Received: March 5, 2018

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Analyzing Continuous Data Streams Using Improved Stratified Sampling and Ensemble Classification

Gayathiri Kathiresan [1]* Krishna Mohanta [2] Khanaa Velumailu Asari [1]

[1]*Bharath University, 173, Agharam Road, Selaiyur, Chennai, Tamilnadu, India*
[2]*Kakatiya Institute of Technology and Science for Woman, Nizamabad, Thlangana, India*
* Corresponding author's Email: gayathrisenthil.k@gmail.com

Abstract: The streaming data technologies play a vital role in real-time applications. To analyze the data, Random sampling with replacement has a problem in drawing inferences from the small random sample, while sampling without replacement is not preferable to sub-streams that correspond to different sources. Hence, to effectively mine the data streams from heterogeneous sources, this work proposes Adaptive Reservoir sampling Of stream In a Time window (AdROIT) which partitions the streams in a window on time factor and determines the size of historical data on reference window regarding the data changes in the observation window. By measuring the standard deviation of the partitioned window, we can identify whether the changes in statistical properties of a data stream is due to one or multiple sources. The AdROIT allocates the reservoir sampling size to the source, ensures the adaptability, updates the ensemble classifier with dynamically estimated weight, decides accuracy of each member regarding weight. The experimental results show that the AdROIT provides better classification and mining results over heterogeneous data streams. The AdROIT increases the precision by 16%, compared to the Chain sampling under a high degree of heterogeneity. Under the same scenario, the proposed scheme increases the recall by 30 %, more than that in Chain sampling. In high degree of heterogeneity, the Chain sampling utilizes 40kb for storage, more than that of Chain sampling. Finally, the high window size reduces the execution time in AdROIT by 15 seconds and improves the recall by 40%, compared to the Chain sampling.

Keywords: Reservoir sampling, Time windows, Ensemble classifier, Heterogeneous sources, Dynamic weighting.

1. Introduction

The streaming data gain considerable attention in data mining, due to the emerging applications of commercial marketing, sensor networks, and telecommunications. The first requirement for a stream data analysis is that it must be able to process instances 'in-stream', without storing them to perform any operation. Sampling over moving window is the process of selecting some instances of the stream data to represent the characteristics of the whole data. Several data stream mining approaches implement sampling techniques using two popular methods such as sampling with replacement and sampling without replacement [1]. The reservoir sampling is the commonly used technique in data stream mining. It selects fixed size random samples without replacement from a stream of an unknown size [2]. The critical assumption of the reservoir sampling algorithm is independence among the samples, or the sample always representing the whole stream. This assumption is not valid for heterogeneous stream applications. The sampling with replacement eliminates the repetitive instances. However, lack of proper allocation of reservoir size and sampling of stream data from heterogeneous groups are the main drawbacks of the existing reservoir sampling algorithms [3].

Stream data analysis is the process of examining big data to discover hidden patterns and unknown correlations that facilitate better decision making. In real applications, the data is not constant, but a data stream simultaneously has multiple concept drifts. Mostly, the ensemble classifiers are used to incrementally learn the continuously arriving data

for providing fast reaction to the data changes [4]. The most important feature for extracting the knowledge of streaming data classification is time window, which likely to be in a fixed or in an adjustable size. The concept of time windows represents the importance of recent data for stream data classification [5]. In this model, the adaptive windowing restricts the data analysis to the most recent portion of the stream, and moreover, the outdated data are not considered in the classification. This method improves the accuracy of stream data classification algorithms. However, biased or distorted distribution of changes in data streaming tends to skewed classes. To deal with the combined challenges of concept drifts and the imbalanced class problem, this work implements the Adaptive Reservoir sampling Of stream In a Time window (AdROIT) over continuously arriving data streams.

The AdROIT improves the precision of sampling using the components of time sliding window, adaptive reservoir sampling, and pattern matching. The recent data can make a significant impact on sampling, and so the time-based windows include the recently arrived data. However, considering the recent data alone is inefficient in analyzing the new data. Thus, the AdROIT plans to consider the subset of old data as reference window and analyze the newly arrived data on the observation window. An adaptive reservoir sampling dynamically allocates the reservoir to individual sub-groups using mean and standard deviation measurement. The standard deviation between the old and new data assists the AdROIT to identify the scenario of both the gradual and sudden concept drift successfully. Finally, the weighted ensemble classification is utilized for pattern detection and sample mapping. This can classify the patterns precisely even under a concept drift. In this paper, the second section provides a detailed discussion of the previous works related to the big data sampling and their drawbacks. The third section illustrates various components of AdROIT with corresponding mathematical equations. The fourth section discusses the experimental evaluation of AdROIT, and finally, the last section concludes the work.

The main contributions of the proposed AdROIT are as follows.

- The main aim of the proposed AdROIT is to analyze the continuously arriving data streams without storing the entire data streams using ensemble classifier and adaptive sampling technique
- By implementing the standard deviation measurement on time partitioned window individually using stratified sampling technique and the dynamic weighting scheme, the AdROIT is jointly handling unpredictable and imbalanced data changes over a data stream
- To balance the sample size over a time window and the accuracy of representing the entire data, the AdROIT dynamically allocates the reference window size of the data streams with a provable memory guarantee
- By designing a new weighting mechanism for ensemble classification members, the AdROIT to react fast to skewed data changes and achieve good performance on classification accuracy

The performance evaluation of the proposed AdROIT confirms its efficiency of handling different types of changes over data streams.

