Detecting the Change of Clustering Structure in Categorical Data Streams

Keke Chen * Ling Liu †

1 Introduction

With the deployment of wide-area sensor systems and Internet-based continuous-query applications, processing stream data has become a critical task. As an important method in data analysis, recently clustering has attracted more and more attention in analyzing and monitoring streaming data [5, 1]. The initial research has shown that clustering stream data can provide important clues about new emerging data patterns so that the decision makers can predict the coming events and react in near real time. Stream data clustering is especially important to the time-critical areas such as disaster monitoring, anti-terrorism, and network intrusion detection. As many of such applications also include a large amount of categorical data, clustering the categorical data streams becomes an interesting and challenging problem. Surprisingly, very few [2] have addressed the problems related to clustering categorical data streams.

The change of critical clustering structure in data streams provides the most important information, which includes three forms: new emerging clusters, drifting cluster centers that is caused by expanding clusters, and disappearing clusters that is caused by the convergence of growing clusters. The latter two aspects can be effectively indicated by the change of the "Best K" number of clusters. To our knowledge, none has addressed the problem of monitoring the change of clustering structure for categorical data streams.

We have developed the BKPlot method and ACE algorithm [3] for identifying the critical clustering structure in large static categorical datasets. However, identifying critical clustering structures in data streams presents particular challenges, primarily in fast processing time and restricted memory consumption. In this paper, we propose a framework for monitoring the change of critical clustering structure in categorical data streams. The key idea is based on the summarization tree structure, called Hierarchical Entropy Tree (HE-Tree for short), and the extended ACE algorithm working on the HE-Tree structure. HE-Tree utilizes a small amount of memory to summarize the entropy property of the data streams, and groups the data records into a bunch of sub-clusters located at the HE-Tree leaf nodes. The extended ACE algorithm is able to handle the snapshot sub-clusters (often a few hundreds) and generate approximate snapshot BKPlots for identifying the Best K at certain time interval. The difference between the clustering structures can be conveniently identified by comparing the distinctive points on the snapshot BKPlots.

*Georgia Institute of Technology, kekechen@cc.gatech.edu

†Georgia Institute of Technology, lingliu@cc.gatech.edu. This research is partially supported by NSF CNS, NSF CCR, NSF ITR, DoE SciDAC, DARPA, CERCS Research Grant, IBM Faculty Award, IBM SUR grant, HP Equipment Grant, and LLNL LDRD

2 Entropy-based Categorical Clustering

The framework is based on entropy-based categorical clustering. Due to the space limitation, please refer to the paper [3] for the detailed background introduction. Below, we give the necessary notations and definitions, and the important metric *Incremental Entropy*, which is used in HE-Tree and the extended ACE algorithm.

Notations and Definitions Consider that a dataset $\mathbb{S}$ with N records and d columns, is a sample set of the discrete random vector $X = (x_1, x_2, \ldots, x_d)$. For each component x_j, $1 \leqslant j \leqslant d$, x_j takes a value from the domain A_j. A_j is conceptually different from $A_k (k \neq j)$. There are a finite number of distinct categorical values in $domain(A_j)$ and we denote the number of distinct values as $|A_j|$. Let $p(x_j = v)$, $v \in A_j$, represent the probability of $x_j = v$, we define the estimated entropy of the dataset $\mathbb{S}$ as follows. $\hat{H}(X) = H(X|\mathbb{S}) = -\sum_{j=1}^{d} \sum_{v \in A_j} p(x_j = v|\mathbb{S}) \log_2 p(x_j = v|\mathbb{S})$

Let $C^K = \{C_1, C_2, \ldots, C_K\}$ represent a partition of dataset $\mathbb{S}$ and n_k be the number of records in the cluster C_k. It has been shown [6, 2] that categorical clustering is equivalent to minimizing the item $\frac{1}{dN} \sum_{k=1}^{K} n_k \hat{H}(C_k)$, which is named as the "expected entropy" of partition C^K, and notated as $\bar{H}(C^K)$.

Incremental Entropy While expected-entropy describes the average intra-cluster quality, incremental entropy is a measure used to describe the similarity be-

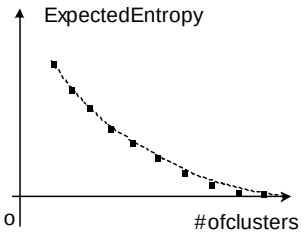

Figure 1: Sketch of expected entropy curve.

