Randomization Methods for Low-Rank Matrix Approximation

James A. Nguyen
University of California, Irvine

March 11, 2025

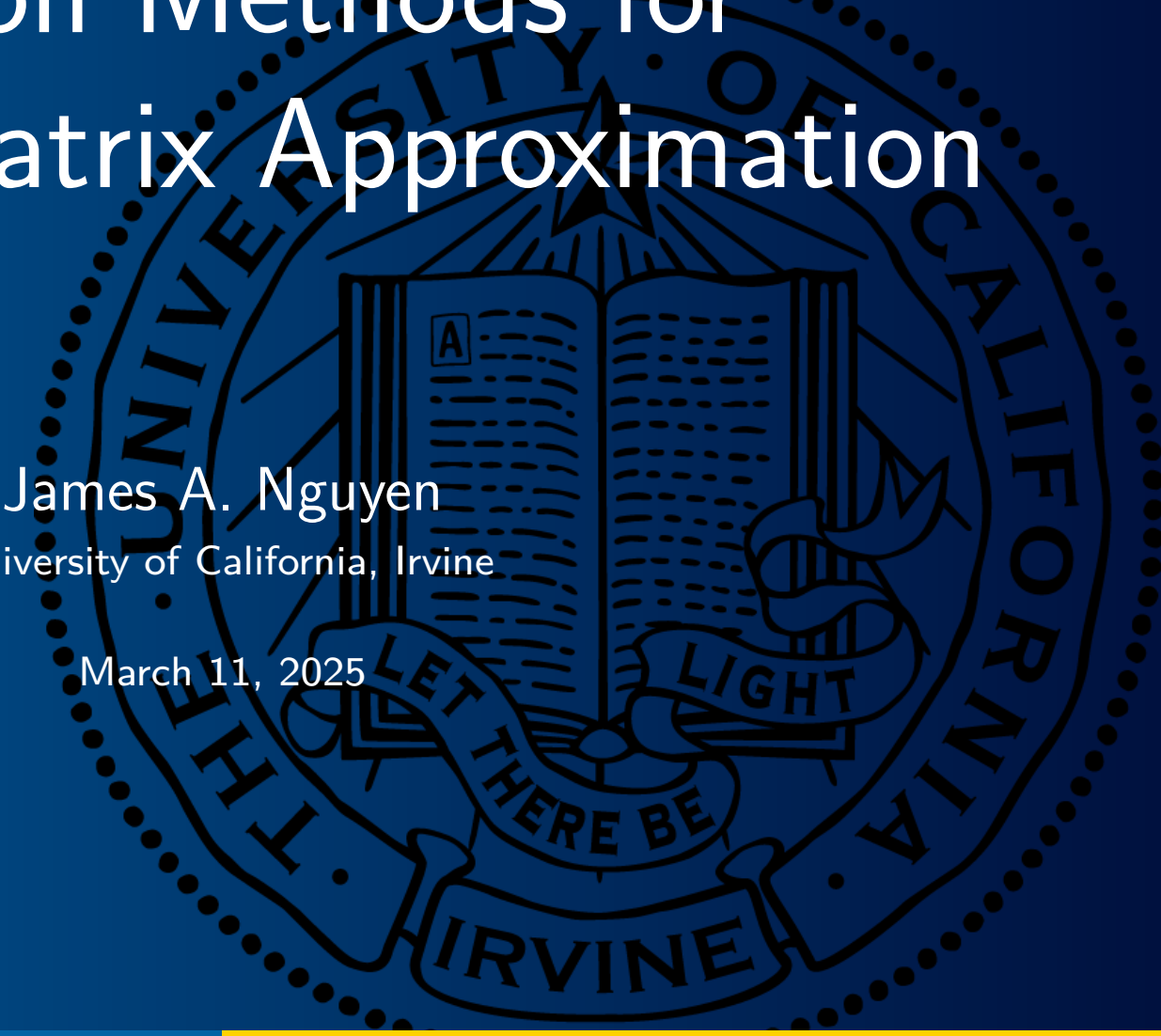


Table of Contents (Summary)

Problem Outline

Advantages of a low-rank matrix approximation?

