Stochastic SVD on Hadoop

Shannon Quinn

(with thanks to Gunnar Martinsson and Nathan Halko of UC Boulder, and Joel Tropp of CalTech)

Lecture breakdown

- Part I
  - Stochastic SVD

- Part II
  - **Distributed** stochastic SVD

Part I: Stochastic SVD

Basic goal

- Matrix A
  - Find a low-rank approximation of A
  - Basic dimensionality reduction

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

Basic algorithm

- INPUT: A, k, p
- OUTPUT: Q

1. Draw Gaussian n x (k + p) test matrix Ω
2. Form product $Y = A\Omega$
3. Orthogonalize columns of Y ➔ Q

Basic evaluation

$$\mathbb{E}\|\mathsf{A} - \mathsf{Q}\mathsf{Q}^{\mathsf{T}}\mathsf{A}\|_2 \leq \left(1 + \sqrt{\frac{k}{p-1}}\right)\sigma_{k+1} + \frac{e\sqrt{k+p}}{p} \cdot \left(\sum_{j>k} \sigma_j^2\right)^{1/2}$$

$$\leq \left[1 + \frac{4\sqrt{k+p}}{p-1} \cdot \sqrt{\min\{m,n\}}\right]\sigma_{k+1}$$

$$= C \cdot \sigma_{k+1}.$$

Approximating the SVD

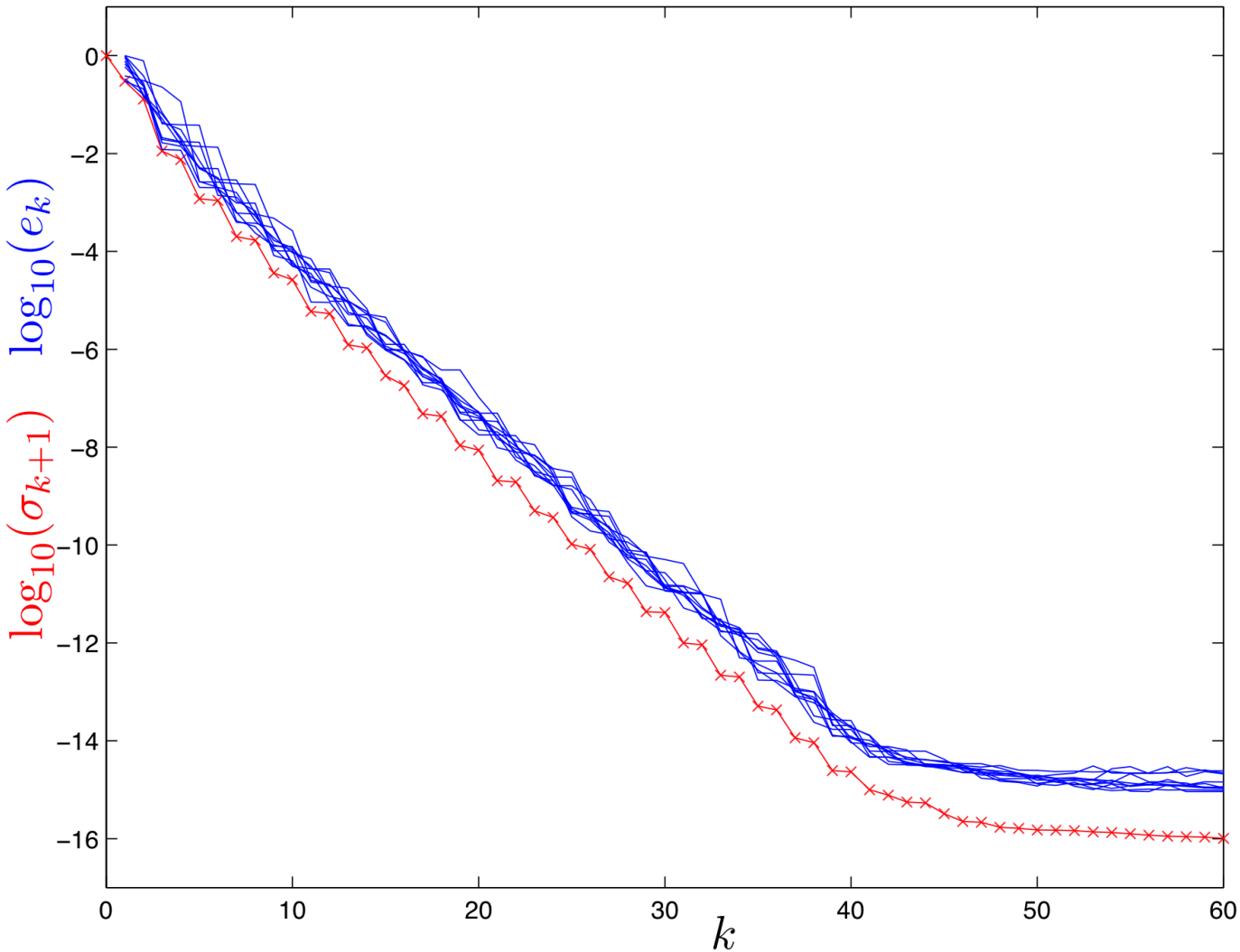
- INPUT: Q
- OUTPUT: Singular vectors U

1. Form k x n matrix $\mathrm{B} = \mathrm{Q}^{\mathrm{T}}\mathrm{A}$
2. Compute SVD of $\mathrm{B} = \hat{\mathrm{U}}\Sigma\mathrm{V}^{\mathrm{T}}$
3. Compute singular vectors $\mathrm{U} = \mathrm{Q}\hat{\mathrm{U}}$