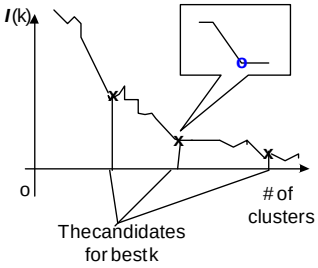

Figure 2: Sketch of ECG graph.

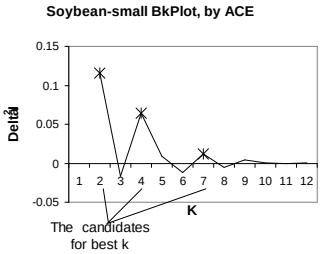

Figure 3: Finding the best k with BKPlot.

tween any two clusters [3]. Intuitively, merging the two clusters that are similar in the inherent structure will not increase the disorderliness (expected-entropy) of the partition, while merging dissimilar ones will inevitably bring larger disorderliness. Therefore, this increase of expected entropy has some correlation with the similarity between clusters. Let $C_p \cup C_q$ represent the mergence of two clusters C_p and C_q, and C_p and C_q have n_p and n_q members, respectively. We name $I_m(C_p, C_q) = (n_p + n_q)\hat{H}(C_p \cup C_q) - (n_p\hat{H}(C_p) + n_q\hat{H}(C_q)) \geqslant 0$ as the "*Incremental Entropy* (IE)" of merging the clusters C_p and C_q. Note that $I_m(C_p, C_q)$ is always non-negative [3] and $I_m(C_p, C_q) = 0$ suggests that the two clusters have the *identical structure* regardless of the cluster size. IE plays an important role in constructing a hierarchical clustering scheme, where minimizing IE measure is equivalent to minimizing the expected-entropy criterion.

BKPlot method Let the neighboring partitions be two clustering schemes having K and $K+1$ number of clusters, respectively. The basic idea of BKPlot is to investigate the entropy difference between any two optimal neighboring partitions and to find the characteristics that are related to the critical clustering structure. Let the expected-entropy of the optimal partition be $\bar{H}_{opt}(C^K) = \min\{\bar{H}_i(C^K)\}$, where $\bar{H}_i(C^K)$ be any possible K-cluster partitions. We define the increasing rate of entropy between the optimal neighboring partitions as $I(K) = \bar{H}_{opt}(C^K) - \bar{H}_{opt}(C^{K+1})$. The important result [3] is that $I(K)$ curve (the Entropy Characteristic Graph (ECG), Figure 2) implies two levels of difference between the neighboring partitions, and this is used to understand where the critical clustering structure emerges. An ECG shows that the similar partition schemes with different K are at the same "plateau". From plateau to plateau there are the critical points implying the significant change of clustering structure. These critical points are highlighted in the second-order differential of ECG, which is named as "Best-K Plot (BKPlot)".

Exact BKPlots cannot be achieved in practice, since $I(K)$ is based on the optimal K-cluster scheme which involves entropy minimization. However, since we only pay attention to the peak/valley points, approximate but accurate BKPlots are possible to get. A hierarchical clustering algorithm ACE in [3] is proposed to generate high-quality BKPlots. However, we can not apply ACE algorithm directly to data streams due to its high complexity.

3 HE-Tree: Summarizing Clustering Structure of Categorical Data Streams

In this section, we design the summarization structure – Hierarchical Entropy Tree (HE-Tree). The basic idea of HE-Tree is to coarsely but rapidly assign the records from the data stream onto hundreds of subclusters. These subclusters will be enough to give us an accurate sketch of the clustering structure. HE-Tree can effectively summarize the old records and adapt to the new coming records. Incremental Entropy metric plays an important role in the summarization process. A HE-Tree consists of two key components:

1. *HE-node structure*, which summarizes the entropy characteristics of a group of records and facilitate fast processing of stream data items;

2. *Incremental-Entropy based lookup/assigning algorithm*, which helps to adapt the changing clustering structure.

Given the fixed height h and fanout f, HE-Tree is constructed in two stages:

1. *the growing stage*, which grows the tree to full size;

2. *the absorbing stage*, which absorb the new coming items into the leaf nodes of the full tree.

Summary Table. Since computing cluster entropy is based on counting the occurrences of categorical values

in each column, summary table is used to keep the counters for each cluster. For each categorical value $v_{ij} \in A_j$, we have an element $T[v_{ij}]$ in summary table as the counter.

Nodes in HE-Tree. HE-Tree is a balanced tree similar to B-tree, where each node has f entries and the entries in the leaf nodes represents the n_c subclusters. As shown in Figure 4, each entry in leaf node contains a summary table, and a leaf node also contains a I_m table (Incremental Entropy table) with $(f+1)^2$ entries and a heap in size of f for fast locating and merging the entries. I_m table, together with the heap, keeps track of the minimum I_m. An internal node (non-leaf) in the tree contains only the aggregation information of its child nodes.

