ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9413853

COUNT SKETCH WITH ZERO CHECKING: EFFICIENT RECOVERY OF HEAVY COMPONENTS

Guanqiang Zhou *Zhi Tian*

Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA 22030, USA

ABSTRACT

The problem of recovering heavy components of a high-dimensional vector from compressed data is of great interest in broad applications, such as feature extraction under scarce computing memory and distributed learning under limited bandwidth. Recently, a compression algorithm called count sketch has gained wide popularity to recover heavy components in various fields. In this paper, we carefully analyze count sketch and illustrate that its default recovery method, namely median filtering, has a distinct error pattern of reporting false positives. To counteract this error pattern, we propose a new scheme called zero checking which adopts a two-step recovery approach to improve the probability of detecting false positives. Our proposed technique builds on rigorous error analysis, which enables us to optimize the selection of a key design parameter for maximum performance gain. The empirical results show that our scheme achieves better recovery accuracy than median filtering and requires less samples to accurately recover heavy components.

Index Terms— Count sketch, heavy components recovery, median filtering, zero checking, false positives

1. INTRODUCTION

In the era of big data, the dimensions of both learning models and training data are growing at a staggering pace. Given limited resources, direct processing of these high-dimensional signals is costly and even impossible. Usually, these signals have to be compressed into a low-dimensional form that makes the relevant tasks manageable. The tenet is that, the most important information of a high-dimensional vector is typically encrypted in a few heavy components that stand out in magnitude, while the remaining components carry very little information. There is great enthusiasm from various fields in studying the heavy components recovery problem, which is stated as follows.

Assuming vector $\mathbf{x} \in \mathbb{R}^d$ is composed of $k \ll d$ heavy components with prominent magnitudes and $d-k$ non-heavy components that are close to zero, we want to find a nonadaptive compressing matrix $\mathbf{A} \in \mathbb{R}^{m \times d}$ such that the heavy components of $\mathbf{x}$ can be recovered from $\mathbf{Ax}$.

Linear compression has been extensively explored in several research areas, including compressive sensing, data stream computing, and combinatorial group testing [2]. In theory, all algorithms designed to recover signals from linear samples could be applied to recover heavy components. However, we impose two extra constraints for practical concerns. First, the measurement matrix $\mathbf{A}$ should be sparse since the time needed to compute $\mathbf{Ax}$ is proportional to the number of nonzeros in $\mathbf{A}$. This excludes some popular dense matrices such as Gaussian matrices. Second, considering that the vector's dimension could be in the order of millions, we enforce linear recovery to alleviate computational burden, thus excluding nonlinear algorithms such as l_1 minimization.

Count sketch [1], which satisfies both constraints, has been widely applied for heavy components recovery in a variety of applications such as distributed learning [3] and feature selection [4], among others [5-7]. Despite extensive implementations in recent years, the count sketch algorithm has rarely been examined or questioned. In this work, we carefully analyze count sketch and find that its default recovery scheme, median filtering, has a strong tendency to produce false positives, which refer to the type of recovery errors that misidentify a non-heavy component as heavy. This issue often causes the recovered signal to be useless and hence prompts practitioners to use additional mechanisms to mitigate it. For example, in the application of distributed learning, [3] keeps the top Pk elements of the recovered vector and requests the original values of these elements. As a result, the false positives would not increase the recovery error as long as they are no more than $(P-1)k$. However, without recognizing the specific error pattern of false positives, the method implemented in [3] is more of an empirical amendment than a theoretically-sound solution. Besides, such technique incurs additional costs to the system and may not be applicable to a more general setting where there is no access to the true values of the compressed signal. In this paper, we first demonstrate the issue of false positives specific to median filtering and then introduce zero checking, a new recovery scheme that is designed to effectively mitigate false positives via a two-step approach. We empirically show that zero checking outperforms median filtering in terms of lower overall error rate and lower sample requirement for successful recovery. To the best of our knowledge, this work is the first one to demonstrate the unique error pattern of median filtering and provide comparable scheme to address such weakness.

