Principal Component Analysis for Distributed Data

David Woodruff
IBM Almaden

Based on works with Ken Clarkson, Ravi Kannan, and Santosh Vempala

Outline

1. What is low rank approximation?
2. How do we solve it offline?
3. How do we solve it in a distributed setting?

Low rank approximation

- A is an n x d matrix
  - Think of n points in R^d
- E.g., A is a customer-product matrix
  - $A_{i,j}$ = how many times customer i purchased item j
- A is typically well-approximated by low rank matrix
  - E.g., high rank because of noise
- Goal: find a low rank matrix approximating A
  - Easy to store, data more interpretable

What is a good low rank approximation?

<u>Singular V</u>

Any ma

- U

s

- R

A_k = argmin$_{\text{rank k matrices B}}$ $|A-B|_F$

The rows of V_k are the top k principal components

$$\begin{pmatrix} \\ \mathbf{A} \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \mathbf{U}_k \\ \\ \end{pmatrix} \begin{pmatrix} \Sigma_k \end{pmatrix} \begin{pmatrix} \quad \mathbf{V}_k \quad \end{pmatrix} + \begin{pmatrix} \\ \mathbf{E} \\ \\ \end{pmatrix}$$

Low rank approximation

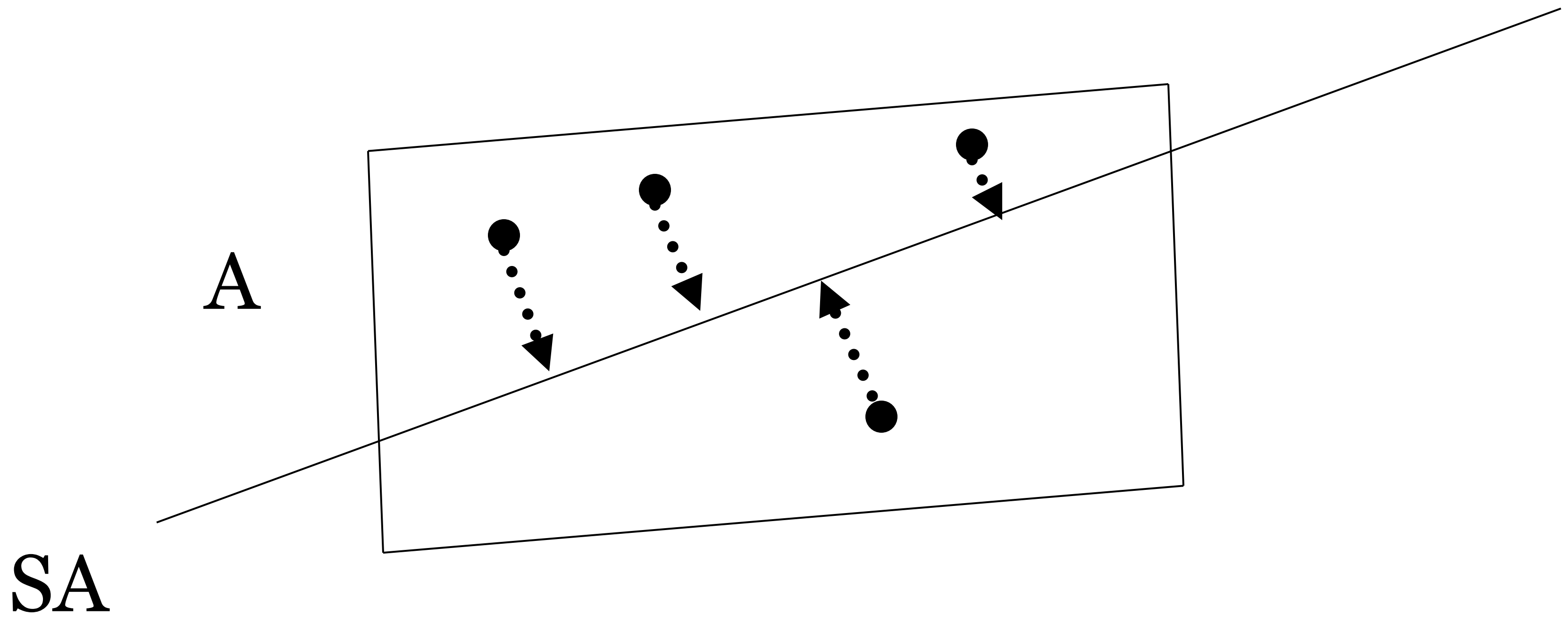
- Goal: output a rank k matrix A', so that

$$|A-A'|_F \cdot (1+\varepsilon)\ |A-A_k|_F$$

- Can do this in nnz(A) + (n+d)*poly(k/ε) time [S,CW]
  - nnz(A) is number of non-zero entries of A