Demo

Empirical Results

- 1000x1000 matrix



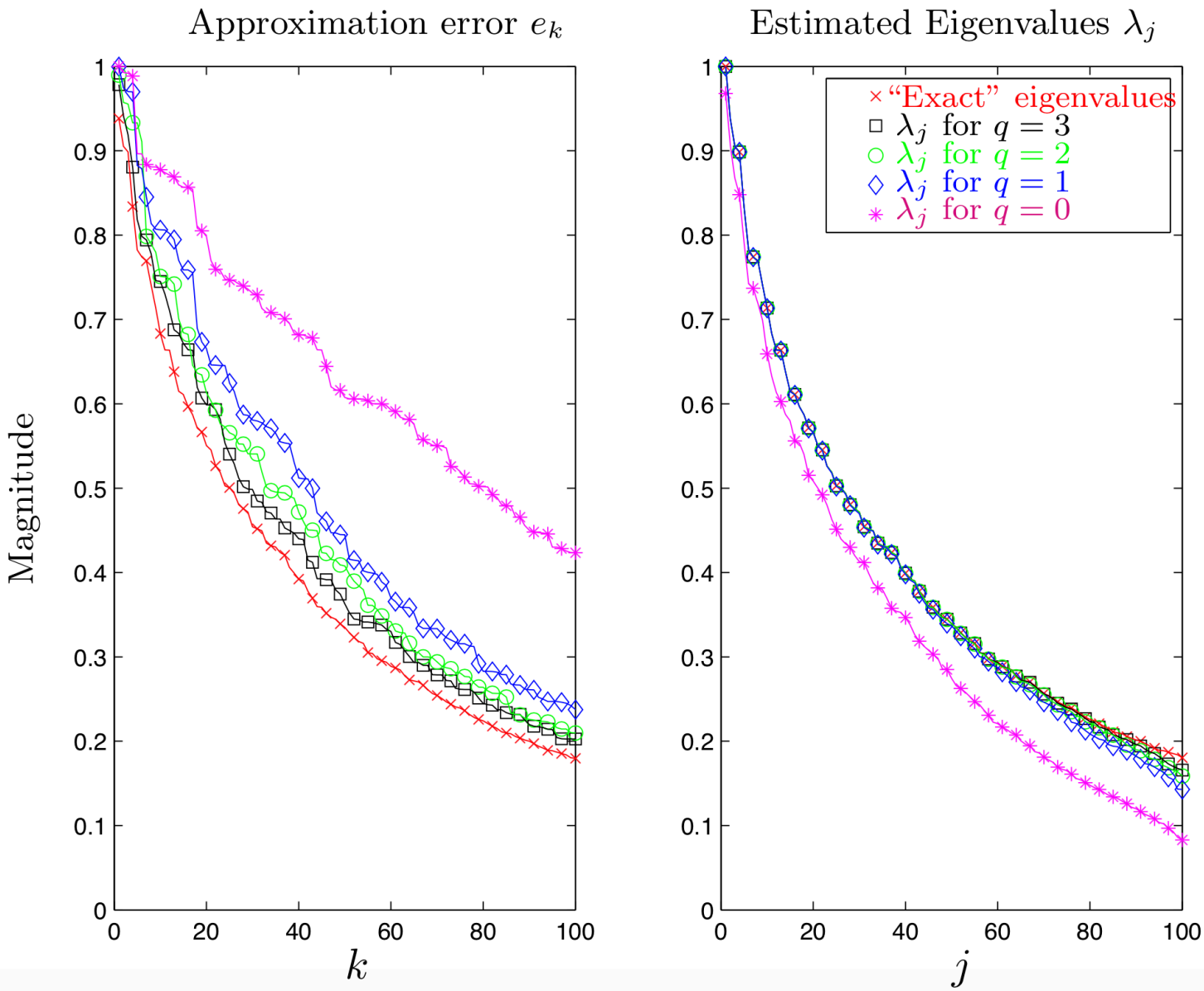
Power iterations

- Affects decay of eigenvalues / singular values

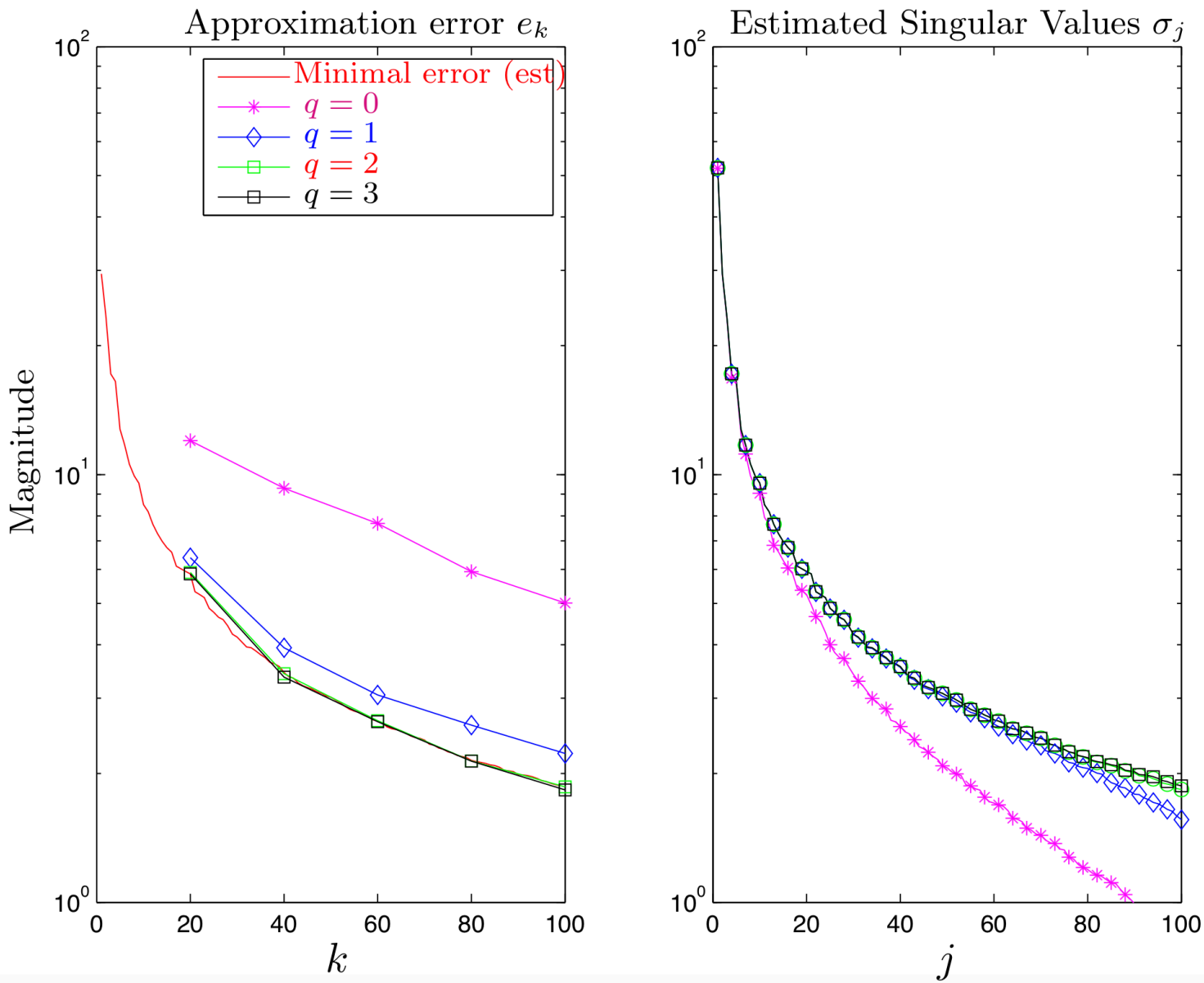
$$\mathsf{Y} = \mathsf{A}\,\Omega.$$

$$\mathsf{Y} = (\mathsf{A}\,\mathsf{A}^*)^q\,\mathsf{A}\,\Omega$$

Empirical Results



Empirical Results



Part II: Distributed SSVD

Algorithm Overview

ALGORITHM 4.3: STOCHASTIC SINGULAR VALUE DECOMPOSITION

Given an $m \times n$ matrix A, a target rank k, an oversampling parameter p, and a number of power iterations q, the following algorithm computes an approximate rank k singular value decomposition $\mathsf{A} \approx \mathsf{U}\Sigma\mathsf{V}^T$.

Draw an $n \times k + p$ random matrix Ω.
Form the product $\mathsf{Y} = \mathsf{A}\Omega$.
Orthogonalize the columns of $\mathsf{Y} \to \mathsf{Q}$. — QR factorization
for $i = 1..q$