Consider a full-rank matrix $A \in \mathbb{C}^{m \times n}$ with condition number $\kappa(A)$

- We've discussed: Singular Value Decomposition to reveal numerical rank $k = \underset{\sigma_i < \epsilon}{max}\ i$
- Construct some orthonormal basis $Q \in \mathbb{C}^{m \times k}$ s.t. $A \approx QQ^*A$
  - $\kappa(Q^*A) < \kappa(A)$
- Computationally more efficient
  - Consider image compression vs. video processing

Fixed-Rank Approximation

In the simplest case, we consider the fixed-rank problem:

Fixed-Rank Problem

Given desired rank k, obtain a matrix $Q \in \mathbb{C}^{m \times k}$ such that $range(Q)$ approximates the action of A, i.e.

$$||A - QQ^*A|| \leq \epsilon$$

How:

1. Draw random vectors $Y = A\Omega$ a random matrix Ω
2. Take the QR factorization $Y = QR$ to obtain an orthonormal basis Q for the range of Y

Fixed-Rank Approximation

In the simplest case, we consider the fixed-rank problem:

Fixed-Rank Problem

Given desired rank k, obtain a matrix $Q \in \mathbb{C}^{m \times k}$ such that $range(Q)$ approximates the action of A, i.e.

$$||A - QQ^*A|| \leq \epsilon$$

Input: $A \in \mathbb{C}^{m \times n}$, k
Output: Q

$\Omega = randn(n, k)$
$[Q, \sim] = qr(A\Omega)$

Important Considerations

Randomization implies some variance in our output.
We reduce this variance by **oversampling**.

Consider, there exists a matrix $Q \in \mathbb{C}^{m \times k}$ that minimizes

$$||A - QQ^*A|| = \min_{rank(X) \leq k} ||A - X||$$

We almost surely won't construct this exact matrix.
Instead, choose some small oversampling parameter p s.t. $Q \in \mathbb{C}^{m \times l}$, where $l = k + p$, closely approximates the minimal solution.

In general, p dependent on size of A, rate of singular value decay, choice of sampling distribution.

Adjusting for Slow Singular Decay

Revisiting the first algorithm that retrieves: $[Q, \sim] = qr(A\Omega)$

- We typically describe the conditioning of a matrix by $\kappa(A)$
- These algorithms highly dependent on rate of singular value decay
  - Perform particularly well when singular values decay rapidly

Solution: Power iteration % Naively: $[Q, \sim] = qr((AA^*)^q A\Omega)$

- susceptible to round-off errors

Adjusting for Slow Singular Decay

Revisiting the first algorithm that retrieves: $[Q, \sim] = qr(A\Omega)$

- We typically describe the conditioning of a matrix by $\kappa(A)$
- These algorithms highly dependent on rate of singular value decay
  - Perform particularly well when singular values decay rapidly

Solution: Power iteration % Naively: $[Q, \sim] = qr((AA^*)^q A\Omega)$

- susceptible to round-off errors

New Algorithm: Randomized Subspace Iteration

$Q_0 = qr(A\Omega)$
$\Omega = randn(n, l)$ % from last slide: l = k + p
for $i = 1, \ldots, q$
$[\tilde{Q}_i, \sim] = qr(A^* Q_{i-1})$
$[Q_i, \sim] = qr(A\tilde{Q}_{i-1})$ % orthonormalize to avoid round-off errors
return Q_q % algebraically equivalent to power iteration

Fixed-Precision

Most commonly, numerical rank k is unknown. Then, we adaptively construct Q until error tolerance satisfied. i.e. We add columns until $Q^{(l)} \in \mathbb{C}^{m \times l}$ satisfies:

$$||(I - Q^{(l)}(Q^{(l)})^*)A|| \leq \epsilon,$$

where l will likely be slightly greater than k, the minimal size.

