Single-Pass PCA of Large High-Dimensional Data

Wenjian Yu[1], Yu Gu[1], Jian Li[1], Shenghua Liu[2], Yaohang Li[3]

[1]Department of Computer Science and Technology,
Tsinghua University, Beijing 100084, China

[2]Institute of Computing Technology, *Chinese Academy of Science*

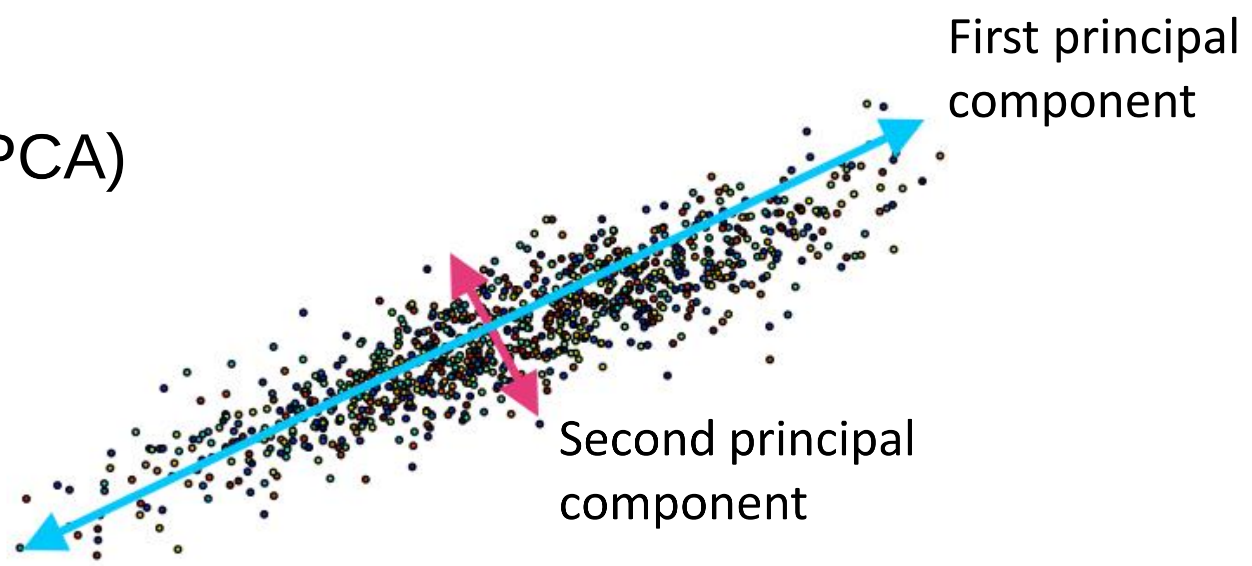
[3]Department of Computer Science, *Old Dominion University*, USA
yu-wj@tsinghua.edu.cn

Outline

- **Introduction**
- **Technical Background**
- **The Single-Pass Algorithm for PCA**
- **Experimental Results**
- **Conclusion**

Introduction

- Background
  - Principal component analysis (PCA)
  - An open problem: calculate PCA of large-size and high-dimensional dense data in a limited-memory computer



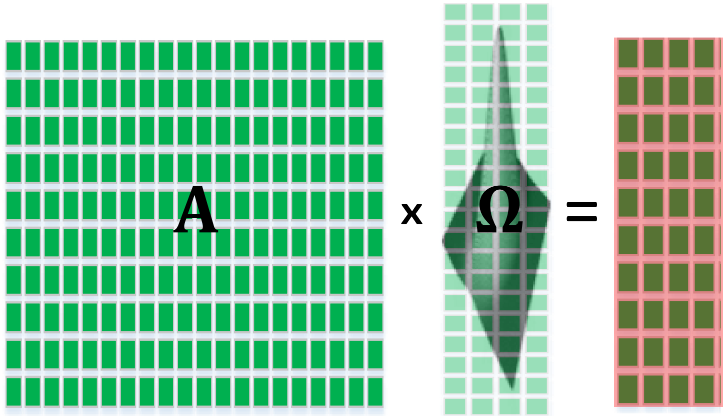
  - A single-pass algorithm: particularly useful / efficient, for data stored in slow memory or streaming data
  - There are single-pass PCA algorithms for SPSD matrix or low-dimensional data, but the study for the algorithm for more general matrices is not sufficient.

