Adaptive Matrix Vector Product

Santosh Vempala
Georgia Tech

David P. Woodruff
IBM Research - Almaden

Problem

- Given m x n matrix A, want to preprocess it so that,

- Given the coordinates of an n-dimensional vector x: $x_1, \dots, x_n$ in order, output the coordinates of Ax: $(Ax)_1, \dots, (Ax)_m$ in order

- A and x have entries from a field F

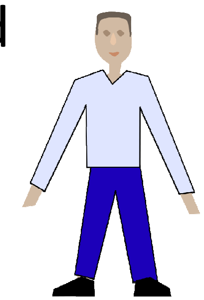
- Goals
  - k passes over the coordinates $x_1, \dots, x_n$
  - Use as little working memory as possible
  - Don't count the output tape containing $(Ax)_1, \dots, (Ax)_m$ towards memory

Applications

- (Special purpose hardware) A is hardwired and special hardware built for efficient products with arbitrary vectors

- (Video streaming) Packet filtering or packet processing applied to video streaming. Input and output should be in correct order

- (Human computation) Humans have a password *schema* represented by a matrix A, and given a *challenge* x (e.g., a website name), must output Ax in order as their password [Blocki], [Blum, Vempala]
    - Humans have small working memory (2-3 characters at a time)
    - Humans can memorize a procedure (how to process x given A)



Examples

- Store $\langle A_{i1}, x \rangle, \ldots, \langle A_{ir}, x \rangle$ where $A_{i1}, A_{i2}, \ldots, A_{ir}$ are rows of a basis for the row span of A
  - O(rank(A)) words of memory, and 1-pass

Can we do better?

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

- Identity matrix A = $I_{n \times n}$, so rank(A) = n

- Output $(Ax)_1, \ldots, (Ax)_n$ in order while reading $x_1, \ldots, x_n$ in order using O(1) words of memory, and 1-pass!

More Examples

- Anti-diagonal matrix A: $(Ax)_1, \dots, (Ax)_n = x_n, x_{n-1}, x_{n-2}, \dots, x_1$

- Have to wait until you see x_n before you can start outputting

- But you need to remember $x_1, x_2, x_3, \dots, x_{n-1}$ so $\Omega(n)$ words of memory for 1-pass algorithms

- Both identity and anti-diagonal matrix have rank n

Is there a parameter better than rank capturing the memory required?

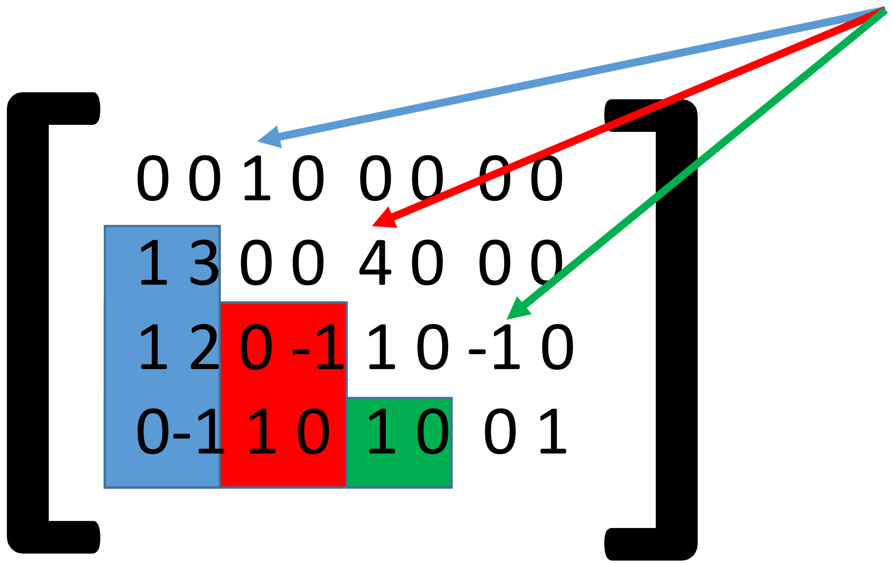
Talk Outline

1. Streaming Rank
    1. Streaming Rank 1-Pass Upper Bound
    2. Streaming Rank 1-Pass Lower Bound

2. k-Pass Streaming Rank
    1. Streaming Rank k-Pass Upper Bound
    2. Streaming Rank k-Pass Lower Bound

3. Applications

Streaming Rank



Boundary point b(i) is the rightmost non-zero entry of row i

B_i is submatrix to the left and underneath b(i)