2. Related works

Traditional sampling techniques need essential ideas to support the data stream applications, especially to handle changes over continuously arriving data streams. Various stream-sampling techniques have been proposed in [1]. Most of the applications select samples from the streaming data in two methods, such as sampling with replacement and without replacement. Sampling with replacement mostly takes the fixed size of samples randomly, whereas sampling without replacement takes the fixed size of samples of every changed stream data. The reservoir and chain sampling technique are the examples of random sampling [6]. However, the conventional chain sampling techniques do not guarantee the sufficient number of samples stored without any repetition in the reservoir, mainly when the data streams emanate from heterogeneous sources [7 - 9]. Even though the chain sampling does not require any prior knowledge about the stream, it occupies reasonable memory and degrades the accuracy of heterogeneous sources. Conventionally, a few of the adaptive reservoir sampling techniques have been proposed in data stream applications.

The majority of the heterogeneous data stream applications exploit stratified sampling technique [10, 11]. Initially, the data population of

heterogeneous sources is clustered into disjoint homogeneous groups, and stratified sampling takes the samples with the same size from those groups. Unlike uniform random sampling, maintaining the non-uniform and bounded size of samples is desirable to reduce the burden of memory management and computational cost [12]. Moreover, a key component needed to implement the adaptive sampling is windowing. In [13], the instances are decomposed into two portions based on the time of instances. To draw a sample of non-uniform size of the sliding windows, the expired sampled items are removed from the sliding window. The continuous monitoring model proposed in [14] takes into account the aggregation process over the distributed data stream. The distributed stream processing physically observes and processes the streams using different processors, whereas the parallel stream processing exploits multiple processors also, but only to reduce the response time. The main aim of distributed streams model is to minimize the communication between processors, while the shared-memory parallel case focuses on the processing efficiency [15]. Integrated learning-based data classification, semi-supervised ensemble approach [16] use constraints to derive unsupervised models. Even though it uses limited instances, it increases the accuracy of classification. The traditional K-means clustering is used to cluster the stream with the same data structure. The Density-Based Distribute Data Stream Clustering (DB-DDSC) determines clusters with different structures under the Big data stream environment [17]. Recruiting similar services in the same clusters provides collaborative recommendation services to the system [18].

Notably, regular research considers the frequency of instances as the primary metric in sampling [19 - 21]. The Hybrid Streaming proposed in [19] maintains histograms for all instances approximately and the frequencies of instances are stored in the internal structure. To determine the frequently appearing patterns across multiple data streams, the hybrid streaming technique [19] addresses the issues of designing an efficient data structure to store historical data patterns, updating of frequently appearing patterns, and pattern matching over continuously arriving data streams. The structure-aware stream sampling in [22] discards one instance when adding a new instance to the reservoir. The hierarchical clustering algorithm provided feedback according to user purchases in the past, and it is utilized to discover the relationships between the users [23]. This technique avoids drifting of the essential data. However, due to high cognitive and uncertain characteristics of massive data mining, these techniques lack in achieving a proper classification with the distance method. Improving mining efficiency even along with the data changes is an important research topic. To accomplish these tasks, it is essential to propose countermeasures based on adaptive windowing and sampling techniques. The utility function in [24] searches the optimal samples using Monte Carlo techniques. The problem in finding an optimal design of reservoir sampling with different types of response variables, such as continuous, counts and proportions. The sampling technique [25] is developed to solve the issues of tuning and timing. To solve those issues, it executes the hybridization of distance and density measures. However, it takes into account only one tuning parameter, named as granularity. Even though the sampling technique is validated on the large dataset, heuristic elimination is essential. In [26], a recursive binary partition is applied to the input instances to select a set of samples for representing the entire stream. The issues related to the concept drift are handled using greedy optimality and explicit error bounds. To reduce the excessive processing time, the sampling technique in [27] ensures overlap independence even though they are inadequate to the stream data generated from heterogeneous sources.

Mainly, current studies have not focused on the problem of maintaining a uniform random reservoir samples over heterogeneous data streams [28]. The statistical properties of stream data change, possibly over time and thus, the allocation of fixed-size reservoir optimizations among heterogeneous sub-streams become a significant concern primarily. To solve this issue, the proposed methodology aims at maintaining the uniformity of the sample among heterogeneous sources, by improving the stratified reservoir of sampling.

2.1 Problem statement

Traditional systems design the storage architectures to scale up with the increasing demands of Big data. However, the stream data makes it impossible to store the entire data on disk. Existing storage systems require unbounded memory space to evaluate queries over streaming data. Notably, sampling techniques reduce the amount of data to process but do not consider the situations, where the flow of input data stream is generated from more than one source. They assume that the arrived data streams as independent and identically distributed. However, the data source heterogeneity creates a negative impact on the

sampling efficiency. Another critical problem to be considered in the sampling techniques is concept drift or unpredictable data changes over time. Existing big data mining techniques are not precise yet on how an analytics system deals with the changes in real-time streaming data. As heterogeneous sources characterize the streaming data, the consolidation of different distributed data sources to a centralized node discourages the data mining process due to the drifting issues. A data stream is an infinite big data scenario in which underlying data distribution of newly arriving data differs from historical data in the real-time applications. The conventional data mining algorithms become inadequate while dealing a large amount of streaming data with changes over time. This issue build need to propose adaptive sampling techniques that effectively support the continuous stream of data arriving from heterogeneous sources.

