# Why do these Matrices Work?

- $\min_{\text{rank}-\ X} |XSA - A|_F^2 \leq |A_k(SA_k)^- SA - A|_F^2 \leq (1+\epsilon)|A - A_k|_F^2$

- By the normal equations,
$$|XSA - A|_F^2 = |XSA - A(SA)^- SA|_F^2 + |A(SA)^- SA - A|_F^2$$

- Hence,
$$\min_{\text{rank}-k\ X} |XSA - A|_F^2 = |A(SA)^- SA - A|_F^2 + \min_{\text{rank}-\ X} |XSA - A(SA)^- SA|_F^2$$

- Can write $SA = U\,\Sigma V^T$ in its thin SVD

- Then, $\min_{\text{rank}-k\ X} |XSA - A(SA)^- SA|_F^2 = \min_{\text{rank}-k\ X} |XU\Sigma - A(SA)^- U\Sigma|_F^2$
$$= \min_{\text{rank}-k\ Y} |Y - A(SA)^- U\Sigma|_F^2$$

- Hence, we can just compute the SVD of $A(SA)^- U\Sigma$

- But how do we compute $A(SA)^- U\Sigma$ quickly?

# 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

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

Can solve with affine embeddings

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

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

  - Want nnz(A) + (n+d)*poly(k/ε) time

# Using Affine Embeddings

- We know we can just output $\arg \min_{\text{rank}} {}_{X} |XSA - A|_F^2$
- Choose an affine embedding R:

$$|XSAR - AR|_F^2 = (1 \pm \epsilon)|XSA - A|_F^2 \text{ for all } X$$

- Note: we can compute AR and SAR in nnz(A) time
- Can just solve $\min_{\text{rank}-k\, X} |XSAR - AR|_F^2$
- $\min_{\text{ran}} {}_{X} |XSAR - AR|_F^2 = |AR(SAR)^{-}(SAR) - AR|_F^2 + \min_{\text{rank}-} {}_{X} |XSAR - AR(SAR)^{-}(SAR)|_F^2$
- Compute $\min_{\text{rank}-k\, Y} |Y - AR(SAR)^{-}(SAR)|_F^2$ using SVD which is $n \cdot \text{poly}\left(\frac{k}{\epsilon}\right)$ time
- Necessarily, $Y = XSAR$ for some X. Output $Y(SAR)^{-}SA$ in factored form. We're done!

# Low Rank Approximation Summary

1. Compute SA

2. Compute SAR and AR

3. Compute $\min_{\text{rank}-k\ Y} |Y - AR(SAR)^{-}(SAR)|_F^2$ using SVD

4. Output $Y(SAR)^{-}SA$ in factored form

Overall time: nnz(A) + (n+d)poly(k/ε)

# Course Outline

- Subspace embeddings and least squares regression
  - Gaussian matrices
  - Subsampled Randomized Hadamard Transform
  - CountSketch
- Affine embeddings
  - Application to low rank approximation
- High precision regression
- Leverage score sampling
- Distributed low rank approximation
- L1 Regression
- M-Estimator regression

# High Precision Regression

- Goal: output x' for which $|Ax'-b|_2 \leq (1+\varepsilon) \min_x |Ax-b|_2$ with high probability
- Our algorithms all have running time poly(d/ε)
- Goal: Sometimes we want running time poly(d)*log(1/ε)
- Want to make A well-conditioned
  - $\kappa(A) = \sup_{|x|_2=1} |Ax|_2 / \inf_{|x|_2=1} |Ax|_2$
- Lots of algorithms' time complexity depends on $\kappa(A)$
- Use sketching to reduce $\kappa(A)$ to O(1)!

# Small QR Decomposition

- Let S be a $(1+\epsilon_0)$- subspace embedding for A
- Compute SA
- Compute QR-factorization, $\mathrm{SA} = \mathrm{QR}^{-1}$
- Claim: $\kappa(\mathrm{AR}) = \frac{(1+\epsilon_0)}{1-\epsilon_0}$
- For all unit x, $(1-\epsilon_0)|\mathrm{ARx}|_2 \leq |\mathrm{SAR\,x}|_2 = 1$
- For all unit x, $(1+\epsilon_0)|\mathrm{ARx}|_2 \geq |\mathrm{SARx}|_2 = 1$
- So $\kappa(\mathrm{AR}) = \sup_{|x|_2=1} |\mathrm{ARx}|_2 / \inf_{|x|_2=1} |\mathrm{ARx}|_2 \leq \frac{1+\epsilon_0}{1-\epsilon_0}$