- Streaming rank $r = \max_i \text{rank}(B_i)$

- Theorem: the space complexity of computing $A \cdot x$ is $\Theta(r)$ words

Streaming Rank Upper Bound for 1-Pass

- Start with i = 0 and an empty basis B
- Initialize b(0) = 1
- Repeat:
  - If b(i+1) $\leq \mathrm{b(i)}$,
    - B stays the same
    - Use inner products of B with x to output $(\mathrm{Ax})_{\mathrm{i+1}}$
  - Otherwise $\mathrm{b(i+1)} > \mathrm{b(i)}$,
    - Extend B to a basis B' of $\mathrm{B_{i+1}}$ by extending r rows of $\mathrm{B_i}$ to rows of $\mathrm{B_{i+1}}$
    - Use B to compute inner products of x with first b(i)-1 coordinates of each row of B', and compute remaining part of inner product of x with rows of B' by advancing along coordinates of x until b(i+1)
  - Output $(\mathrm{Ax})_{\mathrm{i+1}}$ using a linear combination of the inner products
  - i $\leftarrow \mathrm{i} + 1$

Streaming Rank Upper Bound for 1-Pass

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 4 & 0 & 0 & 0 \\ 1 & 2 & 0 & -1 & 1 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Streaming rank r = 2

When reading x_1 and x_2,

- store inner products: <x, (1, 3)> and <x, (1, 2)>
- look at x_3 and output $(Ax)_1 = x_3$

When reading x_3 and x_4,

- use <x, (1, 3)> and <x, (1, 2)> to get <x, (0, -1)>
- extend <x, (1,2)> to <x, (1, 2, 0, -1)>
- extend <x, (0,-1)> to <x, (0,-1, 1, 0)>
- look at x_5 and output $(Ax)_2 =$ <x, (1, 3)> $+4x_5$

When reading x_5 and x_6,

- extend <x, (0, -1, 1, 0)> to <x, (0, -1, 1, 0, 1, 0)>
- look at x_7
- output $(Ax)_3 = < x, (1, 2, 0, -1) > +x_5 - x_7$

Streaming Rank Lower Bound for 1-Pass

- Never store more than r inner products, so O(r) word upper bound

  Is this optimal?

- By Yao's minimax principle suffices to give a distribution on x such that any deterministic algorithm succeeds with probability < 1/3 if using less than r words of memory

- Let streaming rank r = rank(B_i)

- If $x \in GF(q)^n$, let the first b(i) coordinates of x be uniform on $GF(q)^n$ and remaining n-b(i) coordinates equal 0
- For infinite fields, reduce problem to computing $A \cdot x$ over $GF(poly(mn))^n$

- Intuition: Upon reading $x_{b(i)}$, Alg must remember r inner products to output $(Ax)_j$ for j > i
- Formalize with information theory and Fano's Inequality

Talk Outline

1. Streaming Rank
    1. Streaming Rank 1-Pass Upper Bound
    2. Streaming Rank 1-Pass Lower Bound

2. k-Pass Streaming Rank
    1. Streaming Rank k-Pass Upper Bound
    2. Streaming Rank k-Pass Lower Bound

3. Applications

k-pass streaming rank

- Upper triangular all 1’s matrix
- 1-pass streaming rank is $\Omega(n)$

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

- But 2-pass streaming rank is O(1):
  - In first pass, sum all entries of x
  - In second pass, subtract next digit from running sum and output it

- How to characterize k-pass streaming rank?

