

Algorithms for Big Data:

Sublinear and Streaming Algorithms

Project Group CS 05

Student 01 : Vivian Guo

Student 02 : Rachel Ma

Under the Mentorship of Vihan Shah

WiM Directed Reading Program

University of Waterloo

Introduction

What characterizes a good time complexity?

Introduction

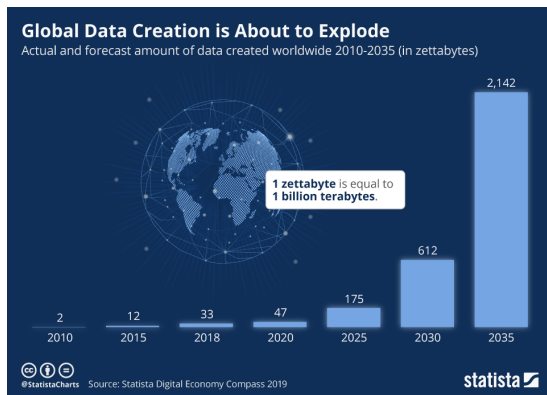


Figure: In an age with so much data, even linear runtime may not suffice

Example Applications

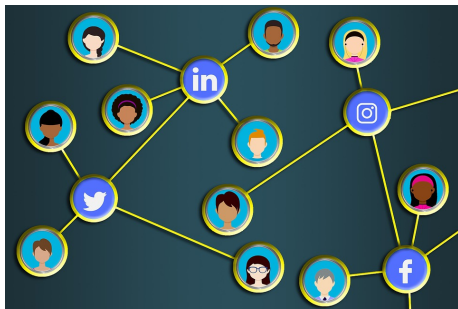


Figure: Computing the average number of friends users have is synonymous to computing the average degree in a graph

Example Applications



Figure: Determining the number of daily unique users to Amazon can be modelled as determining number of distinct elements in a stream

Sublinear Algorithms

What are sublinear algorithms?

Algorithms whose resource requirements (e.g. time or space) are substantially smaller than the size of the input (n) that they operate on.

Sublinear Algorithms

What are sublinear algorithms?

Algorithms whose resource requirements (e.g. time or space) are substantially smaller than the size of the input (n) that they operate on.

Today we will look at the **sparse recovery** problem where we will minimize the number of queries/measurements to recover a vector.

Sparse Recovery

Consider the following problem in \mathbb{F}_2 ...

$$\begin{bmatrix} ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Sparse Recovery


Consider the following problem in \mathbb{F}_2 ...

$$\begin{array}{|c|} \hline ? \\ \hline \end{array} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

- We are allowed queries in the form of inner products

Sparse Recovery

Consider the following problem in \mathbb{F}_2 ...


$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

- We are allowed queries in the form of inner products
- How would you solve this problem?

Sparse Recovery

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

A straightforward solution: let $A = I_n$ (the identity matrix) to get $A \cdot x = I_n x = x$ which obviously allows us to recover any x .

Sparse Recovery

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

A straightforward solution: let $A = I_n$ (the identity matrix) to get $A \cdot x = I_n x = x$ which obviously allows us to recover any x .

Can we do better than this solution?

Sparse Recovery

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

A straightforward solution: let $A = I_n$ (the identity matrix) to get $A \cdot x = I_n x = x$ which obviously allows us to recover any x .

Can we do better than this solution? NO!

1-Sparse Recovery

A 1-sparse vector has just 1 non-zero entry. What measurements can we perform to recover the index of that non-zero bit?

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

1-Sparse Recovery

With structured data such as a 1-sparse vector, we have more tools at our disposal ... such as simulating binary search.

see here!

1-Sparse Recovery: Adaptive Solution

Adaptive Solution:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = 1 \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = 0 \rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0$$

An output of 1 at each stage tells us to focus in on the bottom half of the vector, while 0 tells us to focus on the top half.

1-Sparse Recovery: Non-Adaptive Solution

- what if we do not get to look at the output at each stage? What queries do we use then?

1-Sparse Recovery: Non-Adaptive Solution

- what if we do not get to look at the output at each stage? What queries do we use then?

$$x = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, a_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, a_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}, a_3 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

- In this case, $n = 8$, $\log(8) = 3$; we can recover the index of 1 in x with 3 vectors.

1-Sparse Recovery: Non-Adaptive Solution

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = 1, \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = 0, \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = 0$$

$$(100)_2 = 4$$

2-Sparse Recovery

- What if the vector has 2 non-zero indices?
- Coming up with a non-adaptive solution is already very tricky.
- What if there are k non-zero indices?
- We want to extend the binary search approach for $k = 1$ case to the general k case, which seems quite challenging.

General k-Sparse

What is k-sparse?