Let a summary table represented with a vector $\vec{s}$ and the entropy characteristic of any internal node C_i denoted as $EC_i(n_i, \vec{s_i})$, where n_i is the number of records summarized by this node. Let C_{ij}, $1 \leqslant j \leqslant f$ represent the child nodes of C_i. HE-Tree maintains the following property.

$$EC_i(n, \vec{s}) = \sum_{j=1}^{f} EC_{ij}(n_{ij}, \vec{s_{ij}}) = EC_i(\sum_{j=1}^{f} n_{ij}, \sum_{j=1}^{f} \vec{s_{ij}})$$

The key of HE-Tree is to approximately minimize the overall expected entropy by locally minimizing the expected entropy of the selected branch $\bar{H}(C_i^f)$ in each update of the tree.

Growing Phase. The tree grows until the number of leaf nodes reaches $\lceil n_c/f \rceil$. The first subroutine is for locating the target leaf node for insertion. The search begins at the root node. Let e denote the inserted record and e_i denote one of the entry in current node. Since each entry in the internal node is the summarization of its sub-tree, we can find the most similar entry to e by finding the minimum I_m among $I_m(e, e_i), i = 1..f$, i.e.

$$e_t = argmin_{e_i}\{I_m(e, e_i), i = 1..f\}$$

Iteratively, the same criterion is applied to the selected child node until a leaf node is reached. If the target leaf node has empty entries and $I_m(e, e_i) \neq 0$ for each occupied entry e_i, the record is assigned an empty entry. Otherwise, the new record is merged to the identical entry.

When the target leaf node is full, a split operation is applied. In split algorithm, we partition the entries into two groups. First, two pivot entries (e_r, e_s) are found in the target node that lead to maximum I_m among all possible pairs, i.e., they are the most dissimilar pair.

$$(e_r, e_s) = argmax_{e_r, e_s}\{I_m(e_r, e_s), i = 1..f\}$$

The rest entries are then assigned to the two clusters so that the overall expected-entropy of the partition keeps minimized. A new node is generated accommodating one of the two sets of entries, while a new entry is added into the parent node pointing to the new node. The insertion/splitting continues until the number of leaf entries reaches n_c, and then the growing phase is turned to the absorbing phase.

Absorbing Phase. In this phase, the same locating algorithm is applied to locate the target leaf node for the new record. Instead of insertion in the leaf node, we merge the most similar two items among the f+1 items – the f entries in the leaf node plus the new record. This allows the tree to rapidly adapt to the change of clustering structure in the leaf level. When a new record comes, only f calculations of incremental entropy are needed to update the I_m table and the heap, which are then used to select the most similar two to merge.

3.1 Setting of Parameters

The setting of the two parameters f and n_c can affect the efficiency and quality of summarization. Let h be the height of the tree (root is at level 1). For simplicity, we always construct full trees and allow $n_c = f^h$ to vary from hundreds to thousands. In experiment, we show that a smaller f always results in faster summarization, but can undermine the quality of summarization when the clustering structure is changing dramatically. On the other hand, larger f increases the ability adapting to the change of clustering structure but also increases the cost in the absorbing phase. To tradeoff the performance and robustness, we can set the tree to be 2 ∼3 layers, with $f = 10 \sim 20$.

3.2 Complexity of HE-Tree

The time complexity of constructing HE-Tree can be divided into two phases. In the growing phase, about f^h records are inserted into the tree and each record needs at most $O(hf)$ comparison to locate the target node. In the absorption phase, besides the cost of locating, each insertion also costs $O(f)$ incremental-entropy calculation that costs O(dm). Therefore, the cost will be $O((h+dm)f)$. With the fixed setting of small f and h, the total cost is only dominated by the factor dm. Similarly, we can infer that a HE-Tree needs $O((dm+f)f^{h+1})$ memory, which is also determined by dm.

4 Framework for Monitoring the Change of Clustering Structure

With the HE-Tree and the extended ACE algorithm, which clusters only the sub-clusters at the HE-Tree leaf entries, we can precisely monitor the change of clustering structure in the categorical data stream. The

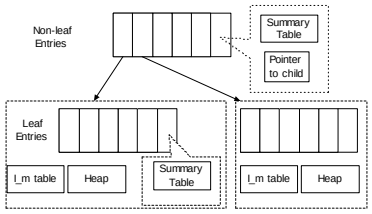

Figure 4: The structure of HE-Tree.

