DOUBLE CM SKETCH: A BRIEF INTRODUCTION AND THEORETICAL ANALYSIS

Zhiwei Ying
521030910376
yingzhiwei@sjtu.edu.cn

Wenjian Sun
521030910392
sun.wen.jian@sjtu.edu.cn

Jingyu Xie
521030910417
xiejy927@sjtu.edu.cn

ABSTRACT

The streaming algorithm is a hot topic in data mining. In this project, we focus on the periodic property of the input data stream and propose a new structure- Double CM Sketch, which is an online algorithm to figure out the most frequent periodic items. We also shed light on the practical significance and application prospects of our work results.

Keywords Double CM Sketch · Periodic items · Sparse analysis

1 Introduction

Streaming algorithm is a rising topic in the field of data mining and online algorithms. We have learned several streaming algorithms in class, such as Dgim, etc., most of which are based on sampling. However, in some domains of data compression, the sampling approach cannot achieve the best result. For example, in an election counting problem, though the sampling method can get good results with high probability, simply maintaining the counting result will function most precisely.

In this project, we focus on the periodic items of a data stream and develop a novel method to count the occurrence of each periodic item. The algorithm functions well in experiments. We discuss the performance of our method and the input dataset and give a theoretical analysis at the end of the report.

2 Preliminary

Stream Model Given a data stream $S = (e_1, e_2, e_3, \dots, e_i, \dots)$, given an item e, let t_i be the ith appearance of e. Then e has many intervals, where interval V_i is $t_{i+1} - t_i$.

Periodic Items For each item e, if one of its intervals occurs multiple times, we consider it as a periodic item. Periodic items have two aspects: interval and frequency. The frequency is defined as the number of arriving intervals that fall in the range $[V - \Delta T, V + \Delta T]$. We also find that in [1], this kind of data is called *simplex items*.

Then the formal definition of our problem is given below: given a data stream, the problem is to count periodic items with the most precision, especially for the K largest ones. For example, given a data stream $S = (a, b, a, b, d, d, a, c, a, c, e, e, a, b, a, b)$, suppose they arrive at the constant speed and $V = 2, \Delta T = 0$ for convenience. Then we obtain the periodic items as follows. Top 1: $(a, 2)$ occurs 3 times. Top 2: $(a, 4)$ occurs twice. Top 2: $(b, 2)$ occurs twice.

3 Method: Double-CM Sketch

We use two structures to achieve our goal, a cover-min sketch [2] and a count-min sketch [3]. Since the cover-min sketch is indeed a variant of the count-min sketch, we introduce the count-min sketch first, though in our algorithm, the order is reversed.

3.1 Count-Min Sketch

Naturally, the idea of the count-min sketch comes out that if we have limited space and then meet collisions, we can ignore them and just store in each hash cell the total number of elements that hash there. A simple analysis is that a cell containing the true count may contain other colliding elements. And any other element has only $\frac{1}{B}$ chance of colliding in this cell, where B is the space of the hash table. Therefore, the expected error is at most $\|x\|_1 / B$, where x is the ground truth vector. Undoubtedly, some elements will have much higher errors than expected. Thus, we can repeat the process several times to get the minimum estimate. You can find details in 1.

Algorithm 1: Count Min Sketch (Space Size B, # of Round R)

Pick R pairwise independent hash functions $h_1, \ldots, h_R : [n] \to [B]$
$y_i^{(r)} \leftarrow 0 \quad \forall i \in [B], r \in [R]$
for u in Stream **do**
  for $r \in [R]$ **do**
    $y^{(r)}_{h_r(u)} += 1$
  end for
end for
for $u \in [n]$ **do**
  $\hat{x}_u \leftarrow \min_{r \in [R]} y_{h_r(u)}$
end for
return $\hat{x}_u$

However, we cannot use the original version straight to our problem. To apply this structure in our framework, the hash function should be modified to 2D, since the output of the cover-min sketch is a 2D-tuple.

3.2 Cover-Min Sketch

The cover-min sketch is just a variant of the count-min sketch. The only difference is that the cover-min sketch records a timestamp when each item comes in, rather than just adding one to the count. The process can be regarded as two parts. In the insertion part, when inputting an item e with timestamp t, the item is mapped to one bucket in each of the d arrays using d hash functions. The Cover-Min sketch then updates the timestamp in each corresponding bucket to the current time. In the report part, for an item e and its timestamp t, we calculate the d hash functions to find the d buckets and extract the timestamps. By subtracting the smallest timestamp from the current time, we obtain the time interval $V = t_{current} - \min t_{before}$ and report it with e. The algorithm details are in 2.