3. Proposed methodology

It is essential to extract the information and the interrelationship among the continuously arriving streaming data. Knowledge extraction from a hidden data needs to describe the patterns or rules and compare the new patterns with historical data. The reservoir sampling and ensemble classifier are widely employed for data stream mining without storing the entire data. The main drawback in reservoir sampling based ensemble classification is the assumption that the sample data always represent the whole data stream. This assumption turns to the reservoir sampling as invalid for many stream data applications, where the stream data is generated with distinct statistical properties, i.e., sub-groups. Instead of random selection, the proposed work provides adaptive sampling over sliding windows and discovers the knowledge of sub-group data stream without the entire data being stored.

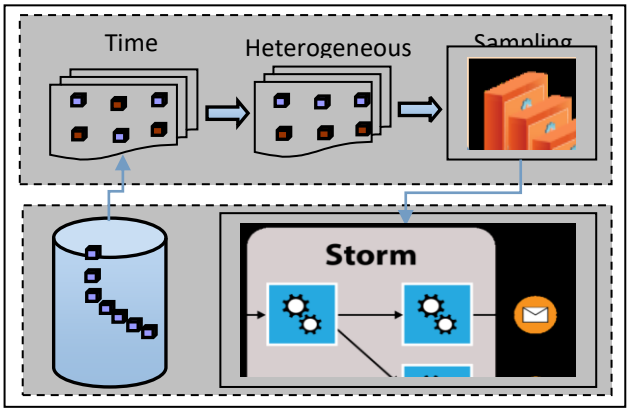

Figure. 1 Adaptive reservoir sampling based distributed data stream processing

The proposed AdROIT includes the components of time sliding window, adaptive reservoir sampling, and pattern matching. Firstly, time-based windows include the stream data arriving within a period. It maintains the recent active data and discards the rest. A subset of old data is selected as a reference window for analyzing the new data in the observation window.

Such windows are called as sliding windows. Dynamic updation of the reference window improves the performance of AdROIT over varied data streams. Secondly, adaptive reservoir sampling dynamically allocates the reservoir to individual sub-groups. For reservoir sampling, the AdROIT takes into account the mean and standard deviation in sampling attribute values. By implementing the standard deviation measurement of time partition window, the AdROIT can differentiate the sudden and gradual changes in the data stream and improve the efficiency of reservoir sampling. Thirdly, the AdROIT exploits the weighted ensemble classification to implement the pattern detection and sample mapping. According to the classification accuracy of majority and minority pattern classes, the AdROIT provides weight to the classifier and precisely classifies the patterns even with data changes. Finally, the analyzed results are stored in the DSMS.

3.1 Sliding window design

Recent works detect the changes in the data stream using comparison techniques. The distribution of new data streams is compared with the past distributed data streams using discrepancy measurement. A subset of past data is selected for the comparison to enhance the window model accuracy. The selected past data in the time window is named as Reference Window *(W_R)*. The new streams arrived within the t time interval forms Observation Window *(W_O)*. A large size reference window which is closer to the entire distribution of data stream S potentially improves the accuracy of data change detection. Still, resource constraints force the size of W_R to be as small. Furthermore, large W_O is likely to reduce the efficiency of change detection technique, due to the data distribution discrepancy. The time interval t of W_O indicates the frequency with which the change detection procedure is triggered. Smaller t tends to more frequent change detection and, thus, it reduces the number of delayed data change detection. However, they increase the computational complexity and may reduce the efficiency of the proposed change detection technique. Therefore, the values of W_R and

W_O have to be determined according to the discrepancy level between reference and observation windows as found in the previous time.

3.1.1. Reference and observation window selection

Consider that the W_R and W_O on stream S contain the S_R and S_O streams, respectively. The timestamps are included with the streams when concatenating those windows. Two instances which have the same value and different timestamps are also included in the Windows, even when they are considered as duplicates. The timestamp guarantees the accuracy in the frequency measure of instances. Therefore, the value of $|(W_R+W_O)|$ is equal to the $|W_R| + |W_O|$ even when the instances in W_R and W_O are likely to be overlapped. Dynamically generating the reference window W_R is the primary concern in varying data streams. The AdROIT dynamically selects the reference window size according to the data changes using the Eq. (1).

After deciding the W_R window size, the new instances are selected using the stratified sampling technique. Where $W_{R'}$ represents the new reference window. When high data change appears between the reference and observation window, the size of the new reference window gets reduced. Otherwise, the AdROIT increases the size of reference window. The intersection between the reference and observation window, i.e. $|W_R \cap W_O|$ decides the size of the new reference window. That means, when the value of intersection between the windows are high, the new reference window size is also increased.

$$|W_{R'}| = \frac{1}{2} \times (|W_R \cap W_O| + W_R) \qquad (1)$$

Fig. 2 below illustrates the dynamic selection of reference window W_R. Let a distribution of stream be initiated at the *t1* time. At the initial time, W_R window records an entire stream S_R, i.e., equal to the size of W_R. The observation window W_O is same as the W_R at the *t1* time. At the time of *t2*, the window W_R is full, and thus, the window gets started to proceed forward.

During *t2* time, the AdROIT concatenates the windows of W_R and W_O, when the changes are not detected in distributed stream data. The deletion of duplicate instances results in same window size. The window $W_{R'}$ selects the instances from the window of either W_R or W_O, as it is a recent reference instance. During *t2* time, small data changes appear and so the new reference window $W_{R'}$ includes the instances maximally from the old reference window W_R. The large size window is appropriate for analyzing the continuously incoming streams when there is no or negligible change in the data. Thus, frequently appearing instances are selected from either the W_R or W_O, and such of those samples replace the stream in W_R at time *t3*. This merge and the selection processes are triggered for every interval. When the data deals large changes, the AdROIT reduces the size of reference window and maximally includes the frequently appearing instances from the old observation window. Fig. 2 shows the dynamic selection of window size under different cases of data changes.

