CSCE 689: Special Topics in Modern Algorithms for Data Science Fall 2023

Lecture 15 — October 2, 2023

Prof. Samson Zhou *Scribe: Jung Hoon Seo*

In the previous lectures, we delved into various streaming algorithms, conducting a comparison between Misra-Gries and Count-Min. Our analysis revealed that Count-Min offers greater versatility since the Misra-Gries algorithm is confined to handling insertions alone. Following this, we delved into Count-Sketch. While Misra-Gries and Count-Min utilize the L_1 norm, which involves examining the entire data stream to identify coordinates with the least percentage, Count-Sketch employs the L2 norm and a geometric interpretation. This approach provides better approximations for each item in the dataset due to the smaller errors associated with the L_2 norm. In this lecture, we will explore another streaming model called Sparse Recovery.

1 Sparse Recovery

1.1 Applications

- **Anomaly Detection** - Imagine a scenario where there is not much activity on the network. However, as time progresses, the network may experience an influx of users. This presents a vulnerability as adversaries attempt to overwhelm the network by flooding it with messages, often resorting to paying spam boxes to do so. In such situations, sparse recovery comes into play, as it involves modeling typical behavior as a sparse signal. This allows for the real-time detection of deviations from this model, helping to safeguard the network.
- **Real-time Compressive Imaging** - Images can undergo sparsification through a transformation known as the Haar wavelet. Following the application of the wavelet, the majority of pixels tend to become dark, leaving only a sparse number of them appearing white. This technique plays a role in facilitating the real-time reconstruction of high-resolution images and videos.
- **Online Natural Language Processing** - Noiseless sparse recovery aids in the extraction of relevant features or patterns from streaming text data by selecting K-words that comprehensively represent the topics at hand. By employing this technique, we can achieve tasks such as text summarization and topic modeling.

1.2 Algorithms

The goal of Sparse Recovery is to retrieve the k non-zero coordinates along with their respective frequencies from an insertion-deletion stream of a certain length, where there are no more than k non-zero coordinates at the end. Suppose $k = 1$, and we are promised the coordinate has a frequency of 1. If this is the case, the Sparse Recovery algorithm works by keeping s running sum of all the coordinates

$$\sum_{i\in[m]} s_i c_i = j$$

- Each insertion to coordinate $c_i \in [n]$ as $u_i \leftarrow (s_i = 1, c_i)$
- Each deletion to coordinate $c_i \in [n]$ as $u_i \leftarrow (s_i = -1, c_i)$

For example, if we have the following:

$$u_1 : \text{Increase } f_6 \qquad u_2 : \text{Increase } f_3$$
$$u_3 : \text{Decrease } f_3 \qquad u_4 : \text{Increase } f_2$$
$$u_5 : \text{Decrease } f_6$$

we can apply the running sum of coordinates, then

$$\sum_{i \in [m]} s_i c_i = j = (1)(6) + (1)(3) + (-1)(3) + (1)(2) + (-1)(6) = 2$$

Nevertheless, this algorithm encounters failure when the frequency is not equal to 1 or when $k \neq 1$. To overcome this limitation, we can enhance the algorithm by incorporating an additional premise: maintaining a running sum of all coordinates and introducing a distinct linear combination of these coordinates. Then, we have

$$\sum_{i \in [m]} s_i c_i = j \cdot f_j \;\; and \sum_{i \in [m]} s_i c_i^2 = j^2 \cdot f_j$$

If we consider this new example, where the frequency value for the last coordinate is now 2,

$$u_1 : \text{Increase } f_6 \qquad u_2 : \text{Increase } f_3$$
$$u_3 : \text{Decrease } f_3 \qquad u_4 : \text{Increase } f_2$$
$$u_5 : \text{Decrease } f_6 \qquad u_6 : \text{Increase } f_2$$

then, we have

$$\sum_{i \in [m]} s_i c_i = j \cdot f_j = (1)(6) + (1)(3) + (-1)(3) + (1)(2) + (-1)(6) + (1)(2) = 4$$

$$\sum_{i \in [m]} s_i c_i^2 = j^2 \cdot f_j = (1)(36) + (1)(9) + (-1)(9) + (1)(4) + (-1)(36) + (1)(4) = 8$$

If we divide $\sum_{i \in [m]} s_i c_i^2$ by $\sum_{i \in [m]} s_i c_i$, then we have $f_j = 2$ and $j = 2$.

$$\frac{\sum_{i \in [m]} s_i c_i^2}{\sum_{i \in [m]} s_i c_i} = \frac{j^2 \cdot f_j}{j \cdot f_j} = j = \frac{8}{4} = 2$$

In the general case, when there are k non-zero coordinates, we compute $2k$ running sums of different linear combinations of all the coordinates. At the end of this process, we have $2k$ equations with $2k$ unknown variables. The space complexity for this case is $O(k)$ words of space.

However, in real-life scenarios, even when there are k items that predominantly contribute to the noise, you may still encounter unwanted interference in other coordinates. For instance, consider a situation where you have K coordinates with significantly high values. Suppose we have a vector

$$100 \quad 0.01 \quad 100 \quad -0.01 \quad 0.1 \quad 1.1 \quad -0.1 \quad 0.1 \quad -0.1$$

then the best k-sparse vector that is close to the vector above is going to be

$$100 \quad 0 \quad 100 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0$$