3.3 Double CM Sketch

The double-CM sketch includes a cover-min sketch and a count-min sketch. A sketch is shown in Figure 1. The cover-min sketch records and reports the interval for each incoming item, while the count-min sketch keeps the number of periodic items with little space complexity. Given a timestamp t for an item e, we insert it into the cover-min sketch to get the interval V. We then form an element $E = (e, V)$ and insert it into the count-min sketch. Note that the same item ID with different intervals is treated as different elements and the hash mapping in count-min is 2D-mapping. Then we can calculate a value for each element with space complexity as small as possible, and the results are accurate with high probability.

4 Experiment Result

We test the performance of our new sketch in 2 dimensions.

Algorithm 2: Cover Min Sketch (Space Size B, # of Round R)

Input a coming item e with the timestamp t
Output the time interval V of e
Pick R pairwise independent hash functions $h_1, \ldots, h_R : [n] \to [B]$
$y_i^{(r)} \leftarrow 0 \quad \forall i \in [B], r \in [R]$
for u in Stream **do**
  result$\leftarrow \infty$
  for $r \in [R]$ **do**
    result$\leftarrow \min\{\text{result}, y_{h_r(u)}^{(r)}\}$
    $y_{h_r(u)}^{(r)} \leftarrow t$
  end for
  Output $\{u, \text{result}\}$
end for

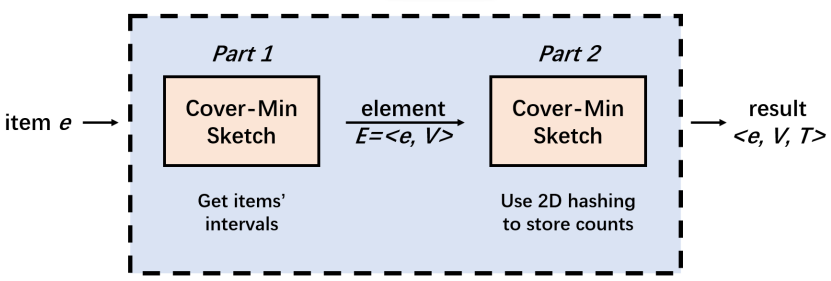

Figure 1: Double CM Sketch Map

4.1 Experiment on the Real-World Data

Based on the original paper about Periodic Sketch [2], we conducted two attempts: replacing the GSU sketch with the cover-min sketch and count-min sketch respectively. We found that the latter achieved better results, and ultimately became our research outcome. The experiment result Figure 2 on the dataset MACCDC [4] shows that, increasing *num_hash* results in higher errors, lower recall rates, and fluctuating precision. The algorithm misses more frequent items with more hash functions, reducing its pattern recall ability. Although it may still identify some items accurately, too many hash functions increase false positives. Thus, optimizing *num_hash* based on application needs and dataset characteristics is crucial.

4.2 Experiment on Generated Data of Different Sparsity

First, we generate some integer sequences with different sparsity. The probability distributions of different data are shown in Figure 3. The sequences are of length $1e6$.

We test our algorithm with different hyper-parameters such as the round and size of both two CM sketches on these different inputs. We demonstrate that our Double-CM algorithm can accurately identify frequently occurring periodic patterns in data using very small space. We also find that for sparse data, our algorithm performs exceptionally well, while for more uniform data without prominent periodic patterns, its performance is average due to collisions. The results are in Figure 4.

5 Theoretical Analysis on Sparsity of Input Data

First, some basic definitions are given here. A vector x is **k-sparse** if the number of non-zero entries of x is k. And if the entry of x is not zero but so small that we can regard it as 0 and the number of meaningful entries is k (x is close to a k-sparse vector), we say that x is **approximately k-sparse**. Another technique is that we use $H_k(x)$ to denote the k-sparse vector of x in $\mathbb{R}^n$ that sets all but the largest k entries of x to zero.