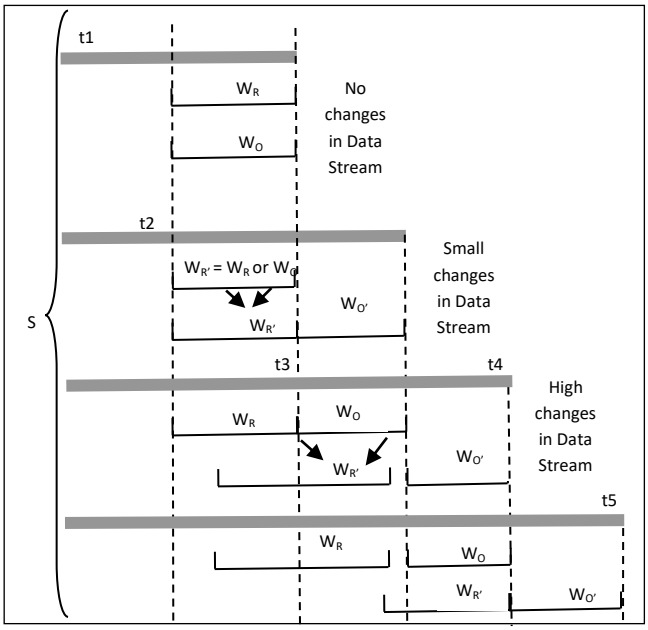

Figure. 2 Dynamic reference window-size selection

3.2 Adaptive reservoir sampling

Instead of processing entire data streams on the window, the AdROIT enhances the stratified sampling and selects an appropriate number of newly arrived samples from W_O to represent the entire data. Most conventional works allocate a fixed size reservoir of R to observation window W_O, even when the data is generated from heterogeneous sources. An input data stream of k sources (S_i) is represented as sub-streams S_1, S_2, S_3,.....S_k. Each source generates n number of streams (S_{in}). The sequence of instances in each source is represented as S_{i1}, S_{i2}, ...S_{in}, such that $S_i \cap S_j = \emptyset$ and $\cup S_i = S$. Considering a total available reservoir size is $|R|$ instances in a reservoir r, and the main aim of sampling is to allocate $|R|$ optimally among the k sub-streams subject to the data change frequency. The AdROIT clusters the instances in a window into k disjoint homogeneous strata, R_1 , R_2 , . . . , . . . , R_k. From each stratum, $|R|/k$ samples are taken

randomly. Due to the heterogeneity among sources, the instances in one strata must not belong to any other strata, such that $R_m \cap R_n = \emptyset$ and $\cup R_i = R$. A stratified sample of an estimated dynamic size provides high statistical precision than a stratified reservoir sampling, because of the usage of statistical properties such as mean and variance.

$$\boldsymbol{Discrepancy\ D(R_i)} = \mathbf{1} - \left[\left\{ \sum_{t=1}^{h} \sum_{j=1}^{|(R_i)t|} \boldsymbol{\alpha_t \times Y_{ij(t)}} \right\} \Big/ \left[\left\{ \sum_{j=1}^{(R_i)} \boldsymbol{Y_{ij}} \right\} \Big/ |\boldsymbol{R}| \right] \right] \quad (2)$$

$$\boldsymbol{Total\ Discrepancy\ TD} = \mathbf{1} - \left[\left\{ \sum_{m=1}^{k} \left\{ \sum_{t=1}^{h} \sum_{j=1}^{|(R_k)t|} \boldsymbol{\alpha_t \times Y_{kj(t)}} \right\} \right\} \Big/ \left[\left\{ \sum_{j=1}^{|R_k|} \boldsymbol{Y_{kj(t)}} \right\} / |\boldsymbol{R}| \right] \right] \quad (3)$$

Eqs. (2) and (3) illustrate the measurement of Discrepancy (D) for reservoir R_i and Total Discrepancy (TD) respectively. The AdROIT divides the instances in R_i into h divisions, where the t represents the time interval within a window and t varies from 1 to h. Where Y_{ij} represents the sampling attribute value of the j_{th} instance belongs to Ri, and α_t denotes the standard deviation of the instances in Ri, at t time. When the weighted summation of the attribute value in the numerator is closer to the average attribute value measured in the denominator, the discrepancy value is increased. When the discrepancy value is high, the size of reservoir R_i should be increased. Since, the corresponding attribute has a significant impact on the stream data concept, compared to others. The conventional stratified reservoir considers a reservoir entirely for measuring the discrepancy. However, it tends to imprecise measurement when sudden changes occur in the data stream. When there is a gradual change, the standard deviation is increased linearly, resulting in a high discrepancy. Consider an example such as a sudden change occurs at a midpoint, but before and after the midpoint, there are no changes. In such cases, the standard deviation is also high, and the conventional discrepancy measurement shows no differences in the above two cases. This factor reduces the accuracy of stratified reservoir sampling. Instead of taking the entire samples, the AdROIT measures the standard deviation for a partitioned window separately and measures the discrepancy value.