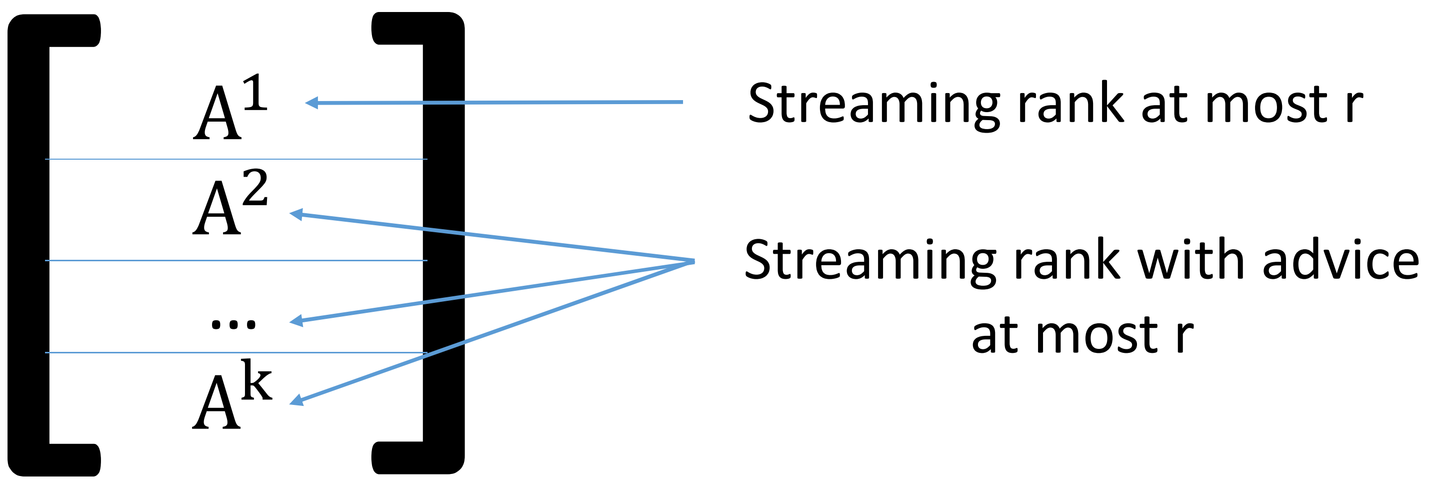
k-Pass Streaming Rank

- Streaming rank with advice
  - given r arbitrary words of advice that may depend on x, output $A \cdot x$ using the advice and r words of working memory
  - with no advice, this coincides with our earlier notion

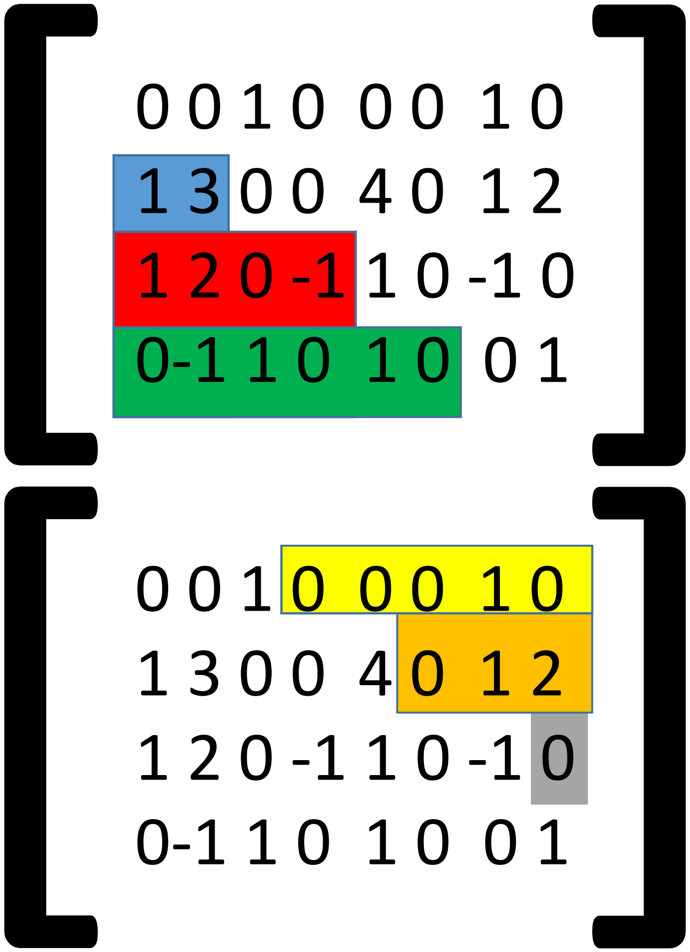
- k-Pass streaming rank
  - smallest integer r so that one can partition A into k contiguous row submatrices $A^1, \dots, A^k$ such that A^1 has streaming rank at most r, and A^j for j > 1 has streaming rank with advice at most r

- Theorem: the k-pass space complexity of computing $A \cdot x$ is $\Theta(r)$ words

k-Pass Streaming Rank Visualization



Intuition About Streaming Rank with Advice



- Streaming rank is a measure of complexity "below" a set of boundary points
- Advice captures the complexity "above" the boundary points
- Let T_i be the vector to the right of b(i)
- Try to maintain inner products of x with T_i for each i as you process the stream
- As you move from one T_i to the next, update your inner products by reading x

This motivates the following definition…

Adaptively Fitting the T_i

- A set S of vectors of F^n *adaptively fits* $T_1, \ldots, T_n$ if for each i, T_i is a linear combination of the (n-b(i))-length suffixes of vectors in S

- Inner products of x with each T_i can be generated in 1 pass using r words of memory, given inner products of x with vectors in S
  - Follows by subtracting prefix of inner products with vectors in S as you read x