# Finding a Constant Factor Solution

- Let S be a $1+\epsilon_0$ - subspace embedding for AR
- Solve $x_0 = \underset{x}{\operatorname{argmin}} |SARx - Sb|_2$
- Time to compute $R$ and $x_0$ is nnz(A) + poly(d) for constant $\epsilon_0$
- $x_{m+1} \leftarrow x_m + R^T A^T (b - AR\, x_m)$
- $AR(x_{m+1} - x^*) = AR\,(x_m + R^T A^T (b - ARx_m) - x^*)$
  $= \left(AR - ARR^T A^T AR\right)(x_m - x^*)$
  $= U(\Sigma - \Sigma^3)V^T(x_m - x^*)$,

  where $AR = U\,\Sigma V^T$ is the SVD of AR
- $|AR(x_{m+1} - x^*)|_2 = \left|(\Sigma - \Sigma^3)V^T\left(x_m - x^*\right)\right|_2 = O(\epsilon_0)|AR(x_m - x^*)|_2$
- $|ARx_m - b|^2{}_2 = |AR(x_m - x^*)|_2^2 + |ARx^* - b|_2^2$

# Course Outline

- Subspace embeddings and least squares regression
  - Gaussian matrices
  - Subsampled Randomized Hadamard Transform
  - CountSketch
- Affine embeddings
  - Application to low rank approximation
- High precision regression
- Leverage score sampling
- Distributed low rank approximation
- M-Estimator regression

# Leverage Score Sampling

- This is another subspace embedding, but it is based on sampling!
  - If A has sparse rows, then SA has sparse rows!

- Let $A = U \Sigma V^T$ be an n x d matrix with rank d, written in its SVD

- Define the i-th leverage score $\ell(i)$ of A to be $|U_{i,*}|_2^2$

- What is $\sum_i \ell(i)$?
  - Let $(q_1, \ldots, q_n)$ be a distribution with $q_i \geq \frac{\beta \ell(i)}{d}$, where β is a parameter

- Define sampling matrix $S = D \cdot \Omega^T$, where D is k x k and $\Omega$ is n x k
  - $\Omega$ is a sampling matrix, and D is a rescaling matrix
  - For each column j of $\Omega, D$, independently, and with replacement, pick a row index i in [n] with probability $q_i$, and set $\Omega_{i,j} = 1$ and $D_{j,j} = 1/(q_i k)^{.5}$

# Leverage Score Sampling

- Note: leverage scores do not depend on choice of orthonormal basis U for columns of A
- Indeed, let U and U' be two such orthonormal bases
- Claim: $|e_i U|_2^2 = |e_i U'|_2^2$ for all i
- Proof: Since both U and U' have column space equal to that of A, we have $U = U'Z$ for change of basis matrix Z
- Since U and U' each have orthonormal columns, Z is a rotation matrix (orthonormal rows and columns)
- Then $|e_i U|_2^2 = |e_i U'Z|_2^2 = |e_i U'|_2^2$

# Leverage Score Sampling gives a Subspace Embedding

- Want to show for $S = D \cdot \Omega^T$, that $|SAx|_2^2 = (1 \pm \epsilon)|Ax|_2^2$ for all x
- Writing $A = U\,\Sigma V^T$ in its SVD, this is equivalent to showing $|SUy|_2^2 = (1 \pm \epsilon)|Uy|_2^2 = (1 \pm \epsilon)|y|_2^2$ for all y
- As usual, we can just show with high probability, $\left|U^T S^T S U - I\right|_2 \leq \epsilon$
- How can we analyze $U^T S^T S U$?
- (Matrix Chernoff) Let $X_1, \ldots, X_k$ be independent copies of a symmetric random matrix $X \in R^{dxd}$ with $E[X] = 0$, $|X|_2 \leq \gamma$, and $\left|E[X^T X]\right|_2 \leq \sigma^2$. Let $W = \frac{1}{k}\sum_{j \in [k]} X_j$. For any $\epsilon > 0$,

$$\Pr[|W|_2 > \epsilon] \leq 2d \cdot e^{-k\epsilon^2/(\sigma^2 + \frac{\gamma\epsilon}{3})}$$

(here $|W|_2 = \sup \frac{|Wx|_2}{|x|_2}$. Since W is symmetric, $|W|_2 = \sup_{|x|_2 = 1} x^T W x$.)