Substituting Eqs. (2) and (3) in Eq. (4), the number of samples required for the reservoir R_i is estimated with the knowledge of data changes.

$$|\boldsymbol{R_i}| = |\boldsymbol{R}| \times \{\boldsymbol{D(R_i)/TD}\} \quad (4)$$

Notably, when the $D(R_i)$ value is zero, the reservoir size is assigned as one, instead of zero. That means the samples in a subset represents the same concept. The AdROIT determines the sample size of the stratum reservoir under the proportional allocation and selects the samples from the observation window. Instead of equal preference to all strata, the AdROIT optimally allocates the reservoir size on the change of the data stream. The AdROIT adjusts the allocation when the new streams appear, or existing instances disappear from the old stream data or the statistical properties of data stream get changed. Thus, the AdROIT generates and matches the pattern, and successfully analyzes the continuously arriving data streams using the sample data.

3.3 Pattern matching and storage using ensemble classifier

Appearing changes in continuously arriving data streams tend to imbalanced data streaming and classification. To deal with this challenge, the AdROIT provides dynamic weight to the ensemble classifiers, according to their classification accuracy. The ensemble classification method with dynamic weighting scheme is illustrated in Fig. 3. For each new stream, an ensemble classifier is trained. The result of all the classifiers in the ensemble decides the dynamic weight using majority vote. The vote is considered as a majority, for instance, when the ensemble member classifies the instances under the class which is predicted by most of the ensemble members for the same window. The classifiers which have errors on same instances render minority class. Eq. (5) demonstrates the estimation of classification Accuracy (AC) for the ensemble members C_i individually using majority and minority classes, predicted on a window. It is measured using the concept of result overlapping with other classifiers. When the result of a classifier is highly overlapped with other classifiers, it returns high majority vote.

$$\boldsymbol{AC_{Ci}} = \left\{ \{\boldsymbol{Majority\ VoteMinority\ Vote}\}_{\boldsymbol{Classes}} \Big/ \boldsymbol{Total\ Classes} \right\} \quad (5)$$

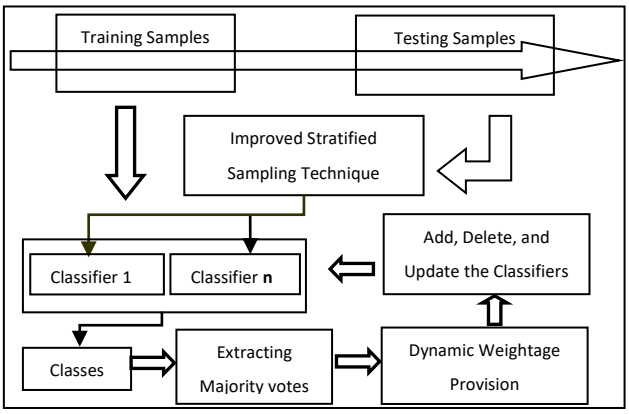

Figure. 3 Architecture of ensemble classifier in AdROIT system

The accuracy of the classifiers is high when most of the instances are classified into the majority vote classes. Otherwise, the AC_{Ci} tends to negative results. The reduced accuracy develops an adverse impact on the Dynamic Weighting (*DW*) at t time, as shown in Eq. (6).

$$\boldsymbol{DW_{Ci}(t) = 0.5 \times (AC_{Ci} \times DW_{Ci})(t-1)} \quad (6)$$

For the classifiers that are currently used in the ensemble, the dynamically estimated weight is updated for every window. When the classification accuracy and previous dynamic weight are high, the classifier is used for the further processing.

When the data changes are detected, the new classifier is added to the ensemble by replacing the old classifier with less weight. Dynamic weight provisioning and classifier updation ensure the accuracy of the AdROIT system over continuously varying data streams.

4. Performance evaluation and setup

The performance of the proposed AdROIT is evaluated against the Chain sampling [6] with the ensemble classifier algorithm. These algorithms have been executed using the following tools. This work exploits Apache Storm 1.0.2 version as a processing engine. A storm has a topology built by the spouts and bolts. A spout reads instances from an external source and emits them into the bolt, whereas the bolt performs the proposed sampling, aggregation, joining, and pattern matching. The Storm engine does not include any Machine Learning (ML) library, and thus, the SAMOA platform provides third-party ML for Storm with several classifications and clustering implementations. The target/SAMOA-Storm-0.4.0-SNAPSHOT.jar file is used to deploy Apache SAMOA in Storm.

The SAMOA supports the AdROIT to create ensemble classifiers in Storm engine. Storm can integrate with any queuing and any database system. In addition to processing and ML libraries, the tool Kafka version 0.8 is used for data stream movement and interaction with other tools. The retrieved patterns and historical pattern are stored in the Druid database version 0.7.3. Storm bolts to do pattern classification, and sending the patterns to the Druid.

4.1 Weather dataset and metrics

This work experiments weather dataset. The dataset is collected from different sources, and the characteristics of the data from each source are different. It develops gradually, and sudden concept drifts over streaming data. The weather dataset includes the data on 30 major USA cities, and it is collected from 18 sources or websites for every 45 minutes. The dataset has some common attributes among heterogeneous sources such as Timestamp, Location, Temperature (°F), and Condition. According to the AdROIT, the sampling technique collects the samples from each source with dynamically estimated window size and the bolt in Storm executes the ensemble classification. To find the accuracy of sampling and ensemble classifier, this work considers the source of Yahoo in collecting the weather dataset as the gold standard. Due to the usage of the gold standard, this evaluation has provided the metrics of precision and recall, in addition to the execution time and memory utilization. Intuitively, three factors affect the performances of the AdROIT and Chain sampling algorithms over a data stream from heterogeneous sub-streams, namely Degree of heterogeneity, Number of Sub-Streams, and Window Size. The degree of heterogeneity refers the level of discrepancy between sources of stream data. These three parameters are used in the performance evaluation between AdROIT and Chain sampling, regarding the precision, recall, execution time, and memory utilization.