  Form the product $\mathsf{Y} = \mathsf{A}\mathsf{A}^T\mathsf{Q}$. — Power iteration
  Orthogonalize the columns of $\mathsf{Y} \to \mathsf{Q}$. — QR factorization
end
Form the projection $\mathsf{B} = \mathsf{Q}^T\mathsf{A}$.
Compute the factorization $\tilde{\mathsf{U}}\Sigma^2\tilde{\mathsf{U}}^T = \mathsf{B}\mathsf{B}^T$. — In-core SVD
Solve $\tilde{\mathsf{V}}^T = \Sigma^{-1}\tilde{\mathsf{U}}^T\mathsf{B}$.
Set $\mathsf{U} = \mathsf{Q}\tilde{\mathsf{U}}(:, 1{:}\,k)$.
Set $\mathsf{V} = \tilde{\mathsf{V}}(:, 1{:}\,k)$.

SSVD Primitives

- Matrix-vector multiplication: y = Ax

ALGORITHM 4.5: MATRIX MULTIPLICATION $\mathsf{A}\mathbf{x}$

This algorithm forms the product $\mathbf{y} = \mathsf{A}\mathbf{x}$ *assuming* A *is stored in row major format.*

Map

Iterate A_{row}

$\mathbf{y}_{row} = \langle \mathsf{A}_{row}, \mathbf{x} \rangle$

output $\mathbf{y}$

- (midterm, anyone?)