The rest of the paper is organized as follows. Section 2 presents the prior work of count sketch. Section 3 analyzes count sketch and illustrates its false positive phenomenon. Section 4 proposes the new recovery scheme of zero checking. Section 5 draws the conclusion.

2. COUNT SKETCH

Count sketch is usually depicted using the concepts of hash function and hash table within the computer science community [1]. Alternatively, thanks to its linear nature, count sketch can be explained using linear compression as follows.

2.1. Initialization

Count sketch seeks to collect t parallel observations of the data vector $\mathbf{x} \in \mathbb{R}^d$ to enable recovery, with t being an odd number. Conceptually, this process starts with generating t random sampling matrices $\mathbf{A}_1, \ldots, \mathbf{A}_t$. Each $\mathbf{A}_j \in \mathbb{R}^{b \times d}$ constructs its columns indepen-

This work was partly supported by the US NSF grants #1527396, #1741338 and #1939553.

dently obeying the rules: 1) each column contains only one nonzero; 2) the single nonzero is randomly drawn from {1, -1} and is positioned at each row with equal probability $1/b$. It is assumed that $\mathbf{A}_1, \ldots, \mathbf{A}_t$ are shared between the encoder and the decoder.

2.2. Encoding

Given data vector $\mathbf{x} = [x_1, \ldots, x_d]^{\mathrm{T}}$, obtain $\mathbf{y}_j \in \mathbb{R}^b$ via

$$\mathbf{y}_j = \mathbf{A}_j \mathbf{x} \quad j = 1, \ldots, t. \tag{1}$$

Altogether t sample vectors $\mathbf{y}_1, \ldots, \mathbf{y}_t$ are acquired with a total sample size $m = t \cdot b$.

2.3. Decoding

Given sample vectors $\mathbf{y}_1, \ldots, \mathbf{y}_t$, recover an estimated vector of $\mathbf{x}$ from each $\mathbf{y}_j$ via

$$\mathbf{x}^j = \mathbf{A}_j^{\mathrm{T}} \mathbf{y}_j \quad j = 1, \ldots, t. \tag{2}$$

In total, t independent estimated vectors $\mathbf{x}^1, \ldots, \mathbf{x}^t$ are available. To estimate data component x_i, we extract the corresponding estimate x_i^j from each $\mathbf{x}^j$ and form a set of t estimates $\{x_i^1, \ldots, x_i^t\}$, which is referred to as the returned list for x_i throughout. The final estimate of x_i is given by the median of its returned list, i.e.,

$$\hat{x}_i = \mathbf{Median}\{x_i^1, \ldots, x_i^t\} \quad i = 1, \ldots, d. \tag{3}$$

We refer to (3) as "median filtering" in this paper.

It is worth noting that both the encoding in (1) and decoding in (2) and (3) are conducted by linear operations. Further, (2) and (3) can be implemented by very simple hash functions because of the highly sparse nature of samplers $\mathbf{A}_j$.

3. ANALYSIS ON COUNT SKETCH

3.1. The failure probability of a single estimate

Combining (1) and (2), each $\mathbf{x}^j$ is obtained in effect by passing $\mathbf{x}$ through a filtering matrix $\mathbf{\Phi}_j \in \mathbb{R}^{d \times d}$, i.e.,

$$\mathbf{x}^j = \mathbf{A}_j^{\mathrm{T}} \mathbf{y}_j = \mathbf{A}_j^{\mathrm{T}} \mathbf{A}_j \mathbf{x} = \mathbf{\Phi}_j \mathbf{x} \quad j = 1, \ldots, t. \tag{4}$$

In (4), the estimate of x_i is given by the inner product between $\mathbf{\Phi}_j$'s i-th row vector $(\mathbf{\Phi}_j)_i$ and $\mathbf{x}$. Based on the imposed structure of $\mathbf{A}_j$, several characteristics of $(\mathbf{\Phi}_j)_i$ can be drawn: 1) its i-th element (which corresponds to x_i) is 1; 2) each off-diagonal element is nonzero with probability $1/b$; 3) the nonzero off-diagonal elements are independently drawn from {1, -1} with equal probability.