Precision: It is defined as the ratio between the numbers of correct results to the total number of returned results.

$$\boldsymbol{Precision = No.of\ Correct\ Results / Total\ Number\ of\ Returned\ Results} \quad (7)$$

Recall: It is defined as the ratio between the number of correct results to the total number of results that have been returned.

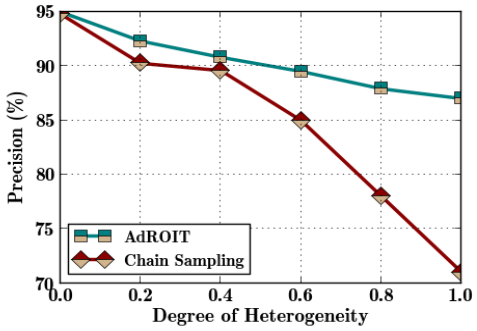

Figure. 4 Degree of heterogeneity vs. precision

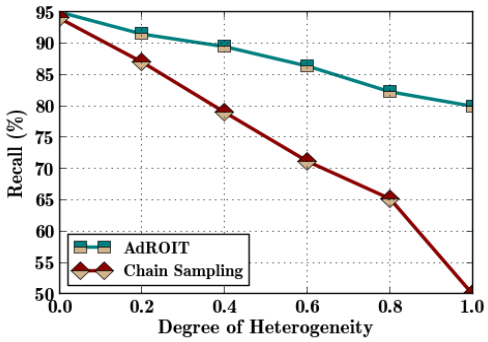

Figure. 5 Degree of heterogeneity vs. recall

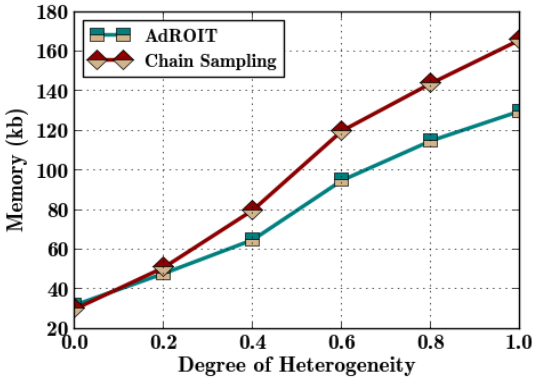

Figure. 6: Degree of heterogeneity vs. memory

$$Recall = \frac{No.\ of\ Correct\ Results}{Total\ Number\ of\ Returned\ Results\ must\ have\ been\ returned} \quad (8)$$

Execution Time (Sec): The time taken by the AdROIT to take samples from continuously arriving streaming data and classify the patterns using ensemble classifier.

Memory Utilization (Bytes): The memory usage of the AdROIT during sampling process on windows and ensemble classification.

4.2 Experimental results

Fig. 4 demonstrates the precision of AdROIT and Chain Sampling over various degrees of heterogeneity. When the degree of heterogeneity is low, the precision of AdROIT and Chain sampling are relatively close to one another compared with the case of a higher degree of heterogeneity. The observed influence of the degree of heterogeneity on the precision of the AdROIT is reasonable, compared to the Chain sampling. The AdROIT allocates the size of sub-reservoir on the heterogeneity of the sources. For instance, both the AdROIT and Chain sampling attains nearly 95% of precision with no heterogeneity among the sources. However, the difference between the algorithms increases with the degree of heterogeneity. As the chain sampling exploits divide and conquer and provides the uniform distribution of samples, the degree of heterogeneity negatively impacts the performance of Chain sampling technique. Moreover, the chain sampling assumes that the most recent items of the stream represent the context of the entire window. However, this is not perfect with the case of a higher degree of heterogeneity. The performance of AdROIT improves the precision by 16% more than that of Chain sampling with a high degree of heterogeneity.

Fig. 5 shows the influence of the degree of heterogeneity on the recall of both AdROIT and Chain sampling. Low heterogeneity among stream data sources is attributed to the uniform sampling in AdROIT, which is closer to the performance of Chain sampling. That is, the statistical change among sub-streams relatively is small and, therefore, it does not cause too much difference in uniformly allocated sub-reservoir sizes.

For instance, the recall of both the AdROIT and Chain sampling is equal to 95%. However, the significance of heterogeneity increases the degradation in the recall while escalating the degree of heterogeneity. As the heterogeneity among sources represents the running statistics of sub-streams change over time, thus leads the AdROIT system to select a fewer number of samples for each source, resulting in recall degradation. High degree of heterogeneity decreases the recall of AdROIT from 95 to 80%. As the chain sampling excludes the frequency of the change among sub-streams in reservoir allocation, the recall of chain sampling has