SSVD Primitives

- Matrix-matrix multiplication: $y = A^T A x$

ALGORITHM 4.5: MATRIX MULTIPLICATION $A^T A \mathbf{x}$

This algorithm forms the product $\mathbf{y} = A^T A \mathbf{x}$ *assuming* A *is stored in row major format.*

Map

Iterate A_{row}

$\mathbf{y}_{partial} = \langle A_{row}, \mathbf{x} \rangle \cdot A_{row}^T$

output $\mathbf{y}_{partial}$

Reduce

$\mathbf{y} = \sum \mathbf{y}_{partial}$

output $\mathbf{y}$

Matrix-matrix multiplication

- Very clever use of map/reduce

$$\begin{aligned} \mathsf{C}\mathbf{x} &= \textstyle\sum_{i=1}^{M} \mathsf{A}_i^T \left(\mathsf{A}_i \mathbf{x}\right) \\ &= \textstyle\sum_{i=1}^{M} \mathbf{v}^{(i)} \\ &= \mathbf{y} \end{aligned}$$

- Each Mapper outputs: $< j, \mathbf{v}_j^{(i)} >$

SSVD Primitives

- Distributed orthogonalization: $Y = A\Omega$
  - Givens rotation

$$G(i,j,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

  - Streaming QR
    - Sliding window
  - Merge factorizations
    1. Merge R
    2. Merge Q^T

SSVD

ALGORITHM 4.7: STOCHASTIC SINGULAR VALUE DECOMPOSITION

The ssvd algorithm produces rank k matrices $\mathsf{U}, \mathsf{V}, \Sigma$ that form an approximate singular value decomposition of matrix A.

1 Q-JOB
2 BT-JOB
for $i = 1..q$
3 ABT-JOB
BT-JOB
end for
Serial step:
compute $\tilde{\mathsf{U}}\Sigma^2\tilde{\mathsf{U}}^T = \mathsf{B}\mathsf{B}^T$
4 U-JOB
5 V-JOB

1: Q-job

ALGORITHM 4.7.1: Q-JOB

This algorithm forms the product $\mathsf{Y} = \mathsf{A}\Omega$, *performs the streaming* QR *factorization and the first level merge.*

Map

Iterate A_{row}

$\mathsf{Y}_{row} = \mathsf{A}_{row} \cdot \Omega$

$\mathsf{Y}_{row} \rightarrow streaming\mathsf{QR}$

output $\mathsf{Q}_i^{r \times \ell}$ and $\mathsf{R}_i^{\ell \times \ell}$.

$merge\{\mathsf{Q}_i, \mathsf{R}_i\}_{i=1}^{z}$

output $\mathsf{Q}^{s \times \ell}\mathsf{R}^{\ell \times \ell}$

2: B^T-job

ALGORITHM 4.7.2: BT-JOB

This algorithm completes the second level merge and computes the product $\mathsf{B}^T = \mathsf{A}^T\mathsf{Q}$. *Optionally, partial sums of* $\mathsf{B}\mathsf{B}^T$ *are formed and output to disk.*

Map

$\mathsf{Q}_i \leftarrow$ merge $\tilde{\mathsf{Q}}_i, \tilde{\mathsf{R}}_1, \ldots, \tilde{\mathsf{R}}_M$

output Q_i as block of final Q.

Iterate A_{row}

$\mathsf{B}^T_{partial} = (\mathsf{A}_{row})^T \cdot \mathsf{Q}_{row}$

output $\mathsf{B}^T_{partial}$

Reduce

$\mathsf{B}^T = \sum \mathsf{B}^T_{partial}$

Option

$(\mathsf{B}\mathsf{B}^T)_{partial} = \mathsf{B}_i\mathsf{B}_i^T$

3: AB^T-job

ALGORITHM 4.7.3: ABT-JOB

This algorithm computes the product AB^T*, performs the streaming* QR *factorization and the first level merge.*

Map

Iterate A_{row}

$\mathsf{A}_{block}(i,:) = \mathsf{A}_{row}$

foreach B^T_{row}

$\mathsf{Y}_{partial} = \mathsf{A}_{block}(:,j) \cdot \mathsf{B}^T_{row}$

output $\mathsf{Y}_{partial}$

Reduce

$\mathsf{Y} = \sum \mathsf{Y}_{partial}$

$\mathsf{Y} \rightarrow streaming\mathsf{QR}$

output $\mathsf{Q}_i^{r\times\ell}$ and $\mathsf{R}_i^{\ell\times\ell}$

$merge\{\mathsf{Q}_i, \mathsf{R}_i\}_{i=1}^{z}$

output $\mathsf{Q}^{s\times\ell}\mathsf{R}^{\ell\times\ell}$

4: U-job

ALGORITHM 4.7.4: U-JOB

This algorithm computes the rank k factor U of the singular value decomposition

Map

Iterate Q_{row}

$\mathsf{U}_{row} = \mathsf{Q}_{row}\tilde{\mathsf{U}}$

output U_{row}

5:V-job

ALGORITHM 4.7.5: V-JOB

This algorithm computes the rank k factor V *of the singular value decomposition*