# Leverage Score Sampling gives a Subspace Embedding

- Let i(j) denote the index of the row of U sampled in the j-th trial
- Let $X_j = I_d - \frac{U_{i(j)}^T U_{i(j)}}{q_{i(j)}}$, where $U_{i(j)}$ is the j-th sampled row of U
- The $X_j$ are independent copies of a symmetric matrix random variable
- $E\left[X_j\right] = I_d - \sum_i q_i \left(\frac{U_i^T U_i}{q_i}\right) = I_d - I_d = 0^d$
- $\left|X_j\right|_2 \le |I_d|_2 + \frac{\left|U_{i(j)}^T U_{i(j)}\right|_2}{q_{i(j)}} \le 1 + \max_i \frac{|U_i|_2^2}{q_i} \le 1 + \frac{d}{\beta}$
- $E[X^T X] = I_d - 2E\left[\frac{U_{i(j)}^T U_{i(j)}}{q_{i(j)}}\right] + E\left[\frac{U_{i(j)}^T U_{i(j)} U_{i(j)}^T U_{i(j)}}{q_{i(j)}^2}\right]$

  $= \sum_i \frac{U_i^T U_i U_i^T U_i}{q_i} - I_d \le \left(\frac{d}{\beta}\right) \sum_i U_i^T U_i - I_d \le \left(\frac{d}{\beta} - 1\right) I_d,$

  where $A \le B$ means $x^T A x \le x^T B x$ for all x

- Hence, $|E[X^T X]|_2 \le \frac{d}{\beta} - 1$

# Applying the Matrix Chernoff Bound

- (Matrix Chernoff) Let $X_1, \ldots, X_k$ be independent copies of a symmetric random matrix $X \in R^{dxd}$ with $E[X] = 0$, $|X|_2 \leq \gamma$, and $\left|E[X^T X]\right|_2 \leq \sigma^2$. Let $W = \frac{1}{k}\sum_{j\in[k]} X_j$. For any $\epsilon > 0$,

$$\Pr[|W|_2 > \epsilon] \leq 2d \cdot e^{-k\epsilon^2/(\sigma^2 + \frac{\gamma\epsilon}{3})}$$

(here $|W|_2 = \sup \frac{|Wx|_2}{|x|_2}$. Since W is symmetric, $|W|_2 = \sup_{|x|_2=1} x^T W x$.)

- $\gamma = 1 + \frac{d}{\beta}$, and $\sigma^2 = \frac{d}{\beta} - 1$

- $X_j = I_d - \frac{U_{i(j)}^T U_{i(j)}}{q_{i(j)}}$, and recall how we generated $S = D \cdot \Omega^T$: For each column j of $\Omega, D$, independently, and with replacement, pick a row index i in [n] with probability $q_i$, and set $\Omega_{i,j} = 1$ and $D_{j,j} = 1/(q_i k)^{.5}$
  - Implies $W = I_d - U^T S^T S U$

- $\Pr\left[\left|I_d - U^T S^T S U\right|_2 > \epsilon\right] \leq 2d \cdot e^{-k\epsilon^2 \Theta\left(\frac{\beta}{d}\right)}$. Set $k = \Theta(\frac{d \log d}{\beta \epsilon^2})$ and we're done.

# Fast Computation of Leverage Scores

- Naively, need to do an SVD to compute leverage scores
- Suppose we compute $SA$ for a subspace embedding S
- Let $SA = QR^{-1}$ be such that Q has orthonormal columns
- Set $\ell_i' = |e_i AR|_2^2$
- Since AR has the same column span of A, $AR = UT^{-1}$
  - $(1-\epsilon)|ARx|_2 \leq |SARx|_2 = |x|_2$
  - $(1+\epsilon)|ARx|_2 \geq |SARx|_2 = |x|_2$
  - $(1 \pm O(\epsilon))|x|_2 = |ARx|_2 = |UT^{-1}x|_2 = |T^{-1}x|_2,$
- $\ell_i = |e_i ART|_2^2 = (1 \pm O(\epsilon))|e_i AR|_2^2 = (1 \pm O(\epsilon))\ell_i'$
- But how do we compute AR? We want nnz(A) time

