FLoMSqueezer: An Effective Approach For Clustering Categorical Data Stream

M Sora[1], S Roy[2] and S I Singh[3]

[1]Department of Computer Science & Engineering,
Rajiv Gandhi University, Doimukh 791112, Arunachal Pradesh, INDIA

[2]Department of Information Technology,
North Eastern Hill University, Shillong 793022, Megalaya, INDIA

[3]Department of Computer Science & Engineering,
Tezpur University, Tezpur 7984028, Assam, INDIA

Abstract

Squeezer is an effective histogram based approach for categorical data stream clustering. Drawback of Squeezer is that it is not scalable in terms of memory. The size of histogram increases with the increase in records in the dataset. Accommodation of unpredictably large histogram in the main memory is not always feasible. To handle the bottleneck, a modified version of Squzeer, FLoMSqueezer, is proposed in this paper. It uses concise sampling technique for handling increasing memory requirement by the Squzeer. Experimental results shows that proposed approach scales better in terms of quantitative cluster, memory as well as execution time.

Keywords: *Cluster analysis, data stream, histogram, sampling, quantitative cluster.*

1. Introduction

With the advent of storage technology, internet and network technology, enormous amount of incremental data generated every day in a timely fashion in the form of data stream. Handling such a huge stream of data gives rise to a new data processing model [1].Unlike traditional data processing, which is normally static in nature, stream data arrives in the form of continuous, high-volume, fast, and time-varying streams and the processing of such streams entail a near real-time constraint. Data streams knowledge discovery systems are usually constrained by three limited resources: time, memory and sample size. Nowadays, time and memory seem to be the bottleneck for machine learning applications, mainly the last one. Many important applications, ranging from network security, sensor data processing, to stock analysis, climate monitoring, are a part of the data stream model. From the last decade, data mining [3] meaning extracting useful information or knowledge from large amounts of data, has become the key technique to analyze and understand data. Typical data mining tasks include association mining, classification, and clustering. These techniques help find interesting patterns, regularities, and anomalies in the data. However, traditional data mining techniques cannot directly apply to data streams. This is because mining algorithms developed in the past target disk resident or in-core datasets, and usually make several passes of the data. Mining data streams are allowed only one look at the data, and techniques have to keep pace with the arrival of new data. Furthermore, dynamic data streams pose new challenges, because their underlying distribution might be changing. The problem of clustering becomes more challenging when the data is categorical, that is, when there is no inherent distance measure between data values. This is often the case in many domains where data is described by a set of descriptive attributes, some of which are neither numerical nor inherently ordered in any way. A data stream is a massive unbounded sequence of data elements continuously generated at a rapid rate. Consequently, the knowledge embedded in a data stream is more likely to be changed as time goes by. Identifying the recent change of a data stream, especially for an online data stream, can provide valuable information by analysis of the data stream [4].

2. Data stream mining

Data stream are continuous flow of data. A data stream mining is ordered sequence of points that can read only one or small number of times. Formally, a data stream is a sequence of point $x_1,\ldots x_i,\ldots\ldots x_n$ read in increasing ordered of indices i. according data stream model[2]. The performance of an algorithm that operates on data stream is measured by number of passes that the algorithm must make over stream, when constrained in terms of available memory, in addition to the more conventional measures. The data stream model is motivated by emerging application involving massive data sets, e.g customer click streams, telephone records; large set of web pages, multimedia data, and sets of retail chain transaction can be modeled as data streams. These data are far too large to fit in main memory and are typically stored in secondary storage devices, making access, partially random access very expensive. Data stream algorithms access the input only a linear scan without random access and require to scan only once over the data. Furthermore, since amount of data far exceeds the amount of space (main memory) available to the algorithm, it is not possible to remember to much of data scanned in the past. These scarcity of space necessities the design of novel kind of algorithm that stores only the past data, leaving enough memory for future data. Clustering has recently been studied across several disciplines, but only few techniques have developed scalable very large datasets. In more recently years, a few categorical stream clustering algorithms has formulated such as Squeezer [5] algorithms for categorical data streams.