Choosing some reliability parameter r, we bound the ith error term by the max norm of the r most recent samples from the range of A:

$$||(I - Q^{(l)}(Q^{(l)})^*)A|| \leq 10\sqrt{\frac{2}{\pi}} \underset{i=1,\ldots,r}{max} ||(I - Q^{(l)}(Q^{(l)})^* A\omega^{(i)}||$$

Fixed Precision Algorithm

Input: $A \in \mathbb{C}^{m \times n},\ \epsilon,\ r = 10$

Output: $Q \in \mathbb{C}^{m \times l}$ with $P(||(I - QQ^*)A||) \geq 1 - m10^{-r}$

Generate Gaussian vectors $\omega^{(1)}, \ldots, \omega^{(r)}$ of length n.

Compute $y(i) = A\omega^{(i)}$ % sample from range(A)

$j = 0; Q^{(0)} = [\ ]$

while $\underset{j+1 \leq i \leq j+r}{max} ||y^{(i)}|| > \epsilon/(10\sqrt{\frac{2}{\pi}})$

$j = j + 1$

$y^{(j)} \leftarrow (I - Q^{(j-1)}(Q^{(j-1)})^*)y^{(j)}$ % orthogonalize to $span(QQ^*A)$

$q^{(j)} = y^{(j)}/||y^{(j)}||$ % normalize column

append $Q^{(j)} \leftarrow [Q^{(j-1)}\ q^{(j)}]$

Generate Gaussian vector $\omega^{(j+r)}$

$y^{(j+r)} = (I - Q^{(j)}(Q^{(j)})^*)A\omega^{(j+r)}$

for $i = j + 1 : j + r - 1$

$y^{(i)} \leftarrow y^{(i)} - q^{(j)}\langle q^{(j)}, y^{(i)} \rangle$

return $Q^{(j)}$

Fixed Precision Algorithm

Input: $A \in \mathbb{C}^{m \times n},\ \epsilon,\ r = 10$

Output: $Q \in \mathbb{C}^{m \times l}$ with $P(||(I - QQ^*)A||) \geq 1 - m10^{-r}$

$\Omega = randn(n, r)$ % draw n Gaussian vectors

$Y = A\Omega$ % sample range(A)

$j = 0; Q^{(0)};$

while maxnorm(last 10 columns of Y) $> \epsilon/(10\sqrt{\frac{2}{\pi}})$

$j = j + 1$

$Y_{:,j} = Y_{:,j} - Q^{(j-1)}(Q^{(j-1)})^* Y_{:,j}$ % orthogonalize to $span(QQ^*A)$

$q^{(j)} = Y_{:,j}/||Y_{:,j}||$ % normalize column

$Q^{(j)} \leftarrow [Q^{(j-1)}\ q^{(j)}]$

Generate Gaussian vector $\omega^{(j+r)}$

$Y_{:,j} = A\omega^{(j+r)}$ − component in $range(QQ^*A)$

for $i = j + 1 : j + r - 1$

$Y_{:,i} \leftarrow Y_{:,i} - q^{(j)}\langle q^{(j)}, Y_{:,i}\rangle$ % project sample vectors onto $q^{(j)}$

return Q_j

Extension to SVD

Generally, the described algorithms generate Q very efficiently. Then, complexity of obtaining SVD reduced to computing $svd(Q^*A)$