# Fast Computation of Leverage Scores

- $\ell_i = (1 \pm O(\epsilon))\ell_i'$
- Suffices to set this $\epsilon$ to be a constant
- Set $\ell_i' = |e_i A R|_2^2$
  - This takes too long
- Let $G$ be a d x O(log n) matrix of i.i.d. normal random variables
  - For any vector z, $\Pr[|zG|_2^2 = \left(1 \pm \frac{1}{2}\right)|z|^2] \geq 1 - \frac{1}{n^2}$
- Instead set $\ell_i' = |e_i A R G|_2^2$.
  - Can compute in (nnz(A) + $d^2) \log n$ time
- Can solve regression in nnz(A) log n + poly(d(log n)/ε) time

# Course Outline

- Subspace embeddings and least squares regression
  - Gaussian matrices
  - Subsampled Randomized Hadamard Transform
  - CountSketch
- Affine embeddings
  - Application to low rank approximation
- High precision regression
- Leverage score sampling
- Distributed low rank approximation
- L1 Regression
- M-Estimator regression

# Distributed low rank approximation

- *We have fast algorithms for low rank approximation, but can they be made to work in a distributed setting?*

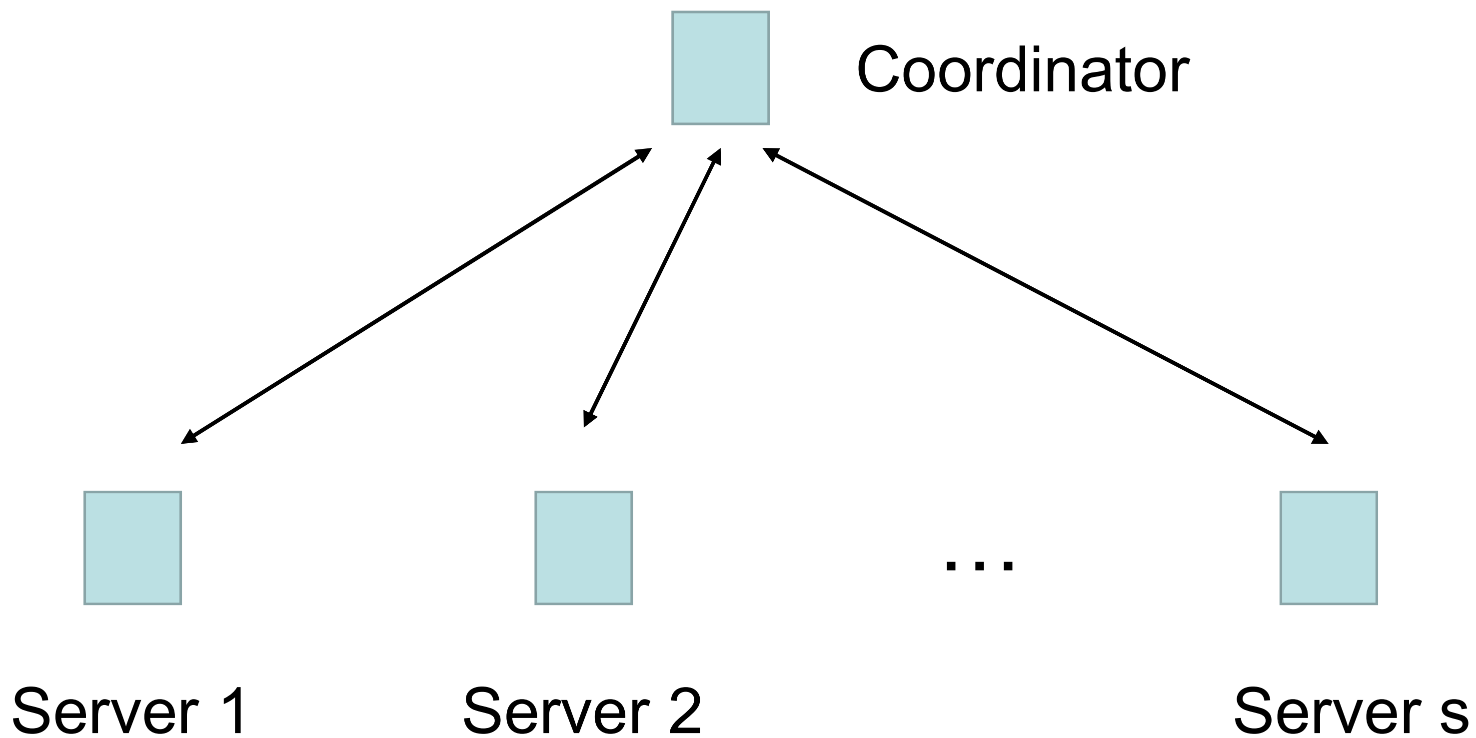
- Matrix A distributed among s servers

- 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 $A = A^1 + A^2 + \ldots + A^s$
  - Model is called the arbitrary partition model