Although the nonzero off-diagonal elements of $(\mathbf{\Phi}_j)_i$ introduce noise into the estimate x_i^j, such noise is innocuous if all involved data components in the inner product are non-heavy ones. This is because non-heavy components have small magnitudes and tend to cancel each other out due to the random signs of the nonzero off-diagonal elements of $(\mathbf{\Phi}_j)_i$. As a result, x_i^j deviates very little from x_i and we consider it a "good estimate".

On the other hand, if one or more interfering heavy components are included in the inner product, then x_i^j tends to deviate greatly from x_i. In such case, we consider x_i^j a bad estimate or recovery failure/error. Note that a bad estimate is bigger or smaller than the ground truth with equal probabilities. Given that each heavy component corrupts the estimate with probability $1/b$, the failure probability of a single estimate is $p = 1 - (1 - 1/b)^k$ with k denoting the number of heavy components in $\mathbf{x}$.

3.2. The failure probability of median filtering

According to (3), median filtering fails when the median of the returned list is bad, which must satisfy two conditions. First, at least $(t + 1)/2$ estimates in the returned list are bad. Second, the bad estimates are tilted toward one side. For example, if 8 out of 9 estimates are bad but they are distributed symmetrically with respect to the ground truth, then the median would still be a good estimate. Denoting the number of bad estimates as Y and tracing down all failed instances for each $Y = i$, our analysis yields the following expression for the failure probability of median filtering:

$$p_{\text{med}} = \sum_{i=\frac{t+1}{2}}^{t} \frac{\sum_{j \in [0, i-\frac{t+1}{2}] \bigcup [\frac{t+1}{2}, i]} \binom{i}{j}}{2^i} \cdot \mathbf{Pr}\{Y = i\} \tag{5}$$

where $\mathbf{Pr}\{Y = i\} = \binom{t}{i} p^i (1-p)^{t-i}$. Note that (5) is with respect to the recovery of one data component, heavy and non-heavy alike.

3.3. The false positive phenomenon

Since heavy and non-heavy components have the same failure probability (5) under median filtering, the ratio of recovery errors resulted from these two categories would be asymptotically equal to the ratio of their composition in the data vector. Since non-heavy components dominantly outnumber the heavy ones ($d-k$ vs. k), the vast majority of recovery errors would be false positives.

4. COUNT SKETCH WITH ZERO CHECKING

4.1. Intuition and scheme

Knowing that false positives are the major source of recovery errors under median filtering, it is helpful to introduce other mechanisms that can spot false positives. Our idea builds on a key observation that, for most false positives, their returned list tends to contain a few good estimates even though the median is a bad one. We utilize this distinct feature of false positives and propose a new recovery mechanism called zero checking. For simplicity, here we assume all non-heavy components are zeros, but the idea applies to general settings where non-heavy components have negligible magnitudes and hence can be rounded to zero.

The recovery process of zero checking is described as follows. After obtaining the returned list of t estimates as stated in Section 2.3, our scheme implements a two-step approach to determine the best estimate:

Step 1: If the median of the returned list is zero or negligibly small, then return it as the output. Otherwise, go to Step 2.

Step 2: If the returned list contains at least R zeros, then rule this case as a false positive and return zero as the output. Otherwise, return the median.

In the above scheme, Step 1 only declares negatives/non-heavy components, while leaving all positives, true or false, for further screening in Step 2. Using a detection threshold R, Step 2 decides whether components that would have been identified as heavy by median filtering are false positives or not. Here the detection threshold R is a key parameter to ensure the success of zero checking. In contrast to median filtering, which imposes the same error rate for non-heavy and heavy components, zero checking obviously decreases error rate for non-heavy components by excluding a portion of false positives. However, a natural concern is that the thresholding operation in Step 2 might inadvertently misclassify some heavy

components as false positives, thus increasing the overall error rate. In the next section, we will address this concern and show how to select the detection threshold R.