Map

Iterate B_{row}^T

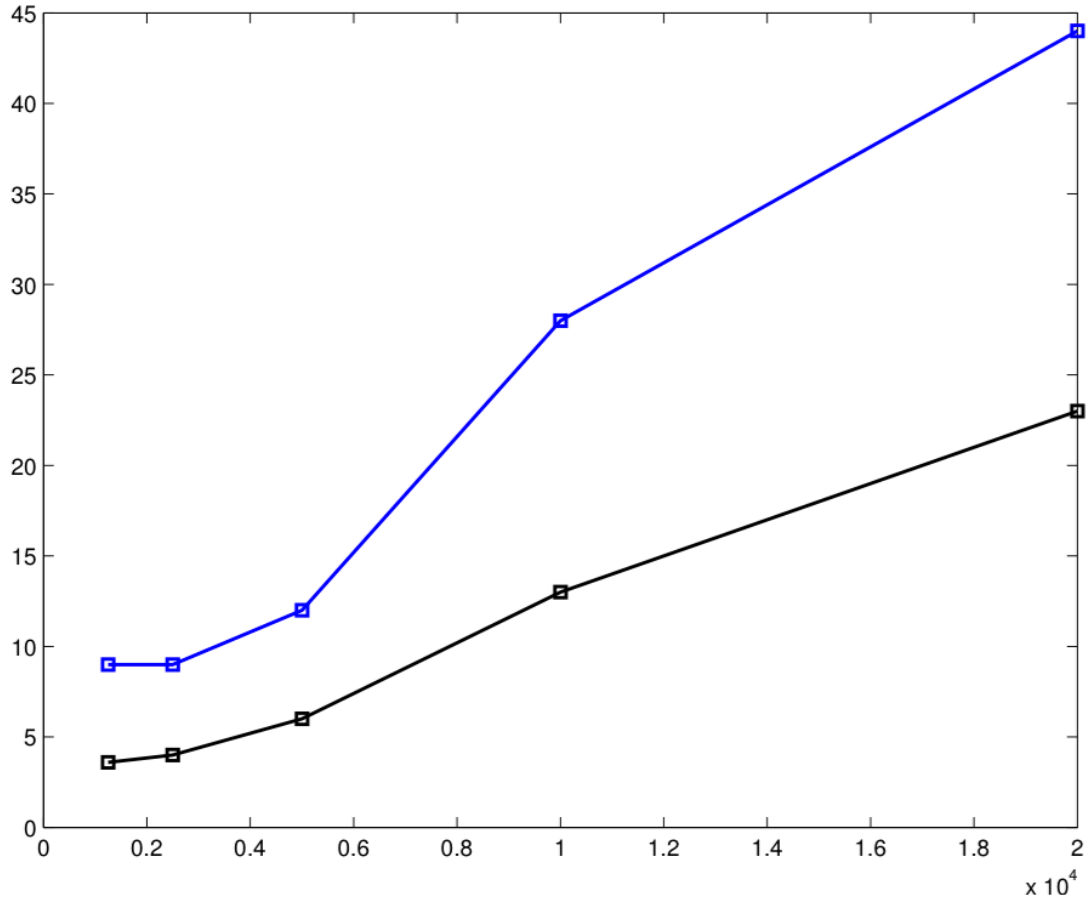
$\mathsf{V}_{row} = \mathsf{B}_{row}^T \tilde{\mathsf{U}} \Sigma^{-1}$

output V_{row}

Mahout SSVD Parameters

parameter	default	description
–rank (-k)	none	decomposition rank
–oversampling (-p)	15	oversampling
–powerIter (-q)	0	number of additional power iterations
–blockHeight (-r)	10,000	Y block height (must be $> (k+p)$)
–outerProdBlockHeight (-oh)	30,000	block height of outer products during multiplication, increase for sparse input
–abtBlockHeight (-abth)	200,000	block height of Y_i in ABtJob during AB^T multiplication, increase for extremely sparse inputs
–reduceTasks (-t)	1	number of reduce tasks (where applicable)
–minSplitSize (-s)	-1	minimum split size

Block height



blockHeight (`-r`)	1250	**2500**	5000	10000	20000
Q-job	9	**9**	12	28	44
per map	3.6	**4**	6	13	23