- More general than the row-partition model in which each customer shops in only one shop

# The Communication Model




- Each player talks only to a Coordinator via 2-way communication
- Can simulate arbitrary point-to-point communication up to factor of 2 (and an additive O(log s) factor per message)

# Communication cost of low rank approximation

- Input: n x d matrix A stored on s servers
  - Server t has n x d matrix $A^t$
  - $A = A^1 + A^2 + \ldots + A^s$
  - Assume entries of $A^t$ are O(log(nd))-bit integers
- Output: Each server outputs the same k-dimensional space W
  - $C = A^1 P_W + A^2 P_W + \ldots + A^s P_W$, where $P_W$ is the projection onto W
  - $|A-C|_F \leq (1+\varepsilon)|A-A_k|_F$
  - Application: k-means clustering
- Resources: Minimize total communication and computation. Also want O(1) rounds and input sparsity time

# Work on Distributed Low Rank Approximation

- [FSS]: First protocol for the row-partition model.
  - $O(sdk/\varepsilon)$ real numbers of communication
  - Don't analyze bit complexity (can be large)
  - SVD Running time, see also [BKLW]
- [KVW]: $O(skd/\varepsilon)$ communication in arbitrary partition model
- [BWZ]: $O(skd) + \text{poly}(sk/\varepsilon)$ words of communication in arbitrary partition model. Input sparsity time
  - Matching $\Omega(skd)$ words of communication lower bound
- Variants: kernel low rank approximation [BLSWX], low rank approximation of an implicit matrix [WZ], sparsity [BWZ]

# Outline of Distributed Protocols

- [FSS] protocol
- [KVW] protocol
- [BWZ] protocol

# Constructing a Coreset [FSS]

- Let $A = U\,\Sigma V^T$ be its SVD
- Let m = k + $k/\epsilon$
- Let $\Sigma_m$ agree with $\Sigma$ on the first m diagonal entries, and be 0 otherwise
- Claim: For all projection matrices Y=I-X onto (d-k)-dimensional subspaces,

$$\left|\Sigma_m V^T Y\right|_F^2 + c \ = (1 \pm \epsilon)|AY|_F^2,$$

where $c = |A - A_m|_F^2$ does not depend on Y

- We can think of S as $U_m^T$ so that $SA = U_m^T U \Sigma V^T = \Sigma_m V^T$ is a sketch

# Constructing a Coreset

- Claim: For all projection matrices Y=I-X onto (d-k)-dimensional subspaces,

$$\left|\Sigma_m V^T Y\right|_F^2 + c \; = (1 \pm \epsilon)|AY|_F^2,$$

where $c = |A - A_m|_F^2$ does not depend on Y

- Proof: $|AY|_F^2 = \left|U\Sigma_m V^T Y\right|_F^2 + \left|U(\Sigma - \Sigma_m)V^T Y\right|_F^2$

$$\leq \left|\Sigma_m V^T Y\right|_F^2 + |A - A_m|_F^2 = \left|\Sigma_m V^T Y\right|_F^2 + c$$

Also, $\left|\Sigma_m V^T Y\right|_F^2 + |A - A_m|_F^2 \; - |AY|_F^2$

$$= \left|\Sigma_m V^T\right|_F^2 - \left|\Sigma_m V^T X\right|_F^2 + |A - A_m|_F^2 - |A|_F^2 + |AX|_F^2$$

$$= |AX|_F^2 - \left|\Sigma_m V^T X\right|_F^2$$

$$= \left|(\Sigma - \Sigma_m)V^T X\right|_F^2$$

$$\leq \left|(\Sigma - \Sigma_m)V^T\right|_2^2 \cdot |X|_F^2$$

$$\leq \sigma_{m+1}^2 k \leq \epsilon\, \sigma_{m+1}^2 (m-k) \leq \epsilon \textstyle\sum_{i \in \{k+1,..,m+1\}} \sigma_i^2 \leq \epsilon |A - A_k|_F^2$$