4.2. The guideline on selecting the detection threshold

Defining $p_{\text{NH}}(R)$ and $p_{\text{H}}(R)$ as the failure probabilities of non-heavy and heavy components respectively, the average failure probability under zero checking is given by

$$p_{\text{ZC}}(R) = \frac{d-k}{d} \times p_{\text{NH}}(R) + \frac{k}{d} \times p_{\text{H}}(R). \tag{6}$$

When $R \geq (t+1)/2$, zero checking reduces to median filtering, for the median estimate is always returned in Step 2. As R decreases, $p_{\text{NH}}(R)$ declines but $p_{\text{H}}(R)$ may go up. In order to guarantee that $p_{\text{ZC}}(R)$ is no larger than p_{med} in (5), the detection threshold $R \in \{0, \ldots, (t+1)/2\}$ should make the following inequality hold:

$$(d-k) \times (p_{\text{med}} - p_{\text{NH}}(R)) \geq k \times (p_{\text{H}}(R) - p_{\text{med}}) \tag{7}$$

where the difference between the left side and right side translates to the performance gain achieved by zero checking over the baseline, and the optimal R that maximizes this gap also equivalently leads to the lowest $p_{\text{ZC}}(R)$ in (6).

To determine the best R, it is necessary to derive the closed-form of $p_{\text{NH}}(R)$ and $p_{\text{H}}(R)$. Here $p_{\text{NH}}(R)$ is directly obtained from (5) by keeping only R terms, i.e.,

$$p_{\text{NH}}(R) = \sum_{i=t-R+1}^{t} \frac{\sum_{j \in [0, i-\frac{t+1}{2}] \bigcup [\frac{t+1}{2}, i]} \binom{i}{j}}{2^i} \cdot \mathbf{Pr}\{Y=i\}. \tag{8}$$

The expression of $p_{\text{H}}(R)$ is less obvious because it has two sources: 1) the baseline failure probability p_{med}; 2) the risk of the returned list containing at least R zero estimates, which hinges on p_0, the probability for a single estimate of a heavy component to be neutralized as zero. Since p_0 is strongly dependent upon the magnitude diversity among heavy components, we will first elaborate such dependency, and then give two upper bounds on $p_{\text{H}}(R)$ under two opposite extreme assumptions on the data diversity.

Recall that an estimate is the sum of randomly-signed interfering heavy components and the target heavy component. When the magnitudes of involved components are not identical, it is very unlikely for the estimate to be neutralized. To illustrate this point, we calculate $p(s) = \binom{k}{s}(1/b)^s(1-1/b)^{k-s}$, the probability of an estimate being corrupted by s heavy components. Since the estimate can only be neutralized with one or more interfering heavy components, the inequality holds: $p_0 \leq \sum_{s \geq 1} p(s)$. If there is no heavy component sharing the same magnitude as the target heavy component, then $p_0 \leq \sum_{s \geq 2} p(s)$. Since $p(s)$ is an exponentially decreasing function of s, p_0 is stringently upper-bounded given different magnitudes. Moreover, the expression of $\sum_{s \geq 2} p(s)$ could still be massively overestimated, because the $s+1$ involved magnitudes may not add up to zero regardless of the sign combination, e.g., $\{2, 3, 4\}$. In addition, given a set of favorable magnitudes, the feasible sign combination is usually unique; for example, given set $\{2, 4, 6\}$, if 2 is the target heavy component, then the interfering components have to be 4 and -6 out of the four possible sign combinations, suggesting a tighter bound of $p_0 \leq \sum_{s \geq 2} p(s)/4$.

Considering the above reasoning, it is clear that p_0 reaches maximum when the magnitudes of involved components are identical. Building on this observation, we can further conclude that $p_{\text{H}}(R)$ achieves its universal upper bound only when all heavy components share the same magnitude, which corresponds to the first extreme assumption of no data diversity. We call this scenario the worst case and our analysis yields the following closed-form of $p_{\text{H}}(R)$ in the worst case:

$$\begin{aligned} p_{\text{H},1}(R) = &\sum_{i=R}^{\frac{t-1}{2}} \frac{\sum_{j=R}^{i} \binom{i}{j}}{2^i} \cdot \mathbf{Pr}\{Y=i\} + \\ &\sum_{i=\frac{t+1}{2}}^{t} \frac{\sum_{j \in [0, i-\frac{t+1}{2}] \bigcup [R, i]} \binom{i}{j}}{2^i} \cdot \mathbf{Pr}\{Y=i\}. \end{aligned} \tag{9}$$