Introduction

- Randomized matrix algorithm
  - Has advantages over traditional algorithms (like SVD) (faster runtime, better parallelism, pass-efficient; suitable for large data)
  - randQB based on *random projection* [1]

$$\mathbf{A} \approx \mathbf{QB}$$

Q captures the dominant actions of **A**
Small sketch **B** facilitates computation



  - Applied to computing PCA [2]
  - Useful for distributed PCA; excellent performance on parallel computers
  - A blocked version for rank-revealing matrix factorization [3]

[1] **N Halko**, **P-G Martinsson**, **J A Tropp**, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix Decompositions," ***SIAM review***, 2011

[2] **N Halko**, **et al.**, "An algorithm for the principal component analysis of large data sets," ***SISC***, 2011

[3] **P-G Martinsson** and **S. Voronin**, "A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices," ***SISC***, 2016

Introduction

- Our contribution
  - We reconstruct the blocked randQB algorithm [3] to obtain a single-pass PCA algorithm
  - Single-pass
    - involves only one pass over specified large high-dimensional data
  - Efficiency
    - O(mnk) time complexity and O($k(m+n)$) space complexity for computing k-PCA, and well adapts to parallel computing
  - Accuracy
    - same theoretic error bounds as the randomized blocked algorithm; much less error than its counterpart

[3] **P-G Martinsson** and **S. Voronin**, "A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices," ***SISC***, 2016

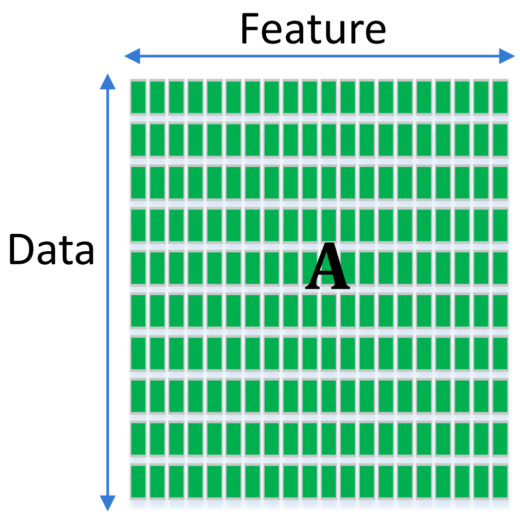
Technical Background – SVD and PCA

- Truncated singular value decomposition (SVD)

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} \Rightarrow \mathbf{A} \approx \mathbf{A}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^{\mathrm{T}}$$

  - $\mathbf{A}_k$: rank-k approximation of $\mathbf{A}$ (optimal in l_2-norm and F-norm)

- SVD and PCA are closely related
  - Suppose each row of matrix $\mathbf{A}$ is an observed data
  - PCA is realized through truncated SVD
  - The leading right singular vectors ($\mathbf{v}_i$) of $\mathbf{A}$ are the principal components. Particularly, $\mathbf{v}_1$ is the first principal component



Technical Background – Randomized SVD

- Basic randQB scheme
  - Produce near-optimal low-rank appr.
  - Accuracy can be improved with power iteration scheme
  - Well suit to parallel computing
  - Result has small random variance
  - Better than the column-pivoted QR
- A single-pass variant
  - Reduce to 1 visit of $\mathbf{A}$

$$\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^{\mathrm{T}}\mathbf{A}\tilde{\mathbf{Q}}\tilde{\mathbf{Q}}^{\mathrm{T}} = \mathbf{Q}\mathbf{B}\tilde{\mathbf{Q}}^{\mathrm{T}}$$

  - More approximation is included

Algorithm 1 Basic randomized scheme for truncated SVD

Require: $\mathbf{A} \in \mathbb{R}^{m\times n}$, rank k, over-sampling parameter s.

$l = k + s$;
$\mathbf{\Omega} = \mathrm{randn}(n, l)$;
$\mathbf{Q} = \mathrm{orth}(\mathbf{A}\mathbf{\Omega})$;
$\mathbf{B} = \mathbf{Q}^\top \mathbf{A}$;
$[\tilde{\mathbf{U}}, \mathbf{S}, \mathbf{V}] = \mathrm{svd}(\mathbf{B})$;
$\mathbf{U} = \mathbf{Q}\tilde{\mathbf{U}}$;
$\mathbf{U} = \mathbf{U}(:, 1:k)$; $\mathbf{V} = \mathbf{V}(:, 1:k)$; $\mathbf{S} = \mathbf{S}(1:k, 1:k)$;
return $\mathbf{U}, \mathbf{S}, \mathbf{V}$.

A is visited twice

Algorithm 2 An existing single-pass algorithm

Require: $\mathbf{A} \in \mathbb{R}^{m\times n}$, rank parameter k.