2.1 Categorical Domains and Attributes

Categorical Domains and Attributes: Let $A_1, \ldots, A_m$ be a set of categorical attributes with domains $D_1,\ldots, D_m$ respectively. Let the dataset $D = \{X_1, X_2, \ldots, X_n\}$ be a set of objects described by m categorical attributes, A1, …, Am. The value set V_i of A_i is set of values of A_i that are present in D. For each $v \varepsilon V_i$, the frequency f (v), denoted as fv, is number of objects $O \varepsilon X$ with $O.A_i = v$. Suppose the number of distinct attribute values of Ai is pi, we define the histogram of A_i as the set of pairs: $h_i = \{(v_1, f_1), (v_2, f_2),\ldots,(v_{p_i}, f_{p_i})\}$. The histogram of the data set D is defined as: $H = \{h_1, h_2, \ldots, h_m\}$.

2.2 Dissimilarity Measures: Let X, Y be two categorical objects described by m categorical attributes. The dissimilarity measure between X and Y can be defined by the total mismatches of the corresponding attribute values of the two objects. The smaller the number of mismatches is, the more similar the two objects. Formally,

$$d_1(X,Y) = \sum_{j=1}^{m} \delta(x_j, y_j) \quad (1)$$

where $\delta(x_j, y_j) = \begin{cases} 0 & (x_j = y_j) \\ 1 & (x_j \neq y_j) \end{cases}$ (2)

Given the dataset D = {X1, X2, …, Xn} and an object Y, The dissimilarity measure between X and Y can be defined by the average of the sum of the distances between Xi and Y.

$$d_2(D,Y) = \frac{\sum_{j=1}^{n} d_1(x_j = y)}{n} \quad (3)$$

If we take the histogram H = {h1, h2, …, hm} as compact representation of the data set D, eq (3) can be redefined as

$$d_2(H,Y) = \frac{\sum_{j=1}^{m} \phi_1(x_j = y)}{n} \quad (4)$$

Where

$$\phi(h_j, y_j) = \sum_{l=1}^{p_j} f * \delta(v_j, y_j) \quad (5)$$

In some cases, it is convenient to use similarity rather than distance. Similarity between Y and H is define as

$$Sim(H,Y) = \frac{\sum_{j=1}^{m} \psi(h_j, y_j)}{n} \quad (6)$$

Where

$$\psi(h_j, y_j) = \sum_{l=1}^{p_j} f * (1 - \delta(v_j, y_j)) \quad (7)$$

From the implementation efficiency viewpoint, eq. (6) can be computed more efficiently because it only requires computing the frequencies of matched attribute value pairs [1].

The clustering accuracy for measuring the clustering results was computed as follows. Given the final number of clusters, k, clustering accuracy r was defined as:

$$r = \frac{\sum_{i=1}^{k} a_i}{n}$$

where n is the number of record in the dataset, a_i is the number of instances occurring in both cluster i and its corresponding class, which had the maximal value. In other words, a_i is the number of records with the class label that dominates clusters i. Consequently, the clustering is defined as e=1-r. Furthermore, we define the absolute clustering error ace as: ace=e*n.

3. Related Works

Below we present some existing works for handling categorical data as well as data streams.