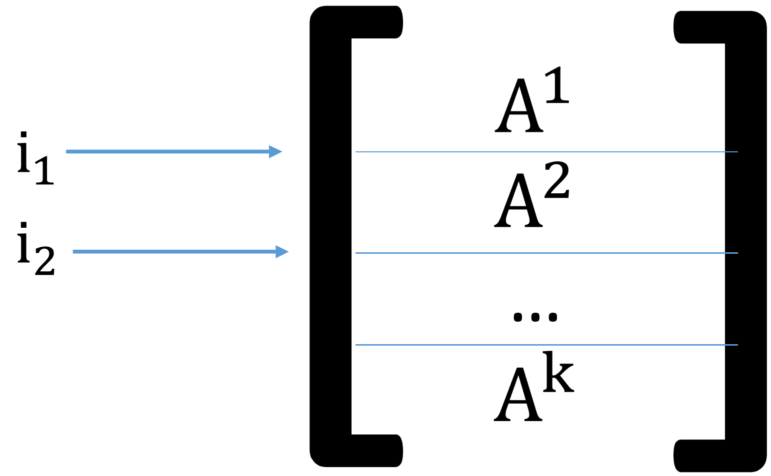
- Find an S of minimal size, given b(1), …, b(n), greedily

Streaming Rank Upper Bound for k Passes

- Preprocessing phase:

- Find "breakpoints" $i_1, \ldots, i_{k-1}$ to partition A into k contiguous matrices

- Choose breakpoints so that
  1. streaming rank of A^1 is at most r
  2. for A^j for j > 1, there are boundary points $b(1), \ldots, b(i_j - i_{j-1} + 1)$ so that $\max_i \text{rank}(B_i) \le r$ and there is a set S of r vectors which adaptively fits $T_1, \ldots, T_n$



A greedy algorithm works

Streaming Rank Lower Bound for k Passes

- Is there "better advice" than the kind we provide the algorithm with by an adaptively fitting set? Maybe non-linear advice?

No!

- If for some A^j for j > 1 there are no boundary points with $\max_i \mathrm{rank}(B_i) \leq r$ and smallest size of an adaptively fitting set at most r, then the memory required is at least r

- Proof is information-theoretic
  - "Breakpoints" and "boundary points" are defined in terms of the ouput behavior, and are random variables depending on x
  - Lower bound holds even if small local deviations in output order are permitted

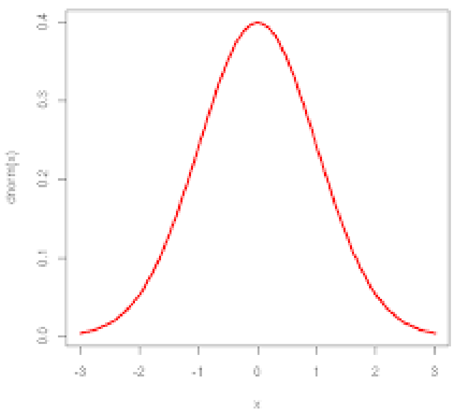
Talk Outline

1. Streaming Rank
    1. Streaming Rank 1-Pass Upper Bound
    2. Streaming Rank 1-Pass Lower Bound

2. k-Pass Streaming Rank
    1. Streaming Rank k-Pass Upper Bound
    2. Streaming Rank k-Pass Lower Bound

3. More Applications

Concrete Bounds for Matrices of Interest

$\Bigg[$  $\Bigg]$ $, P \cdot H \cdot D,$ $\begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

- If A is m x n, m < n, and is a Gaussian, Fast Hadamard Transform, or a CountSketch matrix, its k-pass streaming rank is $\Theta\left(\frac{m}{k}\right)$

High Streaming Rank for JL Transforms

- JL Transform: an m x n matrix A such that for any fixed x, $|Ax|_2^2 = (1 \pm \epsilon)|x|_2^2$ with probability at least 1-δ
- Any JL transform with $m = O\left(\epsilon^{-2}\log\left(\frac{1}{\delta}\right)\right)$ rows has streaming rank $\Omega\left(\epsilon^{-2}\log\left(\frac{1}{\delta}\right)\right)$

Maximal Separation: k Passes vs. k+1 Passes

- There is an A for which one can compute $A \cdot \mathrm{x}$ in k+1 passes using O(1) space, but any k pass algorithm requires $\Omega\left(\frac{\mathrm{n}}{\mathrm{k}}\right)$ space

Conclusion and Open Questions

- Gave tight per instance space bounds for computing $A \cdot x$ using k passes
- Question 1: generalize our results to approximate matrix product? Formalizing approximation is non-trivial
- Question 2: sometimes outputting $A \cdot x$ in a permuted order suffices. Can one efficiently find a permutation of rows of A to minimize its k-pass streaming rank?