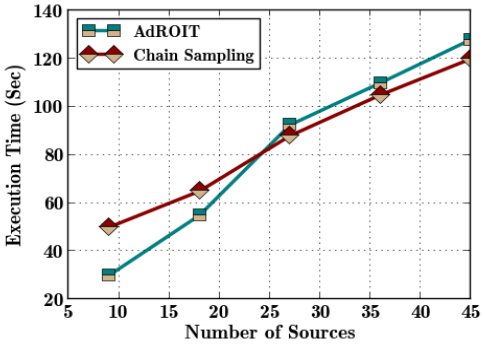

Figure. 7 Number of sources vs. execution time

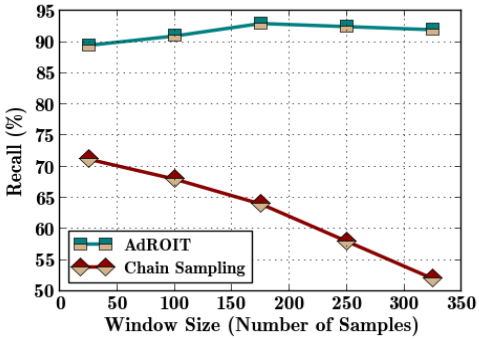

Figure. 8 Window size vs. recall

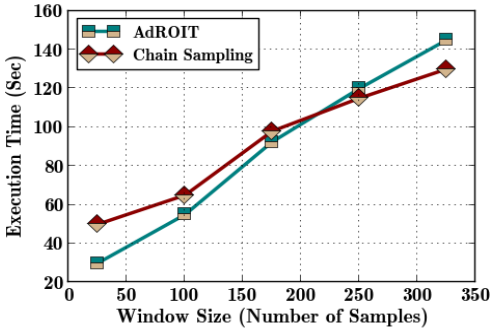

Figure. 9 Window size vs. execution time

degraded by 30% with the scenario of high heterogeneity, compared to the AdROIT.

Fig. 6 holds the experiment results on the memory utilization on both AdROIT and Chain sampling while varying the degree of heterogeneity from 0 to 1. The AdROIT has demonstrated less memory occupation since the proper measurement of standard deviation over time partitioned window efficiently allocates the ensemble classifiers. Due to no variation in the statistic of sub-streams over time, initially, both the AdROIT and Chain sampling occupy same memory, i.e. nearly 30 KB. The concurrent graphical representations in Fig. 6 reveal the adaptivity of AdROIT under the scenario of high heterogeneity. When a new classifier adds in both AdROIT and Chain sampling, memory has to be enhanced to the sub-reservoir wherein the sub-streams arrive newly. However, improper allocation of classifiers due to the result of Chain sampling increases the memory utilization up to 165 KB. Without the consideration of the standard deviation among the sampling attributes of sub-streams, the sampling techniques are impossible to allocate the classifiers properly. Thus, it results in an unnecessary increase in the number of classifiers and memory utilization.

Fig. 7 shows the execution time against the number of sources. Notably, the increasing number of sources also increases the degree of heterogeneity. The number of samples has a significant influence on the execution time of AdROIT and Chain sampling. From low to a medium number of sources (i.e., from 9 to 18), it is observed that the Chain sampling extends the execution time of an algorithm longer than that in AdROIT. For a higher number of sources (e.g., 27 or higher), there is a decrement in execution time compared to the AdROIT. The reason for this is that Chain sampling does not take into account the heterogeneity measurement between sub-streams, whereas the proposed AdROIT does. The proper allocation of reservoir size and classifiers using dynamic weight influences the speed of execution only with the scenario of a high number of sources. Even though the execution time of AdROIT is high compared to the Chain sampling, the recall value of AdROIT is always better than the Chain sampling.

Figs. 8 and 9 reveal the result of the recall and execution time against the window-size with the same degree of heterogeneity (0.4) respectively. As in Fig. 8, the sub-sample recall of AdROIT distinctively maintains a superior performance over the Chain sampling. The AdROIT recall oscillates within the range of 90 to 92%, even when the window size is increased from 25 to 325 samples. The Chain sampling technique does not provide importance to the historical information, and so it increases the samples with window size, resulting in weaker recall value. For instance, with the window size of 325 samples, the AdROIT increases the recall by 40%, more than that of the existing work. However, from Fig. 9, it is observed that the execution time of AdROIT extends longer than Chain sampling method after the point of 175 samples. This is due to the computational time of discrepancy and updating frequency of dynamic weight to the classifiers.

5. Conclusion

This paper has presented an adaptive reservoir sampling of the stream in a time window, AdROIT over continuously arriving data streams. The proposed sampling model allocates the samples on the degree of heterogeneity among sources, as it introduces more than one source for stream generation. The standard deviation of time partitioned window improves the accuracy of discrepancy measurement that influences the accurate representation of entire stream data with sampled instances. Moreover, the proposed AdROIT includes the ensemble classifier with the dynamic weight, which is estimated based on the extracted majority votes for each window. The majority of votes decide the accuracy of the classifier as well as its lifetime in the ensemble. This system improves the accuracy even with the unbound size of arriving streams over time. The evaluated results show the effectiveness of the AdROIT and its outstanding performance over the Chain sampling. Unlike the existing system, AdROIT allocates the sample instances with the knowledge of statistical changes over time, instead of random selection, and thus it ensures the enhanced accuracy of sampling. The dynamic weight of AdROIT enables accurate measurement and proper maintenance of the classifiers in the ensemble. From the evaluation, the AdROIT results indicate 80-90% of recall, when varying the degree of heterogeneity among sources, as compared to Chain sampling.

There are possible directions for the sampling technique to extend in the future and those are discussed as follows.

- In future, adaptive sampling techniques need to be extended to support the multivariate sampling environment.
- In future, it is essential to choose the stream data by importance, by analyzing the dependencies multi- variate response.
- Diversity between a different subset of sampling needs to be considered for improving the prediction accuracy.

References

[1] P.J. Haas, “Data-stream sampling: basic techniques and results", *Data Stream Management, Springe*r, Berlin Heidelberg, pp. 13-44, 2016.

[2] [2] W. Hu, and B. Zhang, “Study of sampling techniques and algorithms in data stream environments”, In: *Proc. of the 9th International IEEE Conf. on Fuzzy Systems and Knowledge Discovery*, pp. 1028-1034, 2012.

