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?

Singular Value Decomposition (SVD)

Any matrix $A = U \Sigma V^T$:
- $U$ has orthonormal columns
- $\Sigma$ is diagonal with non-increasing positive entries down the diagonal
- $V$ has orthonormal rows

Rank-$k$ approximation: $A_k = U_k \Sigma_k V_k^T$

$$A = \arg\min_{\text{rank } k \text{ matrices } B} \| A - B \|_F$$

Computing $A_k$ exactly is expensive

The rows of $V_k$ are the top $k$ principal components
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 $\text{nnz}(A) + (n+d)\cdot\text{poly}(k/\varepsilon)$ time [S,CW]
  - $\text{nnz}(A)$ is number of non-zero entries of $A$
Solution to low-rank approximation [S]

- Given \( n \times d \) input matrix \( A \)
- Compute \( S \cdot A \) using a sketching matrix \( S \) with \( k/\varepsilon \ll n \) rows. \( S \cdot A \) takes random linear combinations of rows of \( A \).

- Project rows of \( A \) onto \( S \cdot A \), then find best rank-\( k \) approximation to points inside of \( S \cdot A \).
What is the matrix $S$?

- $S$ can be a $k/\varepsilon \times n$ matrix of i.i.d. normal random variables

- $[S]$ $S$ can be a $k/\varepsilon \times n$ Fast Johnson Lindenstrauss Matrix
  - Uses Fast Fourier Transform

- $[CW]$ $S$ can be a $\text{poly}(k/\varepsilon) \times n$ CountSketch matrix

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

$S \preceq A$ can be computed in $\text{nnz}(A)$ 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
    \[
    \min_{\text{rank-k} X} |X(SA)-A|_F^2
    \]
  - \(\text{nnz}(A) + (n+d)\cdot\text{poly}(k/\epsilon)\) 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 $t = 1, \ldots, s$, 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:** \( n \times d \) matrix \( A \) stored on \( s \) servers
  - Server \( t \) has \( n \times d \) matrix \( A^t \)
  - \( A = A^1 + A^2 + \ldots + A^s \)

- **Output:** Server \( t \) has \( n \times d \) 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)
- Let $S$ be one of the poly($k/\varepsilon$) x $n$ random matrices above
  - $S$ can be generated pseudorandomly from small seed
  - CP chooses small seed for $S$ and sends it to all servers
- Server $t$ computes $SA_t$ and sends it to CP
- CP computes $\sum_{i=1}^{s} SA_t = SA$
- CP sends orthonormal basis $U^T$ for row space of $SA$
  - to each server
- Server $t$ computes $A_t^T U$

Problems:

- Can’t output $A_t^T U U^T$ since rank too large
- Could communicate $A_t^T U$ to CP, then CP computes SVD of $\sum_t A_t^T U U^T = AUU^T$
- But communicating $A_t^T U$ depends on $n$

CP

Server $t$ computes $A_t^T U$
Approximate SVD lemma

- Problem reduces to
  - Server $t$ has $n \times r$ matrix $B^t$
  - $B = \Sigma_t B^t$
  - CP outputs top $k$ principal components of $B$

- 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 $r/\epsilon^2 \times n$ matrix,
  \[
  |B - BW W^T|_F \cdot (1+\epsilon) |B - B_k|_F
  \]

- 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 $W$

Communication independent of $n$!
The protocol

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

\[
\text{cost} \cdot (1+\varepsilon)|A-A_k|_F
\]
The protocol

- Phase 2:

- Find an approximately optimal space $W$ inside of $U$

\[
\text{cost} \cdot (1 + \epsilon)^2 |A - A_k|_F
\]
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.