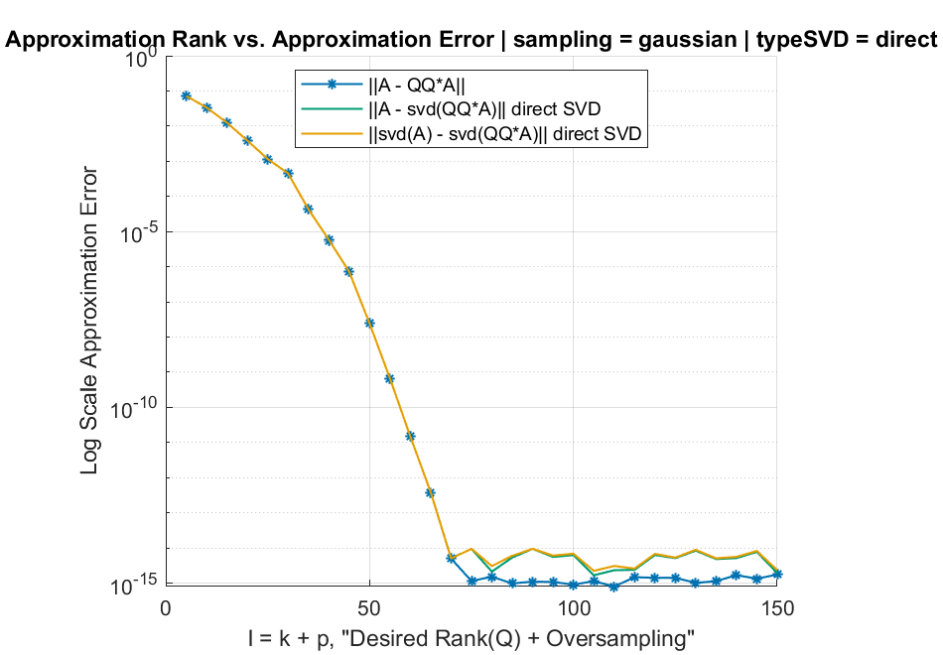
- Idea: $svd(QQ^*A) = Q(svd(Q^*A)) + O(\epsilon_{machine})$

Input: A, Q
Output: U, Σ, V satisfying $||U\Sigma V^* - A|| \leq \epsilon$
$[\tilde{U}, \Sigma, V] = svd(Q^*A)$
$U = Q\tilde{U}$ % recover original matrix dimensions
return U, Σ, V

Complexity Reduction $(m \leq n)$

- $O(mn^2) \rightarrow O(mn\log(l))$

Numerical Results



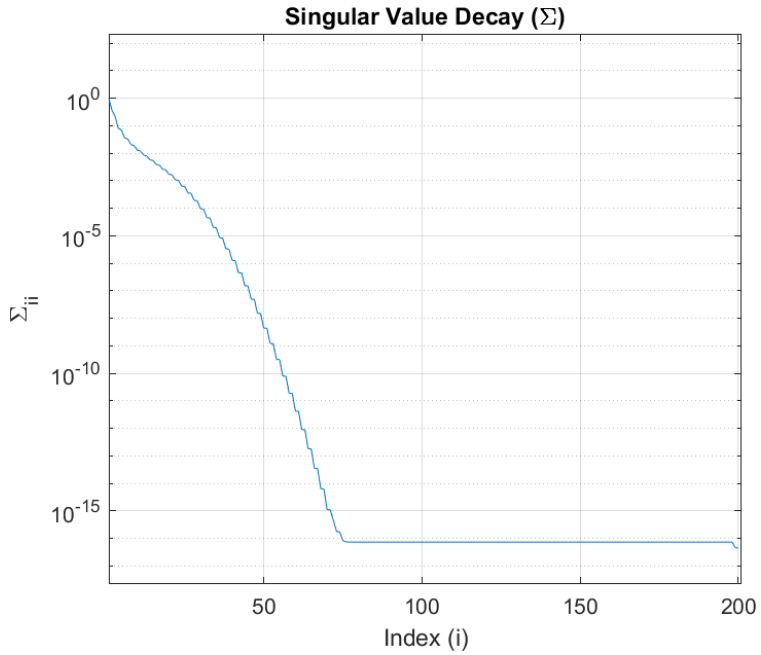


Figure:

Randomly Samples Low-Rank Matrix Approximation of discretized Laplace Integral matrix. Basis approximated at each of numerical ranks $k \in \{5z : \mathbb{Z}^{+}_{\leq 30}\}$.

Parameters: m = 200; n = 200; p = 0

Thank you

Thank you.

[1] [2]

N. Halko, P. G. Martinsson, and J. A. Tropp.
Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions.
SIAM Review, 53(2):217–288, 2011.

Vladimir Rokhlin, Arthur Szlam, and Mark Tygert.
A randomized algorithm for principal component analysis, 2009.