3.1 STIRR [11]: (Sieving through Iterated Relational Reinforcement) is an algorithm based on non-linear dynamical systems. The database is represented as a graph where each distinct value in the domain of each attribute is represented by a weighted node. Thus if there are N attributes and the domain size of i-th attribute is di. Then the number of nodes is the graph is $\sum_i d_i$. For each tuple is the database, an edge represents a set of nodes which participate in that tuple. Thus a tuple is collection of nodes one from each attribute type. The set of weights of all the nodes define a configuration of this structure. The initial weights of all the nodes can be either assigned uniformly or randomly or by focusing technique. STIRR iteratively changes the configuration by updating weight of any node. The new weight of the node is calculated based on a combiner function, which combines the weights of other nodes participating in any tuple with given node for which the weight to be updated. Thus it moves from one configuration to the other till it reaches a stable point, called as basin. The convergence is dependent on the combiner function. Analyzing the stability is hard for any arbitrary combiner function. However, for simple combination function like sum of multiplication, the system definitely converges to a fixed point. It is easy to see that for categorical attributes, the values which are related through common tuples influence each other during weight modification. Thus one does not require any similarity metric to be defined for categorical attributes. Interestingly, in order to cluster the set of tuples, STIRR maintains multiple copies of weights. When the fixed point is reached, the weight is one or more of the basins isolate two groups of attribute values on each attribute the first with large positive weights and second with small negative weights. The nodes with large positive weights and second with small negative weights are grouped to determine cluster. These groups correspond intuitively to projections of clusters on the attribute. However, the automatic identification of such sets of closely related attribute values from their weights requires a non-trivial post-processing step; such a post-processing step was not addressed in their work. Moreover, the post-processing step will also determine what 'cluster' are output. The underlying idea of STIRR is unique but it may be hard to analyze the stability f the system for any useful combiner function. One requires rigorous experimentation and fine tuning of parameters to arrive at a meaningful clustering[4].

3.2 CACTUS[8]: Catagorical data ClusTering Using Summeries is a sort of subspace clustering. CACTUS attempts to split the database vertically and tries to cluster the set of projection of these tuples to only a pair of attributes. Its basic principle can be described as follows: let us consider two attributes values of two different attributes in the database. Say, ai of attribute type A and aj of attribute type B. there may be tuples where ai and aj co-occur. The support of these two values in the database is the proportion of tuples in which they appear together. If this support exceeds a pre-specified value, we say these values are strongly connected. This concept can be to compute the inter-attribute and intra-attribute summaries of the given data set. Most interesting aspect of these steps are that these can be computed using-attribute and intra-attribute summary. It is not necessary to refer to the original data base. CACTUS first identifies the cluster projections on all pairs of attributes by fixing one attribute. Then it generates an interesting set to represent the cluster projection on this attribute for n-cluster(involving all the attributes). Once all the cluster projections on individual attributes are generated, these are synthesized to get the clusters of the database. The major steps of CACTUS are:
Finding cluster projection on given attribute Ai with respect to another attribute Aj Intersecting all the cluster projection for any given Ai to get the cluster projection Ai with respect to all attribute. Synthesizing the resulting cluster projections to the main clusters.

3.3. COOLCAT [6]: It is capable of efficiently cluster large data sets of records with categorical attributes. COOLCAT's clustering results are very stable for different sample sizes and parameter settings. Also, the criteria for clustering are a very intuitive one, since it is deeply rooted on the well-known notion of entropy. Entropy and Clustering Entropy is the measure of information and uncertainty of a random variable. Formally, if X is a random variable, S(X) the set of values that X can take, and p(x) the probability function of X, the entropy E(X) is defined as shown in Equation:

$$E(X) = -\sum_{x \varepsilon S(X)} p(x) log(p(x))$$

The entropy of a multivariate vector $\bar{x} = \{X_1, \ldots..X_n\}$ can be computed as shown in Equation

$$E(\bar{x}) = -\sum_{X_i \in S(X_i)} p(x_i \ldots x_n) log(x_i \ldots x_n)$$

Entropy is sometimes referred to as a measure of the amount of "disorder" in a system. It is novel method which uses the notion of group records.
COOLCAT initially finds a suitable set of clusters out of a sample, |s| taken from the dataset(|s|<<N, where N is the size of entire dataset). Form the sample dataset it first find k most "dissimilar" records by maximizing the minimum

pairwise entropy of chosen points. That is COOLCAT starts by finding the two points p_{s1} and p_{s2} that maximize E(p_{s1} , p_{s2}) and placing them into two separate clusters(C1, C2) making records. From there it proceeds incrementally .

3.4 ROCK [10]: Robust hierarchical clustering with links uses a novel concept of links to measure the similarity/proximity between a pair of data points. The number of links between two tuples is the number of common neighbors they have in the data set. It is an agglomerative hierarchical clustering algorithm that employs links and not distances when merging clusters. Methods in ROCK naturally extend to non-metric similarity measures that are relevant in situations where a domain expert/similarity table is the only source of knowledge.