Compared to the worst case with no data diversity, an ideal case is considered where all heavy components have different magnitudes. In the ideal case, p_0 is upper-bounded by $\big(1-(1-1/b)^k - k \cdot (1/b) \cdot (1-1/b)^{k-1}\big)/4$ as discussed. Adding two aforementioned failure sources together, we can upper-bound $p_{\text{H}}(R)$ in the ideal case as

$$p_{\text{H},2}(R) \leq p_{\text{med}} + \binom{t}{R} \times p_0^R. \tag{10}$$

Depending on which assumption is used, we can plug either (9) or (10), together with (8), into (7) to obtain the effective range for R. Moreover, the optimal R can be selected by minimizing $p_{\text{ZC}}(R)$ in (6). Note that we do not claim that either the worst case or the ideal case is a practical scenario. Our intention of hypothesizing these two cases is to assess how much zero checking can outperform median filtering in both the least favorable and the most favorable settings.

4.3. Experimental validation

This section investigates the empirical performance of zero checking compared to the median filtering baseline. The task is to recover all k heavy components of a data vector $\mathbf{x} \in \mathbb{R}^d$ under the count sketch regime with $m = t \cdot b$ samples. Throughout, the setting parameters are fixed to be $d = 10000, k = 50, b = 250$. We first compute the optimal detection threshold R in (6) and the corresponding failure probability $p_{\text{ZC}}(R)$ under both scenarios, shown in Figure 1 and Figure 2 respectively. As baseline, the threshold for median filtering is $(t+1)/2$ as discussed and its failure probability is p_{med} in (5).

Figure 1 shows that the optimal threshold of zero checking is consistently lower by a small margin than the baseline in the worst case, whereas a much lower threshold, approximately half of the baseline's threshold, is allowed in the ideal case. Accordingly, Figure 2 shows that, in theory, zero checking can always outperform median filtering in terms of error rate, with marginal performance gain in the worst case and significant advantage in the ideal case.

Next we apply the obtained thresholds in Figure 1 to test the actual performance of zero checking under both scenarios. For the worst case, the data vector is generated by inserting $k = 50$ nonzeros into a 10000-dimensional all-zero vector, each nonzero having a random sign and a fixed magnitude. The only difference for the ideal case is that the magnitudes of nonzero elements are drawn from $\{1, \ldots, 50\}$ without replacement. We run 1000 Monte Carlo experiments under both scenarios. The average recovery errors for zero checking and median filtering match their theoretical failure probabilities in Figure 2 very well, and we omit these curves for lack of space. Another relevant metric is the likelihood of successful recovery on the whole data vector. The successful recovery rates of zero checking under the worst case and the ideal case are plotted in Figure 3 and Figure 4 respectively.

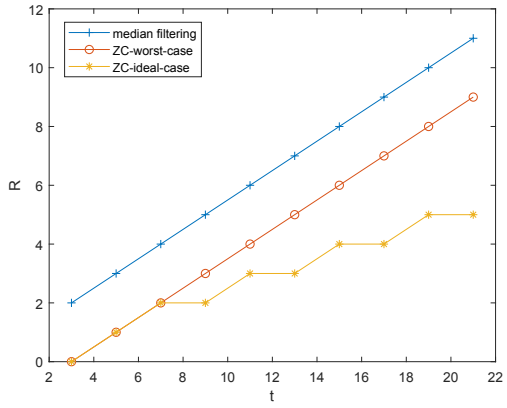

Fig. 1. The optimal threshold R of zero checking.

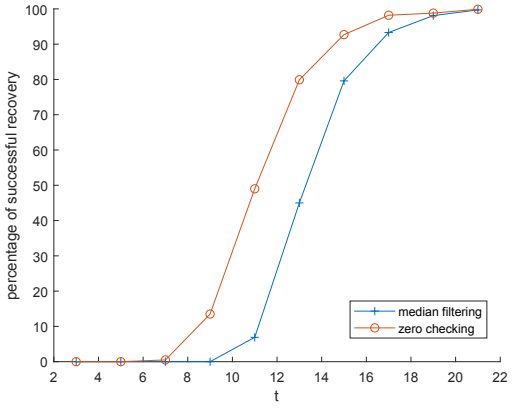

Fig. 3. Successful recovery rate in the worst case.