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow k = 1, \quad \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \Rightarrow k = 3$$

General k-Sparse

What is k-sparse?

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow k = 1, \quad \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \Rightarrow k = 3$$

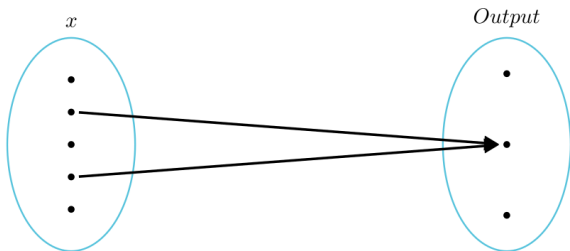
- We will use another approach based on probabilistic arguments to show that we can pick random $A \in \mathbb{F}_2^{m \times n}$ and with high probability we can recover x from $A \cdot x$ for all $x \in \mathbb{F}_2^n$

General k-Sparse

Uniqueness of Measurement Output

We should make sure that for any two different $x \neq y$, $A \cdot x \neq A \cdot y$ as otherwise we cannot distinguish x and y from the resulting vector.

We do NOT want:



Probabilistic Analysis

Before we describe the algorithm completely, we look at the following important lemma:

Lemma:

For all $x \neq y \in \mathbb{F}_2^n$, and $r \in_R \mathbb{F}_2^n$, $\Pr_r(\langle r, x \rangle = \langle r, y \rangle) = \frac{1}{2}$

Proof: Let $i \in [n]$ be any index where $x_i \neq y_i$. Consider picking r by first pick $r_{-i} \in \mathbb{F}_2^{n-1}$ where r_{-i} is the vector of r excluding the index i , and then pick $r_i \in \mathbb{F}_2$. We have two cases:

See here!

Probabilistic Analysis

$$x = [0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1]$$

$$y = [0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1]$$

$$r = [1 \quad 1 \quad ? \quad 0 \quad 0 \quad 1 \quad 1 \quad 0]$$

Case 1: $\langle r_{-i}, x_{-i} \rangle = \langle r_{-i}, y_{-i} \rangle$

We need $r_i = 1$ to determine $x \neq y$. The probability of having $r_i = 1$ is $\frac{1}{2}$, so we have

$$\Pr_r(\langle r, x \rangle = \langle r, y \rangle) = \frac{1}{2}$$

Probabilistic Analysis

$$\begin{aligned}x &= [0 & 1 & 0 & 1 & 0 & 1 & 0 & 1] \\y &= [0 & 1 & 1 & 1 & 0 & 1 & 0 & 1] \\r &= [1 & 1 & ? & 0 & 0 & 1 & 1 & 0]\end{aligned}$$

Case 1: $\langle r_{-i}, x_{-i} \rangle = \langle r_{-i}, y_{-i} \rangle$

We need $r_i = 1$ to determine $x \neq y$. The probability of having $r_i = 1$ is $\frac{1}{2}$, so we have

$$\Pr_r(\langle r, x \rangle = \langle r, y \rangle) = \frac{1}{2}$$

Case 2: $\langle r_{-i}, x_{-i} \rangle \neq \langle r_{-i}, y_{-i} \rangle$

In this case, we need $r_i = 0$ to determine $x \neq y$, as $\langle r_i, x_i \rangle = \langle r_i, y_i \rangle$ promises $\langle r_{-i}, x_{-i} \rangle + \langle r_i, x_i \rangle \neq \langle r_{-i}, y_{-i} \rangle + \langle r_i, y_i \rangle$. The probability of having $r_i = 0$ is $\frac{1}{2}$, so we have

$$\Pr_r(\langle r, x \rangle = \langle r, y \rangle) = \frac{1}{2}$$

General k-Sparse

- Using the above lemma, we can conclude that with a random matrix, we can distinguish x and y with “high enough” probability:

$$\Pr_A(A \cdot x = A \cdot y) = \frac{1}{2^m}$$

where m is the number of rows of A .

- If m is large enough we can argue that with high probability, A can distinguish between all pair of k -sparse vectors $x, y \in \mathbb{F}_2^n$
- $m = O(k \log(n))$

Conclusion

- 1 Probabilistic analysis provides us with tools to solve problems in sublinear time with “high enough” accuracy.
- 2 Sublinear algorithms are designed to operate with resource requirements significantly lower than the size of their input. To achieve this, they rely on techniques of approximation and randomization, providing valuable information about the problem with a fraction of the resources typically needed.

Conclusion

- 1 Probabilistic analysis provides us with tools to solve problems in sublinear time with “high enough” accuracy.
- 2 Sublinear algorithms are designed to operate with resource requirements significantly lower than the size of their input. To achieve this, they rely on techniques of approximation and randomization, providing valuable information about the problem with a fraction of the resources typically needed.

Thank You!