Generate random $n \times k$ matrix $\mathbf{\Omega}$ and $m \times k$ matrix $\tilde{\mathbf{\Omega}}$;
Compute $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$ and $\tilde{\mathbf{Y}} = \mathbf{A}^\top \tilde{\mathbf{\Omega}}$ in a single pass over $\mathbf{A}$;
$\mathbf{Q} = \mathrm{orth}(\mathbf{Y})$; $\tilde{\mathbf{Q}} = \mathrm{orth}(\tilde{\mathbf{Y}})$;
Solve linear equation $\tilde{\mathbf{\Omega}}^\top \mathbf{Q}\mathbf{B} = \tilde{\mathbf{Y}}^\top \tilde{\mathbf{Q}}$ for $\mathbf{B}$;
$[\tilde{\mathbf{U}}, \mathbf{S}, \tilde{\mathbf{V}}] = \mathrm{svd}(\mathbf{B})$;
$\mathbf{U} = \mathbf{Q}\tilde{\mathbf{U}}$; $\mathbf{V} = \tilde{\mathbf{Q}}\tilde{\mathbf{V}}$;
return $\mathbf{U}, \mathbf{S}, \mathbf{V}$.

The Single-Pass PCA Algorithm

The blocked randQB algorithm [3]

function $[\mathbf{Q}, \mathbf{B}] = \texttt{randQB\_b}\ (\mathbf{A}, \varepsilon, b)$

for $i = 1, 2, 3, \cdots$
  $\mathbf{\Omega}_i = \text{randn}(n, b)$;
  $\mathbf{Q}_i = \text{orth}(\mathbf{A}\mathbf{\Omega}_i)$;
  $\mathbf{Q}_i = \text{orth}(\mathbf{Q}_i - \sum_{j=1}^{i-1} \mathbf{Q}_j \mathbf{Q}_j^\top \mathbf{Q}_i)$;
  $\mathbf{B}_i = \mathbf{Q}_i^\top \mathbf{A}$;
  $\mathbf{A} = \mathbf{A} - \mathbf{Q}_i \mathbf{B}_i$;
  if $\|\mathbf{A}\| < \varepsilon$ **then stop**
end for
$\mathbf{Q} = [\mathbf{Q}_1 \ \cdots \ \mathbf{Q}_i]$; $\mathbf{B} = [\mathbf{B}_1^\top \ \cdots \ \mathbf{B}_i^\top]^\top$.

- Mathematically equivalent to the basic randQB algorithm (Gram-Schmidt procedure)
- Iterative blocked procedure for monitoring approximation error while keeping high efficiency
- Mainly aimed at the problem of adaptive rank determination

Convert it to a pass-efficient procedure (multiplications with A moved out of loop)

Theorem 1: The $\mathbf{Q}$ and $\mathbf{B}$ obtained with Alg. 3 satisfy: $\mathbf{Q}$ is orthonormal and $\mathbf{B} = \mathbf{Q}^{\mathrm{T}}\mathbf{A}$

Algorithm 3 A pass-efficient blocked algorithm

Require: $\mathbf{A} \in \mathbb{R}^{m \times n}$, rank parameter k, block size b.
$\mathbf{Q} = [\,]$; $\mathbf{B} = [\,]$;
$\mathbf{\Omega} = \text{randn}(n, k)$;
$\mathbf{G} = \mathbf{A}\mathbf{\Omega}$;
$\mathbf{H} = \mathbf{A}^\top \mathbf{G}$;
for $i = 1, 2, \cdots, k/b$ **do**
  $\mathbf{\Omega}_i = \mathbf{\Omega}(:, \ (i-1)b+1 : ib)$;
  $\mathbf{Y}_i = \mathbf{G}(:, \ (i-1)b+1 : ib) - \mathbf{Q}(\mathbf{B}\mathbf{\Omega}_i)$;
  $[\mathbf{Q}_i, \ \mathbf{R}_i] = \text{qr}(\mathbf{Y}_i)$;
  $\mathbf{B}_i = \mathbf{R}_i^{-\top}(\mathbf{H}(:, \ (i-1)b+1 : ib)^\top - \mathbf{\Omega}_i^\top \mathbf{B}^\top \mathbf{B})$;
  $\mathbf{Q} = [\mathbf{Q}, \ \mathbf{Q}_i]$; $\mathbf{B} = [\mathbf{B}^\top, \ \mathbf{B}_i^\top]^\top$;
end for

The Single-Pass PCA Algorithm

- Algorithm 3
  - Equivalent to the randQB alg.
  - Steps 3 and 4 can be executed with only one pass over $\mathbf{A}$
- Add re-orthogonalization steps to alleviate round-off error
- Memory cost is about $(m+2n)l$ floating numbers
- Time complexity (flop count) close to the basic randQB (Alg.1)
- With the power scheme, accuracy can be largely improved at the cost of one more visit of $\mathbf{A}$