Starting with each tuple in its own cluster, the two closet cluster are merged until the required number of clusters are obtained. ROCK, instead of working on the whole data set, clusters a sample randomly drawn from the data set, and then partitions the entire data set on the clusters from the sample. The basic idea of ROCK is based on the following definition.

Neighbor: An object's neighbors are those objects that are considerably similar to it. Let *sim*(O_i , Oj) be similarity function that is normalize and captures the closeness between the pair of objects O_i and O_j. The similarity function *sim* assumes values between 0 and 1. Given a threshold θ(between 0 and 1), a pair of objects O_i and O_j are define as neighbours if *sim*(O_i, O_j)≥θ.

Link: The *link*(O_i , O_j) between the object is define as the number of common neighbors between O_i and O_j . ROCK attempts to maximize the sum of *link*(O_q , O_r) for pairs of objects O_q and O_r belonging to single cluster and at the same time to minimize the sum of the *link*(O_q , O_r) for object pairs belonging to other cluster.

Link between clusters: It is a summation of *links* of all pairs, where each pair is formed by taking one object from each cluster.

Goodness measure: The goodness measure between two cluster is the result obtained after dividing the number of cross-links between the clusters by the expected number of cross-links between the cluster.

The link based adopts the global perspective of the clustering problem. It captures the global knowledge of neighboring data points into the relationship between the individual pair of points. After drawing the random sample from the database, a hierarchical clustering algorithm that employs links is applied to the sample objects. It follows the standard principle of hierarchical clustering. It start with singleton objects as an individual class and progressively merges the two clusters based on the *goodness criteria*, determined by link structure. The merging is continued till one of the following two criteria is met: (1) a specified number of clusters is obtained or, (2) no links remain between the clusters [4].

3.5 CLOPE [7] : Clustering with CLOPE algorithm is developed starting from heuristic method of increasing the height-to-width ratio of the cluster histogram. While being quite effective, CLOPE is very fast and scalable when clustering large transactional databases with high dimensions, such as market basket data and web server logs.

To construct cluster histogram, occurrence of every distinct item is counted for each cluster, and then height (H) and width (W) of the cluster is obtained. For example, cluster {ab, abc, acd} has occurrences of a :3,b :2,c :2 and d:1. Therefore, its corresponding cluster histogram's (H), width (W) and H/W ratio are 2.0, 4.0 and 0.5 respectively.

CLOPE uses a global criterion function that tries to increase the intra-cluster overlapping of transaction items by increasing the height-to-width ratio of the cluster histogram. Moreover, it generalizes the idea by introducing a parameter to tightness of the cluster. Different number of clusters can be obtained by varying parameter. A larger height-to-width ratio of the cluster histogram means better intra-cluster similarity. And, the global criterion function is defined using the geometric properties of the cluster histogram [5].

3.6 SQUEEZER [5]: It is an efficient algorithm for clustering categorical data, Squeezer, which can produce high quality clustering results and at the same time deserve good scalability. The Squeezer algorithm read tuples from dataset one by one. When the first tuple arrives, it form a cluster alone. The consequent tuples are either put into existing cluster or rejected by all existing clusters to form a new cluster by given similarity function define between tuple and cluster.. It is obvious that the squeezer algorithm only makes one scan over the dataset, thus, highly efficient for disk resident datasets where the I/O cost become bottleneck of efficiency. It suitable for clustering data streams, where given a sequence of points, the objective is to maintain consistently good clustering of the sequence so far, using a small amount of memory and time. It can also be handle outliers efficiently and directly in Squeezer. Squeezer achieves both high quality of clustering results and scalability. The summery of Squeezer algorithm as follows:

A novel algorithm for clustering categorical data, Squeezer combines both efficiency and quality of clustering results.

The algorithm is extremely suitable for clustering data streams, where a given sequence of points, the objective is to maintain consistently good clustering of the sequence so far, using a small amount of memory and time. Outlier can be handled efficiently and directly. The algorithm does not require the number of desired clusters as an input parameter.