Power iterations

powerIters

phase	time
Q-job	4
Bt-job	**16**
ABt-job	**12**
U-job	2
V-job	2

q	time
0	26
1	52
2	80
3	107

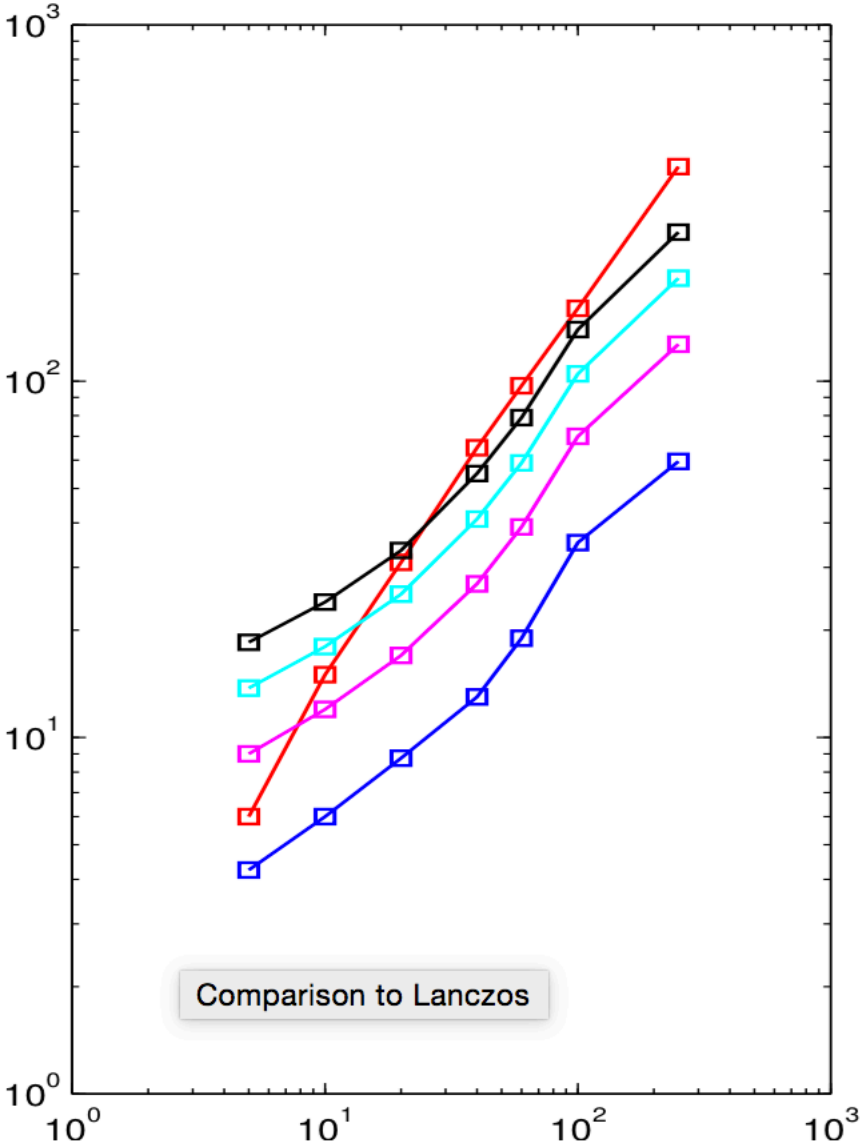
Comparison to Lanczos

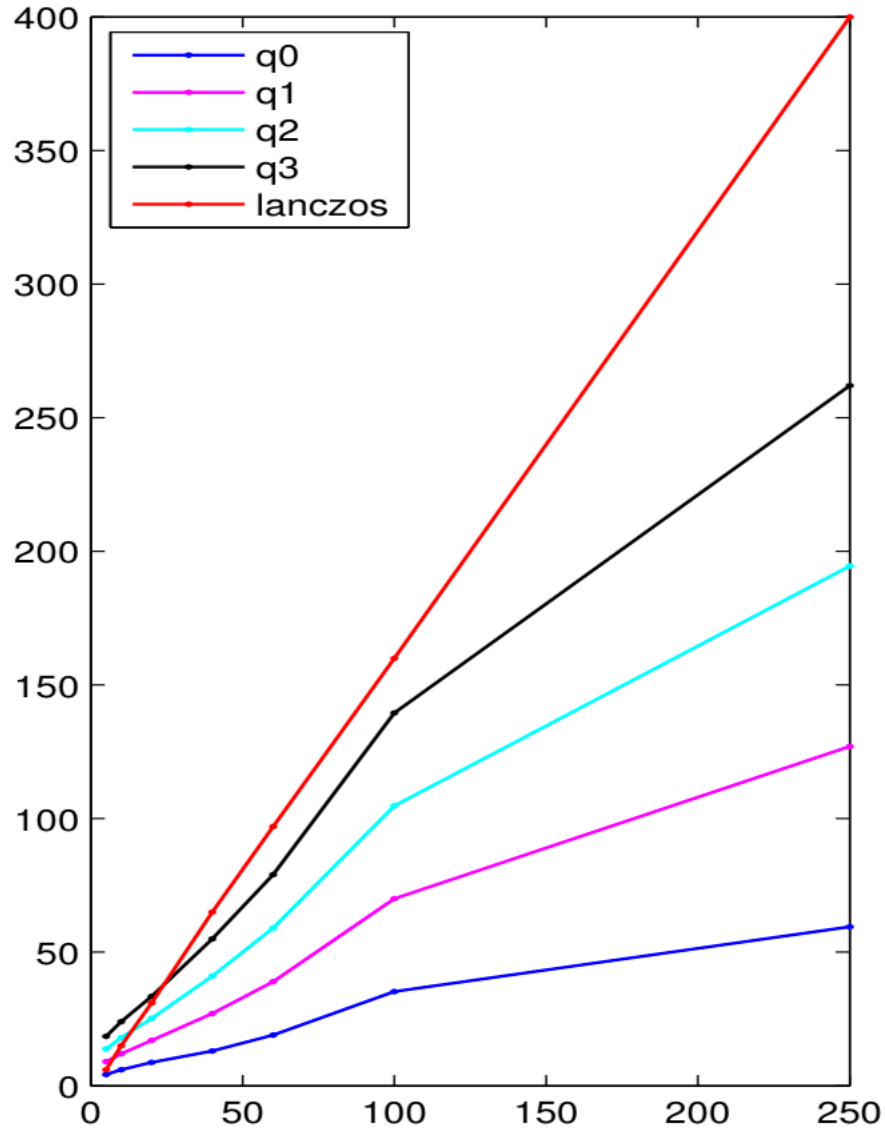
ALGORITHM 4.8.2: LANCZOS SVD

Given an $m \times n$ matrix A and a desired rank k, Lanczos SVD computes the k dimensional eigen-decomposition, $\mathsf{V}\Sigma^2\mathsf{V}^T$, of $\mathsf{A}^T\mathsf{A}$ which yields the right singular vectors V and singular values Σ of A.

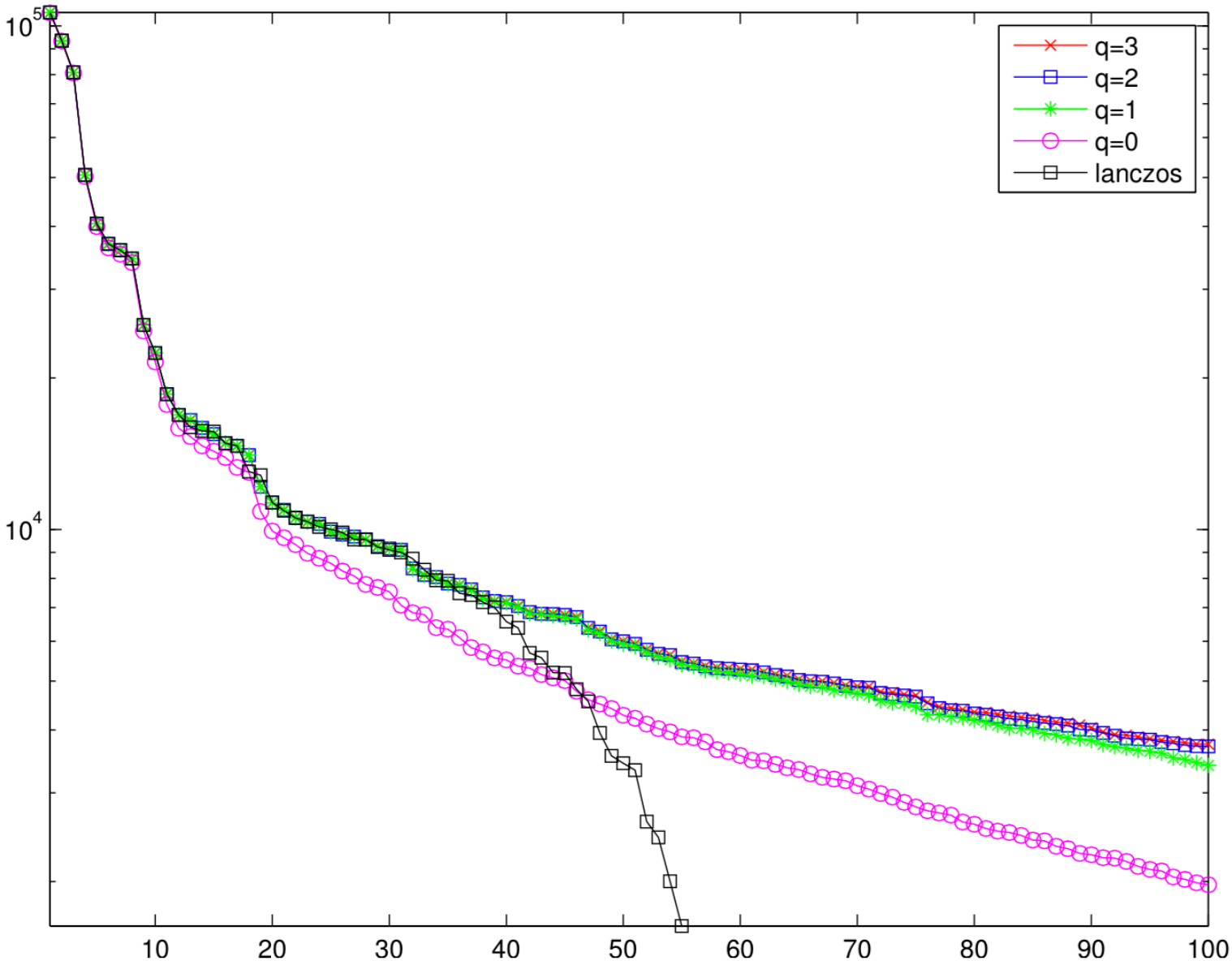
$\mathbf{q} = \mathsf{A}^T\mathsf{A}\boldsymbol{\omega}$
$\mathbf{q}_1 = \mathbf{q}/\|\mathbf{q}\|$
while $i \leq k$ **do**
  MapReduce step: Algorithm 4.5
    $\mathbf{q} = \mathsf{A}^T\mathsf{A}\mathbf{q}_i$
  Serial step:
    for $j = 1..i$
      $\mathbf{q} \leftarrow \mathbf{q} - \langle \mathbf{q}_j, \mathbf{q} \rangle \mathbf{q}_j$
      $\mathbf{q}_{i+1} = \mathbf{q}/\|\mathbf{q}\|$
      collect: α, β
    end for
end while
compute $\mathsf{X}\Lambda\mathsf{X}^T = T$
$\mathsf{V} = \mathsf{Q}\mathsf{X}$
$\sigma_i = \sqrt{\lambda_i}$