Solution to low-rank approximation [S]

- Given n x d input matrix A
- Compute S*A using a sketching matrix S with k/ε << n rows. S*A takes random linear combinations of rows of A



- Project rows of A onto SA, then find best rank-k approximation to points inside of SA.

What is the matrix S?

- S can be a k/ε x n matrix of i.i.d. normal random variables
- [S] S can be a k/ε x n Fast Johnson Lindenstrauss Matrix
    - Uses Fast Fourier Transform
- [CW] S can be a poly(k/ε) x n CountSketch matrix

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

S ¢ A can be computed in nnz(A) time!

Caveat: projecting the points onto SA is slow

- Current algorithm:
  1. Compute S*A
  2. Project each of the rows onto S*A
  3. Find best rank-k approximation of projected points inside of rowspace of S*A

- Bottleneck is step 2

- [CW] Approximate the projection
  - Fast algorithm for approximate regression

$$\min_{\text{rank-k } X} |X(SA)-A|_F^2$$

  - nnz(A) + (n+d)*poly(k/ε) time

Distributed low rank approximation

- *We have fast algorithms, but can they be made to work in a distributed setting?*
- Matrix A distributed among s servers
- For t = 1, …, s, we get a customer-product matrix from the t-th shop stored in server t. Server t's matrix = A^t
- Customer-product matrix $A = A^1 + A^2 + \ldots + A^s$
- More general than row-partition model in which each customer shops in only one shop

Communication cost of low rank approximation

- Input: n x d matrix A stored on s servers
  - Server t has n x d matrix A^t
  - $A = A^1 + A^2 + \ldots + A^s$
- Output: Server t has n x d matrix C^t satisfying
  - $C = C^1 + C^2 + \ldots + C^s$ has rank at most k
  - $|A-C|_F \cdot (1+\varepsilon)|A-A_k|_F$
  - Application: distributed clustering
- Resources: Each server is polynomial time, linear space, communication is O(1) rounds. Bound the total number of words communicated
- [KVW]: $O(skd/\varepsilon)$ communication, independent of n

Protocol

- Designate one machine the Central Processor (CP)
- L
- CP … A to each server
- Server t computes A^tU

Problems:

- Can't output A^tUU^T since rank too large
- Could communicate A^tU to CP, then CP computes SVD of $\Sigma_t A^tU U^T = AUU^T$
- But communicating A^tU depends on n

Approximate SVD lemma

- Problem reduces to
  - Server t has n x r ma
  - $B = \Sigma_t B^t$
  - CP outputs top k p

Communication independent of n!

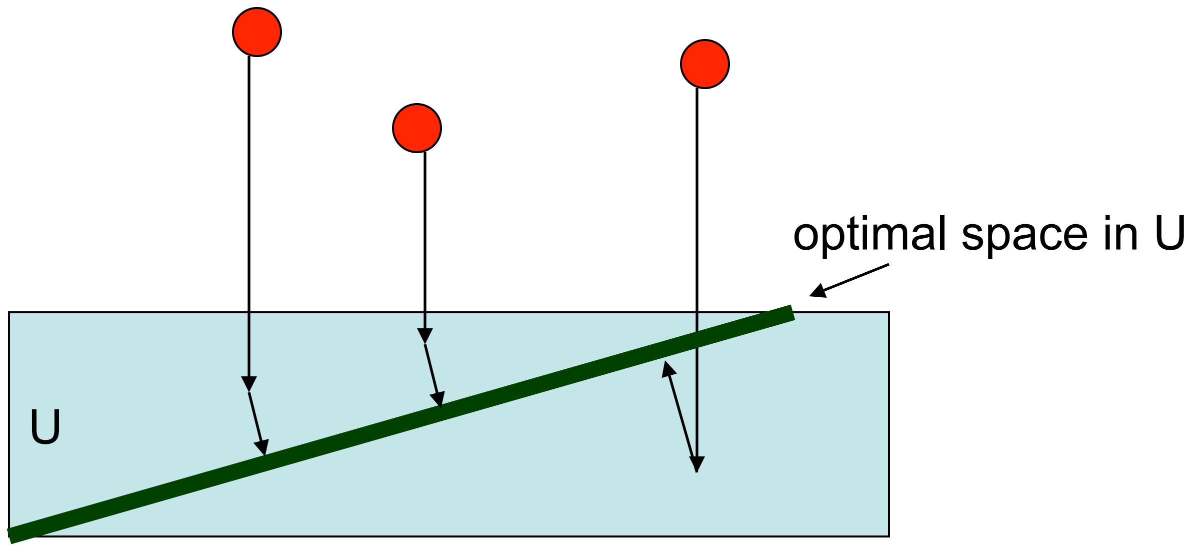
- Approximate SVD
  - If W^T 2 $R^{k \times r}$ is the matrix of top k principal components of PB, where P is a random $r/\varepsilon^2 \times n$ matrix,