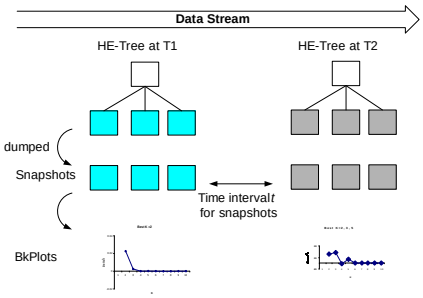

Figure 5: Framework for Monitoring Categorial Data Streams.

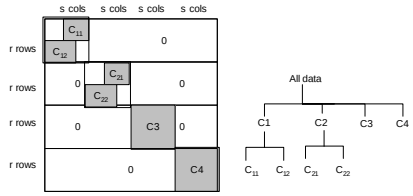

Figure 6: Clustering structure of DS1

framework is illustrated in Figure 5. The working mechanism can be described as follows.

1. The records from the data stream are inserted into the HE-Tree sequentially.
2. At certain time interval Δt, the summary tables in the leaf nodes are dumped out (to a piece of memory or to hard disk).
3. the extended ACE algorithm are performed on the snapshot as soon as it is dumped, the result of which generates a BKPlot.

The consecutive BKPlots are analyzed to see the difference between the clustering structures. BKPlots can be represented as a function $B(K)$, where K is the number of clusters and the distinctive $B(K)$s indicate the candidate best Ks. Without loss of generality, we suppose the first κ distinctive Ks on BKPlots are $\Gamma = \{k_1, k_2, \ldots k_\kappa\}$. Let Γ^{old} and Γ^{new} represent two set of Ks on the consecutive BKPlots, respectively. There are two kinds of important differences we need to notice.

1. If $\Gamma^{old} \neq \Gamma^{new}$, the clustering structure is dramatically changed and we need to analyze the snapshot of Γ^{new} in detail.
2. If $\Gamma^{old} = \Gamma^{new}$, but at certain k_i that $|B(k_i^{new}) - B(k_i^{old})| > \theta$, where θ is a threshold raising an alarm, which indicates some minor change of clustering structure.

5 Experimental Results

The goal of the experiments is two-fold. 1) We investigate the parameter setting of HE-Tree and give the estimate of appropriate settings; 2) We want to show that HE-Tree summarization together with the extended ACE algorithm can provide high-quality monitoring.

Datasets The synthetic dataset DS1 has a two-layered clustering structure (Figure 6) with 30 attributes and N rows. It has four same-sized clusters in the top layer. Each cluster has random categorical values selected from {'0','1','2','3','4', '5'} in some distinct set of attributes (the dark area in Figure 6), while the rest attributes are set to '0'. Two of the four clusters also have clustering structure in the second layer. This dataset is primarily used in exploring the effect of the parameters of HE-Tree to summarization. We also use a real discretized dataset, "US Census 1990 Data ", which has 68 dimensions and about 2 million records. The visualization of its sample dataset is shown in Figure 11.

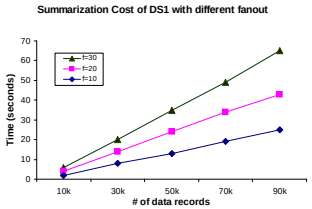

Figure 7: Cost of HE-Tree summarization with different fanout f.

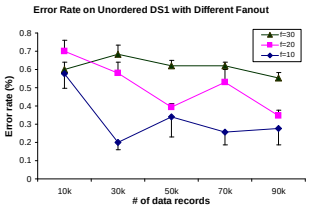

Figure 8: Error rate on unordered records

5.1 Parameter Setting for HE-Tree For simplicity, we always use full trees in the experiment. Intuitively, for a fixed f, the higher the tree , the finer the granularity of summarization will be delivered. Since we often care about the clustering structures with less than 20 clusters, a short tree with less than one thousand subclusters is enough for achieving a high-quality BKPlot. Thus, we fix $h = 2$ with varying fan-out f from 10 to 30 in experiments. A set of datasets (20 datasets) in the same structure shown in Figure 6 are generated and the result is based on the 20 runs of different datasets.

Figure 7 shows the linear complexity of HE-Tree summarization, which is consistent with our analysis. Figure 8 shows the effect of different settings to the qual-

ity of clustering result for "Unordered DS1". Unordered DS1 randomly orders the records from different clusters. Thus, any considerably long segment of the unordered DS1 stream contains the primary clustering structure shown in Figure 6. The result shows some variances between the error rates for different f, but overall the error rates are similar and low, which is consistent with the clustering structure of the data stream.