This is very important since the user usually does not know this number in advance. The only parameter to be specified is the value of similarity between the tuple and cluster, which incorporates the user's exception that how closely the tuples in a cluster should be.

SQUEEZER *Algorithm:*
Inputs :
DS = Categorical Data Stream
St = user defined similarity threshold
Algorithm
for each record t in DS do Begin
for each existing cluster Ci
begin
Measure the similarity between t and Ci
end
Smax = Max(all similarity measures)
Tc = target cluster with Smax
if smax > = St then
Include t in Tc
Update the histogram of Tc
else
Create a new cluster with t
Create its histogram
end if
end

The time and space complexities of the Squeezer algorithm depend on the size of dataset and the number attributes.

4. Motivation

4.1 Unbounded Memory Requirements

Since data streams are potentially unbounded in size, the amount of storage required to compute an exact answer to a data stream query may also grow without bound. While external memory algorithms for handling data sets larger than main memory have been studied, such algorithms are not well suited to data stream applications since they do not support continuous queries and are typically too slow for real-time response. The continuous data stream model is most applicable to problems where timely query responses are important and there are large volumes of data that are being continually produced at a high rate over time. New data is constantly arriving even as the old data is being processed; the amount of computation time per data element must be low, or else the latency of the computation will be too high and the algorithm will not be able to keep pace with the data stream. For this reason, we are interested in algorithms that are able to confine themselves to main memory without accessing disk. In the data stream model of computation, once a data element has been streamed by, it cannot be revisited. A few algorithms have been proposed in recent years for clustering categorical data stream. Squeezer is one of the promising technique, however its performance degrades as the size of the histogram increase. Below we present a new sampling based approach to overcome the bottleneck of Squeezer.

5. FLoMSqueezer: A sampling based approach

The aim was to improve Squeezer to make suitable for clustering categorical data stream and to make less error, reduce the size of histogram and computational time. It is found that if concise sampling technique is applied the new algorithm gives less memory footprint.

5.1 Definition and Notations

1) Similarity Measure: Similarity between an object Y and D is now defined as Sim(H,Y) = SUM(f(hi,yi)/n ; I = 1, 2, …. M Where f(hi,yi) = SUM(fj * (1-distance(vj,yi))

2) Histogram: Histograms are commonly-used summary structures to succinctly capture the distribution of values in a data set. A compact representation of a cluster, H = Histogram of D = $\{h_1,h_2,h_3,\ldots..h_n\}$ where hi = $\{(v_1, f_1), (v_2,f_2),\ldots.(v_p,f_p)$ where p is the no of distinct value of attribute A_i and f_i is the no of objects in D having this value v_i. For every record processed, we have to calculate the similarity with existing clusters, update or create a new histogram, and do the pruning if it's at footprint bound.

3) Cluster error: e = 1 – r where r = SUM(A_i)/n I = 1,2……k where n = total no of data points in the data set and A_i is the no of instances occurring in both the cluster I and its corresponding class. Absolute cluster error ace = e*n.

4) Sampling: Sampling is the most versatile approximation technique available. Most data processing algorithms can be used on a random sample of a data set rather than the original data with little or no modification. Sampling-based algorithms can produce approximate answers that are provably close to the exact answer.

5.2 Processing step of FLoMSqueezer

Sampling techniques have been introduced into the update histogram module of squeezer algorithm. For every incoming tuple **t** in target cluster **Tc,** for each attribute value **t.A** of tuple **t** in target cluster **Tc**. It checks the corresponding dimensions of the histogram tc with a probability of **1/p**. Whether **t.A** is present in the histogram

Tc. If **t.A** present in the histogram of corresponding dimensions, that simply increment the count of attribute value. Otherwise **t.A** remains singleton entry. After every operation on the histogram the histogram for particular dimension is greater some pre-specified Footprint bound "g" then it first raise the value of (**P**) to (**P'**) sampling rate and sample the histogram for that dimensions in terms of **P**'.