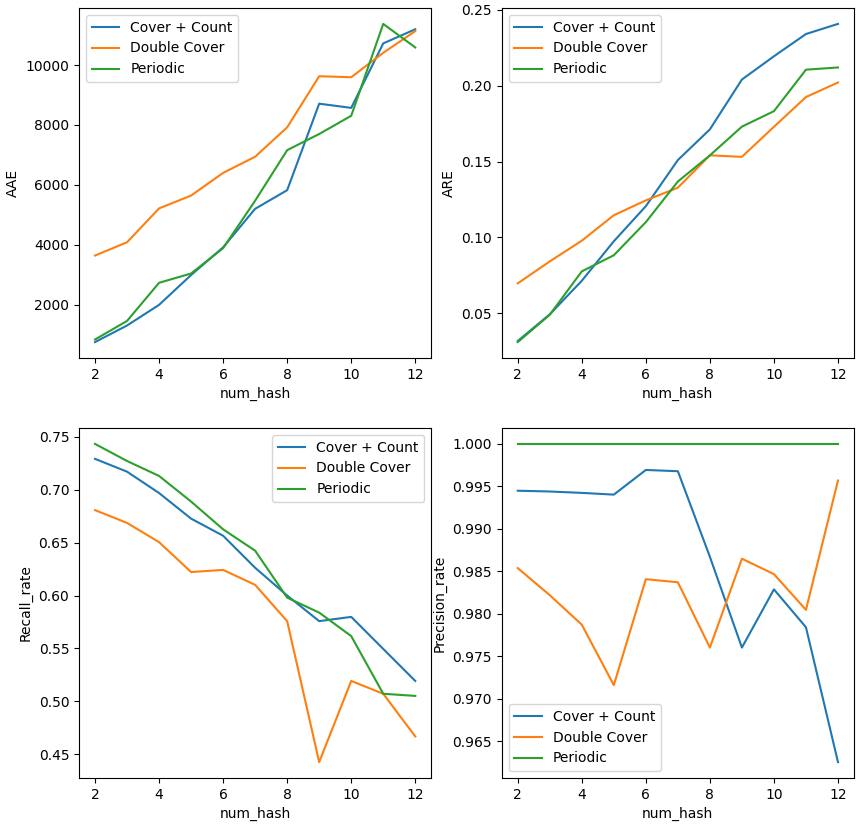

Figure 2: Result on the MACCDC dataset

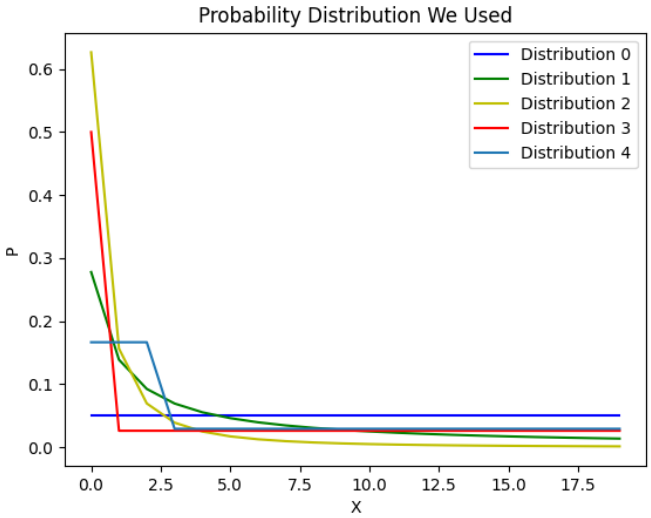

Figure 3: Probability distributions of different generated data

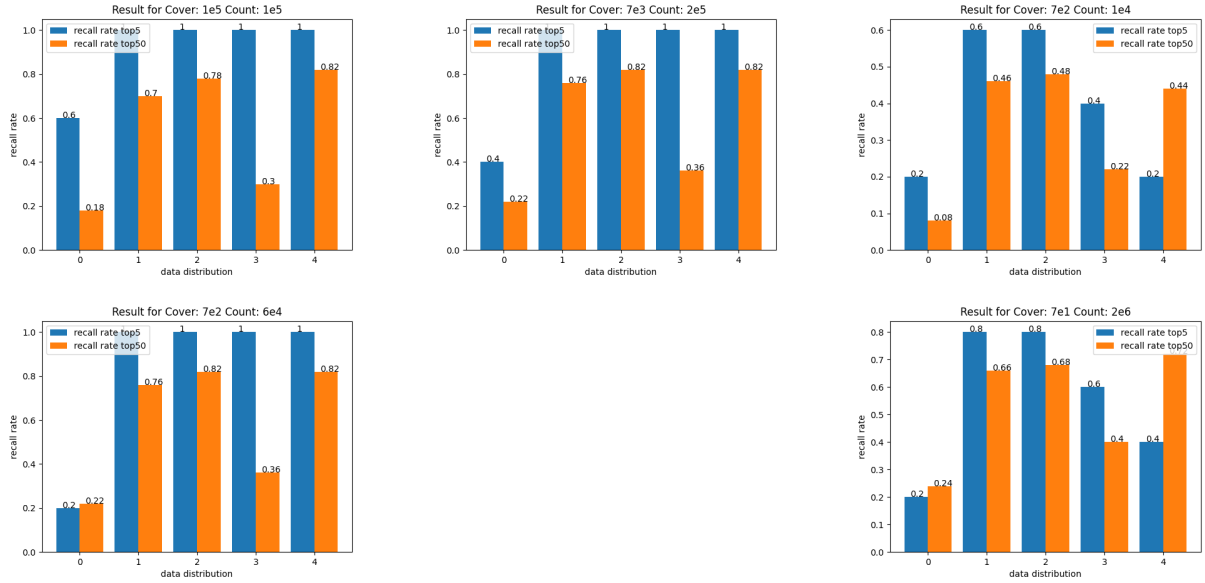

Figure 4: Results of different hyper-parameters on different data

Then for Count-Min Sketch, if the input data is k-sparse, $B \geq 4k$ and $R \geq 2\log_2 n$, then for all u with $1 - 1/n$ probability,

