1 (2005), 58–75.

[6] Graham Cormode and S. Muthukrishnan. 2005. Summarizing and Mining Skewed Data Streams. In *Proceedings of the 2005 SIAM International Conference on Data Mining (SDM)*. 44–55.

[7] Quang-Huy Duong, Heri Ramampiaro, Kjetil Kjetil Nørvåg, Philippe Fournier-Viger, and Thu-Lan Dam. 2018. High utility drift detection in quantitative data streams. *Knowledge-Based Systems* 157 (2018), 34–51.

[8] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A Survey on Concept Drift Adaptation. *Comput. Surveys* 46, 4, Article 44 (2014), 37 pages.

[9] Amit Goyal, Hal Daumé III, and Graham Cormode. 2012. Sketch Algorithms for Estimating Point Queries in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. 1093–1103.

[10] Sudipto Guha, Kyuseok Shim, and Jungchul Woo. 2004. REHIST: Relative Error Histogram Construction Algorithms. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*. 300–311.

[11] Piotr Indyk and Rajeev Motwani. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing (STOC)*. 604–613.

[12] Sergey Ioffe. 2010. Improved Consistent Sampling, Weighted Minhash and L1 Sketching. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM)*. 246–255.

[13] Ping Li. 2015. 0-Bit Consistent Weighted Sampling. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 665–674.

[14] Ping Li, Art B Owen, and Cun-Hui Zhang. 2012. One Permutation Hashing. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS)*. 3113–3121.

[15] Mark Manasse, Frank McSherry, and Kunal Talwar. 2010. Consistent Weighted Sampling. *Microsoft Technical Report* (June 2010).

[16] Thomas M. Mitchell. 1997. *Machine Learning*. McGraw-Hill, Inc., New York, USA.

[17] Viswanath Poosala, Peter J. Haas, Yannis E. Ioannidis, and Eugene J. Shekita. 1996. Improved Histograms for Selectivity Estimation of Range Predicates. In *Proceedings of the 1996 ACM International Conference on Management of Data (SIGMOD)*. 294–305.

[18] Pratanu Roy, Arijit Khan, and Gustavo Alonso. 2016. Augmented Sketch: Faster and More Accurate Stream Processing. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD)*. 1449–1463.

[19] Caitlin Sadowski and Greg Levin. 2007. SimHash: Hash-based Similarity Detection. *Google Technical Report* (2007).

[20] Saket Sathe and Charu C. Aggarwal. 2018. Subspace histograms for outlier detection in linear time. *Knowledge and Information Systems* 56, 3 (2018), 691–715.

[21] Raquel Sebastião, João Gama, and Teresa Mendonça. 2017. Fading histograms in detecting distribution and concept changes. *International Journal of Data Science and Analytics* 3, 3 (2017), 183–212.

[22] Mingwang Tang and Feifei Li. 2014. Scalable Histograms on Large Probabilistic Data. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 631–640.

[23] Robert Tibshirani. 1996. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* 58, 1 (1996), 267–288.

[24] W. Wu, B. Li, L. Chen, C. Zhang, and P. Yu. 2018. Improved Consistent Weighted Sampling Revisited. *IEEE Transactions on Knowledge and Data Engineering* (2018), 1–14.

[25] W. Xie, F. Zhu, J. Jiang, E. P. Lim, and K. Wang. 2016. TopicSketch: Real-Time Bursty Topic Detection from Twitter. *IEEE Transactions on Knowledge and Data Engineering* 28, 8 (2016), 2216–2229.

[26] Z. Xu, M. A. T. Figueiredo, X. Yuan, C. Studer, and T. Goldstein. 2017. Adaptive Relaxed ADMM: Convergence Theory and Practical Implementation. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 7234–7243.

[27] Bo Yang, Ahmed S. Zamzam, and Nicholas D. Sidiropoulos. 2018. ParaSketch: Parallel Tensor Factorization via Sketching. In *Proceedings of the 2018 SIAM International Conference on Data Mining (SDM)*. 396–404.

[28] Dingqi Yang, Bin Li, and Philippe Cudré-Mauroux. 2016. POIsketch: Semantic Place Labeling over User Activity Streams. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*. 2697–2703.

[29] D. Yang, B. Li, L. Rettig, and P. Cudré-Mauroux. 2017. HistoSketch: Fast Similarity-Preserving Sketching of Streaming Histograms with Concept Drift. In *Proceedings of the 2017 IEEE International Conference on Data Mining (ICDM)*. 545–554.

[30] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu. 2015. Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45, 1 (2015), 129–142.

[31] Tong Yang, Yang Zhou, Hao Jin, Shigang Chen, and Xiaoming Li. 2017. Pyramid Sketch: A Sketch Framework for Frequency Estimation of Data Streams. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1442–1453.

[32] Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the Elastic Net. *Journal of the Royal Statistical Society, Series B* 67 (2005), 301–320.