FLoMSqueezer Algorithm

Inputs :

DS = Categorical Data Stream
St = user defined similarity threshold
p = initial sampling rate
g = pre specified footprint bound

Algorithm

```
for each record t in DS do
begin
  for each existing cluster Ci
      begin
        Measure the similarity between t and Ci
        end
          Smax = Max(all similarity measures)
          Tc = target cluster with Smax
          if smax > = St then
              Include t in Tc
              for each t.A in t begin
                with a probability (1/p) check
                if t.A is in Histogram for Tc.
                    if t.A is present then
      Increase its count in its <value,count> pair
                    else
                        t.A remain as a sigleton
                    end
                    if S.A >= g
                    then Raise p to p'
          Subject each sample points in S.A to p'
                    end if
                    p = p'
              else
                Create a new cluster with t
                Create its histogram
        end if
end
```

5.3 Analysis

Firstly, It is shown that the FLoMSqueezer algorithm can be proved to have space guarantee on main memory consumed.

1) ***Theorem 1:*** A concise sample method is a uniform random sample of the data set such that values appearing more than once in the sample are represented as <value, frequency> pairs. S = {<v1,f1>,<v2,f2>,……<vj,fj>, vj+1,…..vl} Sample size represented by S = l – j + SUM(fk) k = 1….j. Memory footprint of S=I+j.

2) ***Theorem 2:*** FLoMSqueezer algorithm uses 'g' pre specified footprint bound. If S.A >= g then sampling start. Initial sampling raise p to p' each sample point in S.A to p' it terminates if p=p'.

6. Results and Discussion

6.1 Quality of Clustering with Real-life Datasets

Comparison of FLoMSqueezer with Squeezer is done with real-life Mushroom dataset, which are obtained from UCI Machine Learning Repository and have been tested both algorithms. The Mushroom dataset has 22 attributes with 8124 tuples. Each tuple records physical characteristics of a single mushroom. A classification label of poisonous or edible is provided with each tuple. The numbers of edible and poisonous mushrooms in the dataset are 4208 and 3916, respectively.

6.2 Tuning Parameter

The footprint bound 'g' and initial sampling 'p' rate makes the FLoMSqueezer algorithm differ from the squeezer algorithm. These parameters can affect the results of clustering and speed of the algorithm. In the sequel, an empirical result shows how they can affect the FLoMSqueezer algorithm Mushroom dataset.

Firstly, each histogram Hi is pruned by deleting some entries at bucket boundaries. Apparently, large 'g' results more pruning. When the initial sampling is increased and the similarity was set to 7,8,9,10,11,12,13,14,15,16 respectively, the processing time decreases. This algorithm tend to produce more stable cluster when similarity threshold become larger and footprint bound become low and initial sampling rate increases.

6.3 Scalability Evaluation

1. Scalability with Synthetic Dataset: The stream size (i.e number of rows), the number of attributes and number of classes are major parameters in the synthetic categorical data streams. The experiments were carried out with synthetic datasets. The scalability of the algorithm is determine by sample size, by taking dataset with 6 attributes and 8 tuples then the performance is same in small dataset. But Squeezer algorithm deteriorates with increase of database size.

2. Scalability with Real Dataset: The experiment is carried out with real-life dataset the Mushroom dataset [9], which are obtained from UCI Machine Learning Repository. Both the algorithms clusters based on the threshold value, initial sampling rate, footprint bound are taken. Table shows that comparison between time, histogram clustering errors and number of clusters produce by two algorithms. The proposed algorithm performs better then Squeezer algorithm in handling histogram size, time and clustering error.

7. Experimental Results

A comprehensive performance study has been conducted to evaluate above method. Both the quality of the clustering results and the efficiency are examined. Synthetic dataset and Real-life dataset are used to evaluate the quality of clustering results. The experiments were divided into three parts:

1) Firstly we tested with execution time on the data stream of different sizes.

2) Secondly, we studied the effect the histogram size.

3) Thirdly, we demonstrated the changes of error value, comparing error value.