$$x_u \leq \hat{x}_u \leq x_u + \frac{1}{k} \left\| x - H_k(x) \right\|_1 .$$

Proof. Let $H \subset [n]$ be the set of largest k entries of x. Define $\hat{x}_u^{(r)} = y_{h_r(u)}^{(r)}$. Then,

$$0 \leq \hat{x}_u^{(r)} - x_u = \sum_{\substack{h_r(v)=h_r(u) \\ v \neq u}} x_v = \sum_{\substack{h_r(v)=h_r(u) \\ v \neq u,\ v \in H}} x_v + \sum_{\substack{h_r(v)=h_r(u) \\ v \neq u,\ v \notin H}} x_v = E_H + E_L,$$

where $E_H = \sum_{\substack{h_r(v)\neq h_r(u) \\ v \neq u,\ v \in H}} x_v$ and $E_L = \sum_{\substack{h_r(v)\neq h_r(u) \\ v \neq u,\ v \notin H}} x_v$. For these two parts, we can obtain that

$$\begin{aligned} \Pr[E_H > 0] &\leq \Pr[\exists v \in H \backslash \{u\}.\ h_r(v) = h_r(u) \wedge x_v > 0] \\ &\leq \Pr[\exists v \in H \backslash \{u\}.\ h_r(v) = h_r(u)] \leq \frac{k}{B}. \end{aligned}$$

$$\begin{aligned} \Pr[E_L \geq \frac{1}{k} \left\| x - H_k(x) \right\|_1] &\leq \frac{k \cdot \mathbb{E}[E_L]}{\left\| x - H_k(x) \right\|_1} \\ &= \frac{k}{\left\| x - H_k(x) \right\|_1} \sum_{v \neq u,\ v \notin H} x_v \cdot \Pr[h_r(v) = h_r(u)] \\ &= \frac{k}{B \cdot \left\| x - H_k(x) \right\|_1} \sum_{v \neq u,\ v \notin H} x_v \leq \frac{k}{B}. \end{aligned}$$

Therefore,

$$\begin{aligned} \Pr[\hat{x}_u^{(r)} - x_u \geq \frac{1}{k} \left\| x - H_k(x) \right\|_1] &= \Pr[E_H + E_L \geq \frac{1}{k} \left\| x - H_k(x) \right\|_1] \\ &\leq \Pr[E_H > 0] + \Pr[E_L \geq \left\| x - H_k(x) \right\|_1] \\ &\leq \frac{2k}{B}. \end{aligned}$$

$$\Pr[\hat{x}_u - x_u \geq \frac{1}{k}\|x - H_k(x)\|_1] = \Pr[\min_r\{\hat{x}_u^{(r)}\} - x_u \geq \frac{1}{k}\|x - H_k(x)\|_1]$$
$$= \Pr[\hat{x}_u^{(r)} - x_u \geq \frac{1}{k}\|x - H_k(x)\|_1]^R$$
$$\leq (\frac{2k}{B})^R \leq \frac{1}{n^2}.$$

□

It's easy to finish the proof of performance bound of the cover-min sketch from the count-min sketch. Then, if the input data is from the real world and sparse enough, we yield that $\|x - H_k(x)\|_1 \approx 0$, which ensures our performance theoretically. (Since this is a brief report, we'll not discuss it in much detail. In fact, the entire proof requires 2D-sparse input. However, it's widely believed that data in the real world is in a way sparse with high probability.)

References

[1] Zhuochen Fan, Jiarui Guo, Xiaodong Li, Tong Yang, Yikai Zhao, Yuhan Wu, Bin Cui, Yanwei Xu, Steve Uhlig, and Gong Zhang. Finding simplex items in data streams. In 2023 IEEE 39th International Conference on Data Engineering (ICDE), pages 1953–1966, 2023. doi:10.1109/ICDE55515.2023.00152. 1

[2] Zhuochen Fan, Yinda Zhang, Tong Yang, Mingyi Yan, Gang Wen, Yuhan Wu, Hongze Li, and Bin Cui. Periodicsketch: Finding periodic items in data streams. In 2022 IEEE 38th International Conference on Data Engineering (ICDE), pages 96–109. IEEE, 2022. 2, 3

[3] Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. Journal of Algorithms, 55(1):58–75, 2005. 2

[4] "Capture files from Mid-Atlantic CCDC. URL https://www.netresec.com/?page=MACCDC. 3