[3] B. Park, G. Ostrouchov, N. Samatova, and A.Geist, “Reservoir-based random sampling with replacement from data stream”, In: *Proc. of SIAM International Conference on Data Mining*, pp. 492-496, 2004.

[4] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM Computing Surveys*, Vol. 46, No. 4, pp. 1-37, 2014.

[5] D. Mejri, R. Khanchel, and M. Limam, “An ensemble method for concept drift in nonstationary environment”, *Journal of Statistical Computation and Simulation*, Vol. 83, No. 6, pp. 1115– 1128, 2013.

[6] E.I.R. Sibai, Y. Chabchoub, J. Demerjian, Z. Kazi-Aoul, and K. Barbar, "A performance study of the chain sampling algorithm", In: *Proc. of the Seventh International Conference on Intelligent Computing and Information Systems*, pp.487-494, 2015.

[7] S.P. Efraimidis and P. Spirakis, “Weighted random sampling with a reservoir”, *Information Processing Letters*, Vol.97, No.5, pp.181-185, 2006.

[8] S. Tirthapura and P.D. Woodruff, “Optimal random sampling from distributed streams revisited”, In: *Proc. of International Symposium on Distributed Computing*, pp. 283-297, 2011.

[9] S.P. Efraimidis, "Weighted random sampling over data streams", *Algorithms, Probability, Networks, and Games, Springer International Publishing*, pp. 183-195, 2015.

[10] S. Chaudhuri, G. Das, and V. Narasayya, “Optimized stratified sampling for approximate query processing”, *ACM Transactions on Database Systems*, Vol. 32, No. 2, pp. 1-50, 2007.

[11] M. Cuong, Y. Cao, R. Klamma, and M. Jarke, “A Clustering Approach for Collaborative Filtering Recommendation Using Social Network Analysis”, *Journal of Universal Computer Science*, Vol. 17, No. 4, pp. 583-604, 2011.

[12] R. Gemulla, P.J. Haas, and W. Lehner, "Non-uniformity issues and workarounds in bounded-size sampling", *Journal of Very Large Data Bases*, Vol.22, No. 6, pp.753-772, 2013.

[13] V. Braverman, R. Ostrovsky, and C. Zaniolo. "Optimal sampling from sliding windows", *Journal of Computer and System Sciences*, Vol.78, No.1, pp. 260-272, 2012.

[14] G. Cormode, S. Muthukrishnan, K. Yi, and Q. Zhang, "Continuous sampling from distributed streams", *Journal of the ACM*, Vol. 59, No. 2, pp. 1-25, 2012.

[15] N. Jain, M. Pozo, R. Chiky, and Z. Kazi-Aoul, "Sampling semantic data stream: Resolving overload and limited storage issues" In: *Proc. of first International Conf. on Advanced Data and Information Engineering*, pp. 41-48, 2013.

[16] J. Liu, G. Xu, D. Xiao, L. Gu, and X. Niu, "A Semi-supervised Ensemble Approach for Mining Data Streams", *Journal of Computers*, Vol.8, No.11, pp. 2873-2879, 2013.

[17] B. Gao and J. Zhang, "Density Based Distribute Data Stream Clustering Algorithm", *Journal of Software*, Vol.8, No.2, pp.435-442, 2013.

[18] S. Joshi and C. Jermaine, "Robust stratified sampling plans for low selectivity queries", In: *Proc. of IEEE International Conference on Data Engineering*, pp.199–208, 2008.

[19] J. Guo, P. Zhang, J. Tan, and L. Guo, "Mining frequent patterns across multiple data streams", In: *Proc. of the 20th ACM International Conference on Information and Knowledge Management*, pp.2325–2328, 2011.

[20] A. Cuzzocrea, "Data warehousing and knowledge discovery from sensors and streams", *Knowledge and Information Systems*, Vol.28, No.3, pp.491–493, 2011.

[21] K. Kutzkov and R. Pagh, "Consistent subset sampling", In: *Proc. of Scandinavian Workshop on Algorithm Theory*, pp. 294-305, 2014.

[22] E. Cohen, G. Cormode, and N. Duffield, "Structure-aware sampling on data streams", In: *Proc. of ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, pp.197–208, 2011.

[23] S.R. Deputter, T. Xiong, and S. Wang, "Combining collaborative filtering and clustering for implicit recommender system", In: *Proc. of the 27th International Conf. on Advanced Information Networking and Applications*, pp. 748-755, 2013.

[24] G.M. Falk, M.J. McGree, and N.A. Pettitt, "Sampling designs on stream networks using the pseudo-Bayesian approach", *Environmental and Ecological Statistics*, Vol.21, No.4, pp.751-773, 2014.

[25] F. Ros and S. Guillaume, "DIDES: A Fast and Effective Sampling for Clustering Algorithm", *Knowledge and Information Systems*, Vol.50, No.2, pp.543-568, 2017.

[26] C. Cervellera and D. Maccio, "Distribution-preserving Stratified Sampling for learning Problems", *IEEE Transactions on Neural Networks and Learning Systems*, Vol.29, No.7, pp.2886–2895, 2018.

[27] Y. Tao, X. Hu, and M. Qiao, "Stream Sampling Over Windows With Worst-Case Optimality and ℓ-overlap independence", *Journal of Very Large Data Bases*, Vol.26, No.4, pp.493-510, 2017.

[28] A. Boicea, C. Truica, F. Radulescu, and E. Buse, "Sampling Strategies for Extracting Information from Large Data Sets", *Data and knowledge Engineering*, 2018, Accepted for publication.