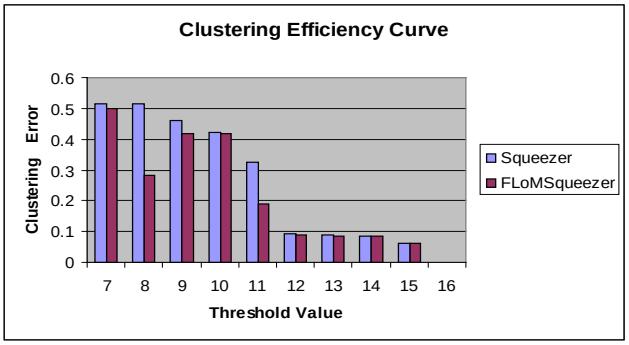

Fig. 1 Clustering result with error comparison

The relative performance of the two algorithms with clustering errors and threshold value shows FLoMSqueezer algorithm out performed Squeezer algorithm.

TABLE I: Relative performance of Squeezer and FLoMSqueezer

Ranking	Average clustering Error
Squeezer	0.256696206
FLoMSqueezer	0.213429346

TABLE I : Histogram and time comparison

Threshold	Squeezer			FLoMSqueezer		
	Histogram Size	Time in seconds	No. of clusters	Histogram Size	Time in seconds	No. of Clusters
7	183	1	2	140	1	2
8	237	1	3	299	1	5
9	289	1	4	279	1	5
10	357	1	6	333	1	7
11	401	2	8	450	2	10
12	583	4	14	556	3	14
13	550	4	15	542	3	15
14	598	5	17	583	3	17
15	637	5	20	633	4	20
16	697	7	23	697	4	23

Fig. 2 shows the results of histogram comparisons between Squeezer and FLoMSqueezer algorithm. Out of 10 different threshold values FLoMSqueezer outperform Squeezer algorithms. It performs best in 6 cases, shows less performs in two cases and perform equal in 2 cases. It never performed worst in other cases.

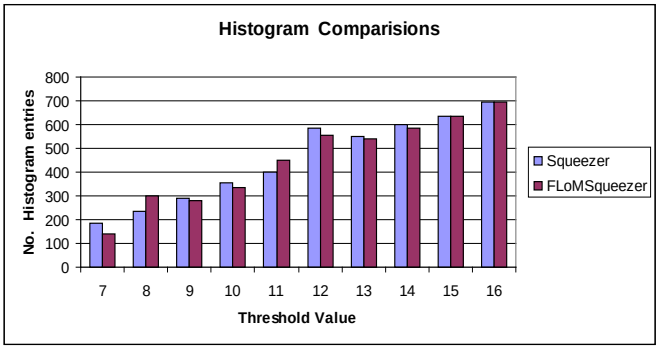

Fig. 2 Comparisons of histogram with different threshold values.

It implies that the FLoMSqueezer can produce good clustering output with limited memory in the data stream environment.