"Ordered DS1" shows a more interesting scenario, where the clustering structure is dramatically changed in the stream. We observed that the setting of f may significantly affect the quality of monitoring. Figure 9 shows the result of sequentially processing the clusters C_{11} to C_4. A tree with larger f is more adaptive to the change of clustering structure. It shows that increasing f from 10 to 20 can considerably reduce the error, but $f = 30$ will not significantly improve the result of $f = 20$. Balanced with the time cost and the robustness, $f = 20$ seems the best for efficiently adapting the change of structure.

5.2 Robustness of HE-Tree/Extended ACE We run the experiment on the real US Census data. A small sample set of 500 records is used to show the original clustering structure, and two large sample sets with 10K and 100K, respectively, simulates two data streams. The sample sets are uniformly drawn from the original dataset, therefore, the clustering structure is preserved. All BKPlots (Figure 10) strongly suggest the best K is 3, while K=2 is probably another candidate, which is consistent with the visualized clustering structure in the paper [4]. The result shows that HE-Tree summarization can preserve the primary clustering structure and thus HE-Tree combined with the extended ACE method is a robust approach for monitoring the change of clustering structure.

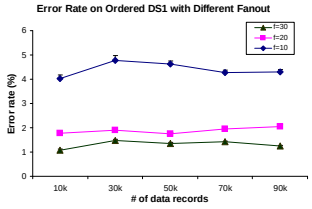

Figure 9: Error rate on ordered records

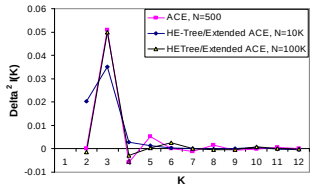

Figure 10: BKPlots for Census data.

5.3 Monitoring the Changes We also demonstrate the progressive monitoring results of the data stream Census-stream. We partition the census dataset into four parts and mix the parts sequentially so that the special clustering structures appear in different stages as

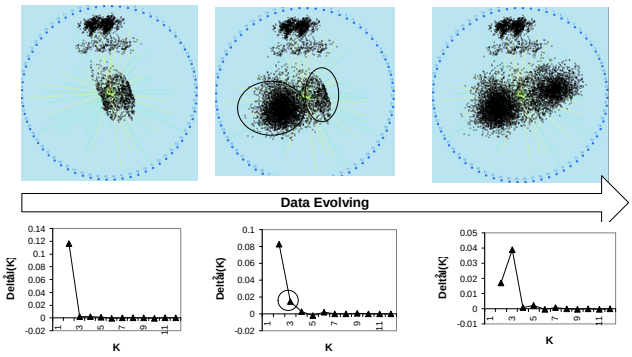

Figure 11: Monitoring Census-stream.

Figure 11 shows. At first snapshot, there are clearly two clusters; in the second one, the third cluster emerges; finally, a two-layer clustering structure (K=2 and 3) appears in the third snapshot.

6 Conclusion

In this paper, we address the problem of detecting the change of clustering structure in categorical data streams, with the combination of the BKPlot method and the Hierarchical Entropy Tree (HE-Tree) algorithm. HE-Tree summarizes the stream clustering structure into a small number of leaf nodes. In order to observe the change of clustering structure, snapshots of the leaf entries of HE-Tree are dumped in certain time interval, which is then processed by the extended ACE clustering algorithm to generate high-quality BKPlots. With the snapshot BKPlots we can conveniently identify whether and how the clustering structure in the stream is changed.

References

[1] Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. A framework for clustering evolving data streams. *Proc. of Very Large Databases Conference (VLDB)* (2004).

[2] Barbara, D., Li, Y., and Couto, J. Coolcat: an entropy-based algorithm for categorical clustering. *Proc. of ACM Conf. on Information and Knowledge Mgt. (CIKM)* (2002).

[3] Chen, K., and Liu, L. The "best k" for entropy-based categorical clustering. *Proc. of Intl. Conf. on Scientific and Statistical Database Management (SSDBM)* (2005).

[4] Chen, K., and Liu, L. iVIBRATE: Interactive visualization based framework for clustering large datasets. *To appear in ACM Transaction on Information Systems* (2006).

[5] Guha, S., Meyerson, A., Mishra, N., Motwani, R., and O'Callaghan, L. Clustering data streams: Theory and practice. *IEEE Trans. on Knowledge and Data Eng. 15* (2003).

[6] Li, T., Ma, S., and Ogihara, M. Entropy-based criterion in categorical clustering. *Proc. of Intl. Conf. on Machine Learning (ICML)* (2004).