Comparison to Lanczos

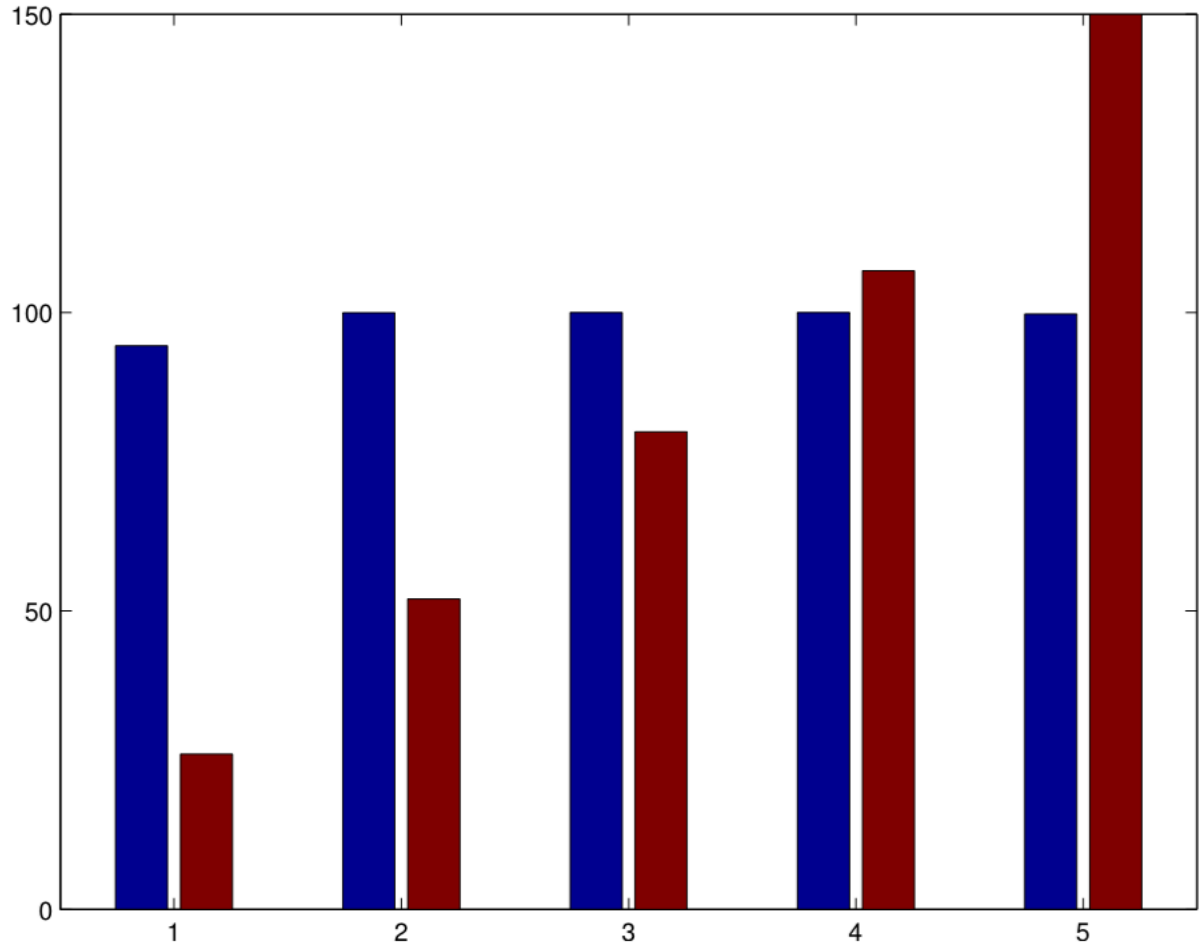




Comparison to Lanczos



Comparison to Lanczos



Datasets

- Wikipedia-all

Execution times	minutes
Q-job	4
Bt-job	49
ABt-job	127
U-job	1.5
V-job	7
$q = 0$	61
$q = 1$	235

- Wikipedia-MAX

phase	time
Q-job	20
Bt-Job	365
ABt-job	571
U-job	8
V-job	14
total	1335

That's SSVD!

Resources

- Randomized methods for computing the SVD of very large matrices
  - http://web.stanford.edu/group/mmds/slides2010/Martinsson.pdf
- Randomized methods for computing low-rank approximations of matrices
  - https://amath.colorado.edu/faculty/martinss/Pubs/2012_halko_dissertation.pdf
- SSVD on Mahout
  - https://mahout.apache.org/users/algorithms/d-ssvd.html