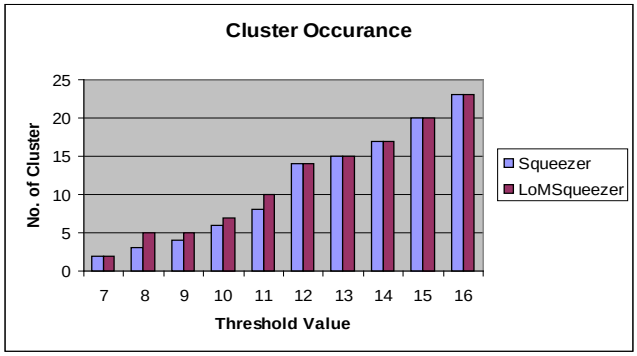

Fig. 3 Comparison of cluster occurrences

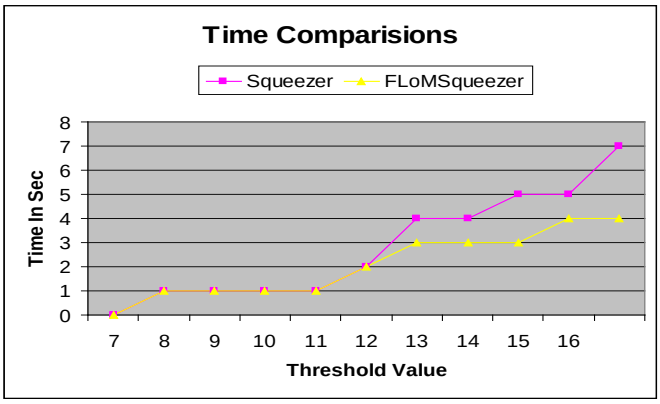

Fig.4 The execution time with different threshold values

The result of Fig.4 shows that the choice of st(threshold value) can affect both the quality of clustering and execution time for FLoMSqueezer. Thus, choosing a proper parameter value is one of the important tasks that must be considered in the FLoMSqueezer algorithm. The threshold St value footprint bound 'g' and initial sampling rate controls the decision to merge a new tuple into an existing cluster, or to place it in a cluster by itself. A good choice for 'g', p , and St is necessary to produce a concise and useful summarization of the data set.

8. Conclusions

An efficient algorithm, FLoMSqueezer is proposed, which produces high quality of clusters from categorical data streams using sampling technique. The performance of algorithm is tested with synthetic dataset and real-life dataset and found effective in terms of handling large stream of categorical data.

References

[1] Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "*Models and issues in data stream systems*". In Proc of PODS, 2002.

[2] Ding Q and Perrizo W, "*Decision Tree Classification of Spatial Data Streams Using Peano Count Trees*", Proc of the ACM Symposium on Applied Computing, Madrid, Spain, March 2002.

[3] Hand D J, "*Statistics and Data Mining: Intersecting Discipline*", ACM SIGKDD Explorations, 1, 1, pp. 16-19, June 1999.

[4] Golab L and Ozsu M. "*Issues in data stream management*". In SIGMOD Record, Vol. 32, No.2, pages 5–14, June 2003.

[5] He Z, Xu X and Deng S, "*Squeezer: an efficient algorithm for clustering categorical data*", Journal of Computer Science & Technology, 17(5), pp.611-624(2002).

[6] Yi Li, Barbara D, Couto J, "*Coolcat: An entropy-based algorithm for categorical Clustering*", In Proc of CIKM'02, pp. 582-589, 2002.

[7] You J, Yang Y, Guan X,"*Clope: A fast and effective clustering algorithm for transactional data*". In Proc of SIGKDD'02, Edmonton, Canada, 2002.

[8] Ganti V, Gehrke J, Ramakrishnan R. "*CACTUS –clustering categorical Using Summeries*" In Proc. of ACM SIGKDD, San Diego, California, USA, pp. 73—83, 1999.

[9] UCI Repository of Machine Learning Databases. (http://www.ics.uci.edu/~mlearn/MLRRepository.html)

[10] Guha S, Rastogi R and Shim K, "ROCK: A robust clustering algorithm for categorical attributes", Proc of the IEEE Intl Conf on Data Engineering, Sydney, March 1999.

[11] Gibson D, Kleiberg J, Raghavan P. "Clustering categorical data: An approach based on dynamic systems". In Proc. Int. Conf. Very Large Databases, New York, August, 1998, pp.311-322.

[12] G.S Maku and R.Motwani, "Approximate frequency count over data stream", Proceeding of 28th VLDB conference, Hong kong, China, 2002.

Marpe Sora obtained B.Tech form NERIST and M.Tech from Tezpur University. Presently he is Assistant professor in Rajiv Gandhi University, Department of Computer Science and Engineering Arunachal Pradesh. His main research interests include signal and speech processing and Data mining.

Swarup Roy did his M.Tech. in Information Technology and pursuing his Ph.D in Comp Sc & Engg. from Tezpur University. Presently he is an Assistant Professor in the department of Information Technology at North Eastern Hill University, Shillong. He is a recipient of university gold medal for securing first position in M.Tech. His research interest includes Data mining and Computational Biology. S Roy has published a number of papers in different International Journals and refereed Int'l. Conf. Proceedings and authored a book. He is a reviewer of few International Journals.

Sarangthem Ibotombi Singh obtained MCA from Manipur University. He is presently working as Assistant Professor in the Department of Computer Science & Engineering at Tezpur University. His current area of interest is Data mining, Spatial Database and Web Services.