$$|B-BW W^T|_F \cdot (1+\varepsilon) |B-B_k|_F$$

- CP sends P to every server
- Server t sends PB^t to CP who computes $PB = \Sigma_t PB^t$
- CP computes W, sends everyone W

The protocol

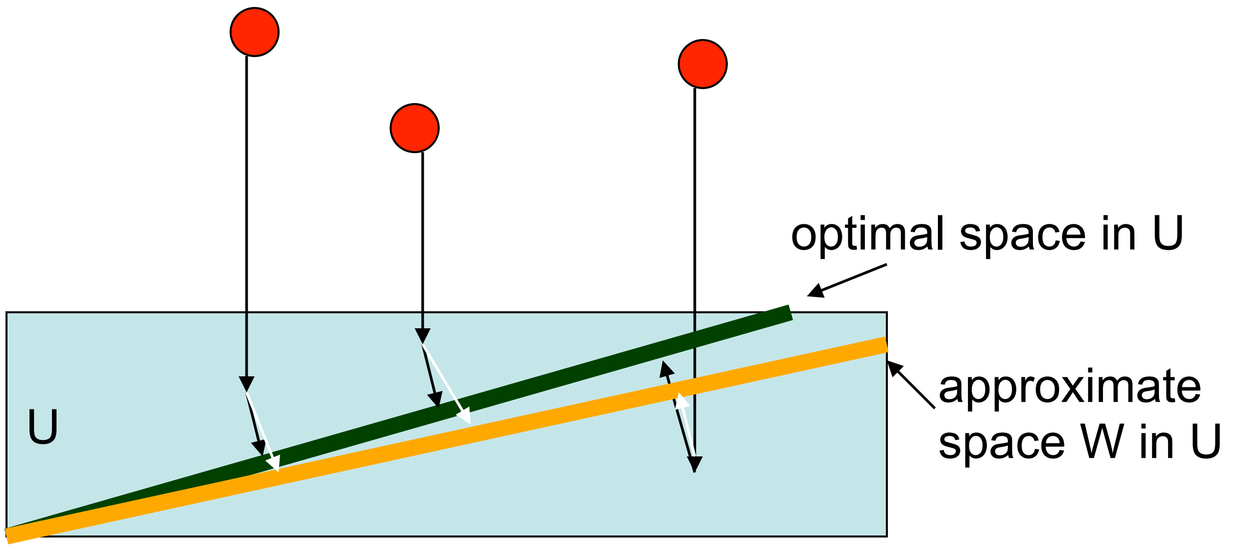
- Phase 1:
- Learn an orthonormal basis U for row space of SA



$$\text{cost} \leq (1+\varepsilon)|A-A_k|_F$$

The protocol

- Phase 2:
- Find an approximately optimal space W inside of U



cost $\leq (1+\varepsilon)^2|A-A_k|_F$

Conclusion

- $O(sdk/\varepsilon)$ communication protocol for low rank approximation

- A bit sloppy with words vs. bits but can be dealt with

- Almost matching $\Omega(sdk)$ bit lower bound
  - Can be strengthened to $\Omega(sdk/\varepsilon)$ in one-way model
  - Can we remove the one-way restriction?

- Communication cost of other optimization problems?
  - Linear programming
  - Frequency moments
  - Matching
  - etc.