CMPUT 675: Algorithms for Streaming and Big Data **Fall 2019**

Lecture 9 (Oct 2, 2019): Sparse Recovery

Lecturer: Mohammad R. Salavatipour *Scribe: Candelario Gutierrez*

9.1 Sparse Recovery

Given a stream σ it defines a frequency vector f where f_i (for each $i \in [n]$) is the frequency of item i. In the past lecture we saw the use of count sketch algorithm applied for sparse recovery. As a recap, we called a vector s-sparse if there are at most s non-zero entries in it. Here our goal is to design an algorithm that can detect if a vector is 1-sparse (or s-sparse in general) and if so find the corresponding indices. We start with 1-sparse detection and recovery and show how we can use it to design an s-sparse and recovery algorithm.

9.1.1 1-Sparse Recovery

Given a vector $a \epsilon \mathbb{R}^n$, we want to detect if there is a single non-zero a_i (and if so, find it), or detect that such index doesn't exist. Consider the streaming model and suppose we are interested in the frequency vector f:

$\ell \leftarrow 0$
$s \leftarrow 0$
While there is token (j, c) do
  $\ell \leftarrow \ell + c$
  $s \leftarrow s + cj$
return $\frac{s}{\ell}$ and $f_{\frac{s}{\ell}} = \ell$

Note that after the algorithm finishes we have:

$$\ell = \sum_{i: f_i \neq 0} f_i \qquad s = \sum_{i \epsilon [n]} i f_i$$

So, if there is a single non-zero f_j then $\ell = f_j$ and $s = j f_j$, and we have $j = \frac{s}{\ell}$. But this algorithm cannot detec if there is a single j.

9.1.2 1-Sparse Detect and Recovery

Let q be a prime $n^2 \le q \le 2n^2$

$\ell \leftarrow 0$
$s \leftarrow 0$
$p \leftarrow 0$

Let r be random from $\{1...q-1\}$

While there is a token (j, c) do
  $\ell \leftarrow \ell + c$
  $s \leftarrow s + cj$
  $p \leftarrow p + cr^j$
if $\frac{s}{\ell} \notin \mathbb{Z}$ then say fail
if $p \neq \ell r^{\frac{s}{\ell}}$ then say fail
else return $\frac{s}{\ell}$ and $f_{\frac{s}{\ell}} = \ell$

Let R be the random value for r:

$$\ell = \sum_{j \epsilon [n]} f_j = \sum_{j : f_j \neq 0} f_j$$

$$s = \sum_{j \epsilon [n]} j f_j = \sum_{j : f_j \neq 0} j f_j$$

$$p = \sum_{j \epsilon [n]} R^j f_j = \sum_{j : f_j \neq 0} R^j f_j$$

If there is a single index i such that $f_i \neq 0$ then $\ell = f_i$, $s = i f_i$ and $p = R^{\frac{s}{\ell}} f_i$, and we find the correct answer. Now, let's suppose that is not 1-sparse and $\frac{S}{\ell} \in \mathbb{Z}^+$:

$$P(x) = \Big(\sum_{j : f_j \neq 0} f_j x^j\Big) - \ell x^{\frac{s}{\ell}}$$

So, $P(x)$ is a degree $\le n$ polynomial and the number of roots of $P(x)$ is $\le n$. We have a false positive if $P(R) = 0$

$$\Pr[\text{false positive}] = \Pr[P(R) = 0] \le \frac{n}{q} \le \frac{1}{n}$$

Total space of: $O(\log n + \log M)$ for ℓ, s and p.

9.1.3 S-Sparse Recovery

We use 1-sparse detection and recovery as a blackbox to build s-sparse recovery.

- Let $D[1..t, 1...2s]$ maintain $2ts$ independent 1-sparse recoveries.
- Let $h_1...h_t[n] \to [2s]$ be independent 2-universal hash functions.
- For each token (j, c): For $1 \leq i \leq t$ we update 1-sparse recovery for $D[i, h_i(j)]$.
- Agregate non-zero coordinates and return them all.

Suppose f is s-sparse, let $S = \{j | f_j \neq 0\}$ for any index $j \in S$. The probability that j lands in a bucket (among $1...2s$) by itself is $\geq \frac{1}{2}$:

$$Pr[\text{row 1 fails to recover } i \in S] \leq \sum_{\substack{j: f_j \neq 0 \\ j \neq i}} Pr[h(i) = h(j)] \leq \sum \frac{1}{2s} \leq \frac{s-1}{2s} \leq \frac{1}{2}$$

Therefore:

$$Pr[\text{all rows } 1...t \text{ fail to recover } i] \leq \frac{1}{2^t} \leq \frac{\delta}{s}$$

So that:

$$Pr[\text{some } i \in S \text{ is not recovered}] \leq \delta$$

9.2 Sampling with a Reservoir

Suppose we want to have a uniform sample of size k from a stream. Based on the algorithm proposed by Pavlos S. Efraimidis and Paul G. Spirakis [ES06] from 2006.

- Given a set of size N, pick a small size k sample.
- Stream model.

```
Easy case: k − 1

s ← ∅
i ← 0
While there are more elements do
    i ← i + 1, say x_i is the current element
   s ← x_i with probability 1/i
return s
```

It is an easy exercise to verify that at any time, s is a sample of the stream seen so far. For $k > 1$ with replacement, we can run k parallel copies of sampler for $k = 1$.

More cases and applications will be presented in the next lecture.

References

CCFC04 M. Charikar, K.C. Chen, and M. Farach-Colton, Finding frequent items in data streams. *Theoretical Computer Science*, 312:03–15, 2004.

ES06 Pavlos S. Efraimidis, Paul G. Spirakis, Weighted random sampling with a reservoir. *Journal Information Processing Letters*, 97(5):181-185, 2006.