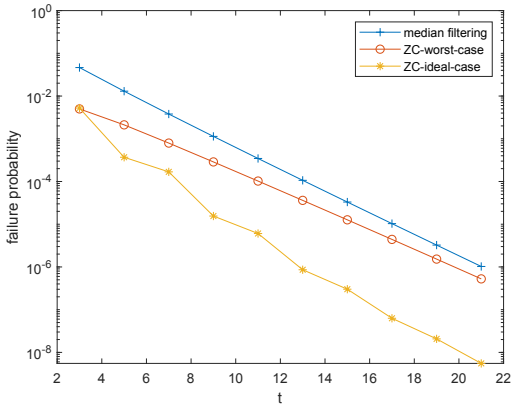

Fig. 2. The average failure probability.

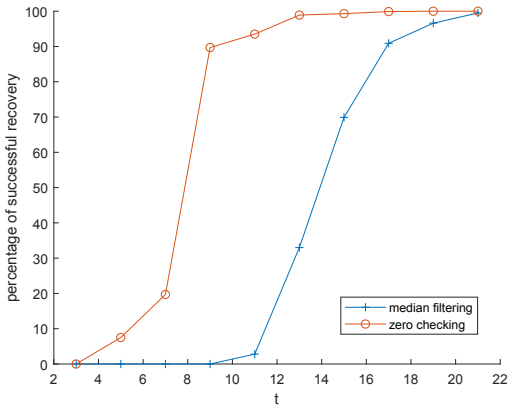

Fig. 4. Successful recovery rate in the ideal case.

In the worst case, although the performance gain of zero checking is not as impressive, it is still beneficial. For example, when $t = 11$, zero checking has a recovery rate improvement of over 40%. In the ideal case, the advantage of zero checking over median filtering is significant. To reach 90% successful recovery rate, zero checking saves almost half of the samples required by the baseline (9 vs. 17).

In practice, the level of diversity among heavy components generally lies between the two discussed cases, suggesting that the performance gain should also be between the two. If there is no prior knowledge about the data vector, we can always adopt the surefire approach of using the conservative detection threshold that matches the worst case. However, if we want to achieve more aggressive improvement beyond the bare minimum, it is necessary to know the distribution of the data vector and to exploit the diversity of the heavy components.

5. CONCLUSION

In this paper we have carefully examined the count sketch algorithm when applied to recover heavy components. We found that the default recovery method, median filtering, has a distinct feature of producing false positives. To overcome this error pattern, we propose a new recovery mechanism, zero checking, that effectively suppresses false positives by applying an optimal detection threshold to maximize overall recovery accuracy. The empirical results show that zero checking outperforms the baseline even in the worst-case scenario. The proposed scheme holds great promise for applications with prior information on the distribution of data vector and a high level of data diversity.

6. REFERENCES

[1] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams." *Proc. of the 29th ICALP*, pp. 693–703, 2002.

[2] A. Gilbert, et al, "Sparse recovery using sparse matrices." *Proceedings of the IEEE*, 98(6): 937-947, 2010.

[3] N. Ivkin, et al, "Communication-efficient distributed SGD with sketching." *Advances in Neural Information Processing Systems*, 2019, pp. 13144–13154.

[4] A. Aghazadeh, R. Spring, et al, "MISSION: Ultra large-scale feature selection using count-sketches." *International Conference on Machine Learning*, 2018, pp. 80–88.

[5] K. Tai, et al, "Sketching linear classifiers over data streams." *Proceedings of the 2018 International Conference on Management of Data*, pp. 757–772. ACM, 2018.

[6] J. Jiang, et al, "SketchML: Accelerating distributed machine learning with data sketches." *Proceedings of the 2018 International Conference on Management of Data*, pp. 1269–1284. ACM, 2018.

[7] R. Spring, A. Kyrillidis, et al, "Compressing gradient optimizers via count-sketches." *International Conference on Machine Learning*, 2019.