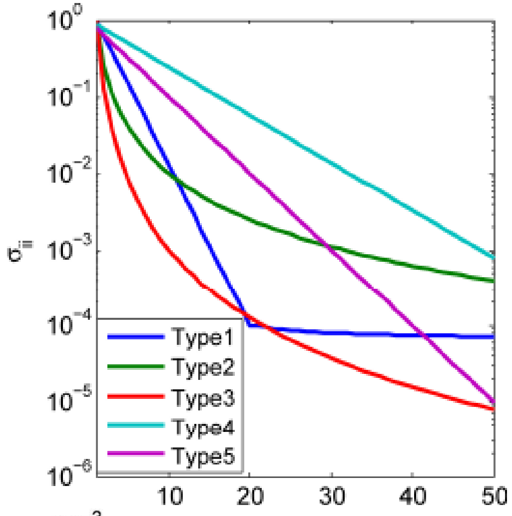
Algorithm 4 A single-pass algorithm for computing PCA

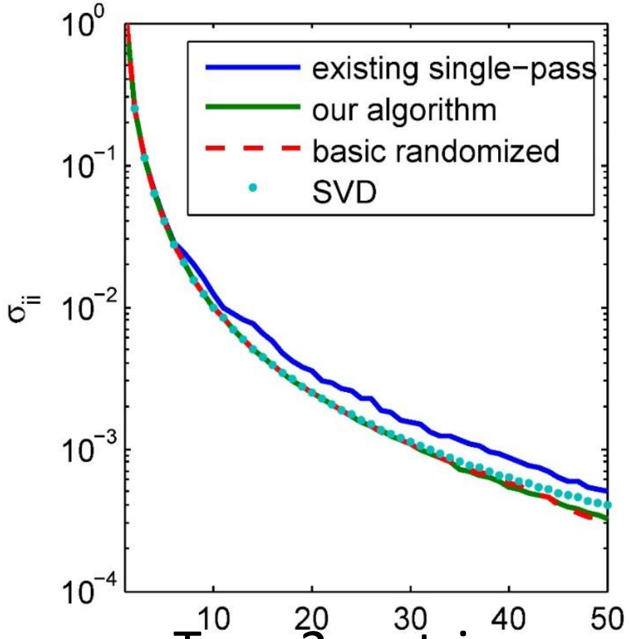
Require: $\mathbf{A} \in \mathbb{R}^{m\times n}$, rank parameter k, block size b.

$\mathbf{Q} = [\,]$; $\mathbf{B} = [\,]$;
Choose $l = tb$, which is slightly larger than k;
$\mathbf{\Omega} = \text{randn}(n, l)$; $\mathbf{G} = [\,]$; Set $\mathbf{H}$ to an $n \times l$ zero matrix;
while $\mathbf{A}$ is not completely read through **do**
  Read next few rows of $\mathbf{A}$ into RAM, denoted by $\mathbf{a}$;
  $\mathbf{g} = \mathbf{a}\mathbf{\Omega}$; $\mathbf{G} = [\mathbf{G};\ \mathbf{g}]$;
  $\mathbf{H} = \mathbf{H} + \mathbf{a}^\top \mathbf{g}$;
end while
for $i = 1, 2, \cdots, t$ **do**
  $\mathbf{\Omega}_i = \mathbf{\Omega}(:, (i-1)b+1 : ib)$;
  $\mathbf{Y}_i = \mathbf{G}(:, (i-1)b+1 : ib) - \mathbf{Q}(\mathbf{B}\mathbf{\Omega}_i)$;
  $[\mathbf{Q}_i,\ \mathbf{R}_i] = \text{qr}(\mathbf{Y}_i)$;
  $[\mathbf{Q}_i,\ \tilde{\mathbf{R}}_i] = \text{qr}(\mathbf{Q}_i - \mathbf{Q}(\mathbf{Q}^\top \mathbf{Q}_i))$;
  $\mathbf{R}_i = \tilde{\mathbf{R}}_i \mathbf{R}_i$;
  $\mathbf{B}_i = \mathbf{R}_i^{-\top}(\mathbf{H}(:, (i-1)b+1 : ib)^\top - \mathbf{Y}_i^\top \mathbf{Q}\mathbf{B} - \mathbf{\Omega}_i^\top \mathbf{B}^\top \mathbf{B})$;
  $\mathbf{Q} = [\mathbf{Q},\ \mathbf{Q}_i]$; $\mathbf{B} = [\mathbf{B}^\top,\ \mathbf{B}_i^\top]^\top$;
end for
$[\tilde{\mathbf{U}}, \mathbf{S}, \mathbf{V}] = \text{svd}(\mathbf{B})$;
$\mathbf{U} = \mathbf{Q}\tilde{\mathbf{U}}$;
$\mathbf{U} = \mathbf{U}(:, 1:k)$; $\mathbf{V} = \mathbf{V}(:, 1:k)$; $\mathbf{S} = \mathbf{S}(1:k, 1:k)$;
return $\mathbf{U}, \mathbf{S}, \mathbf{V}$.

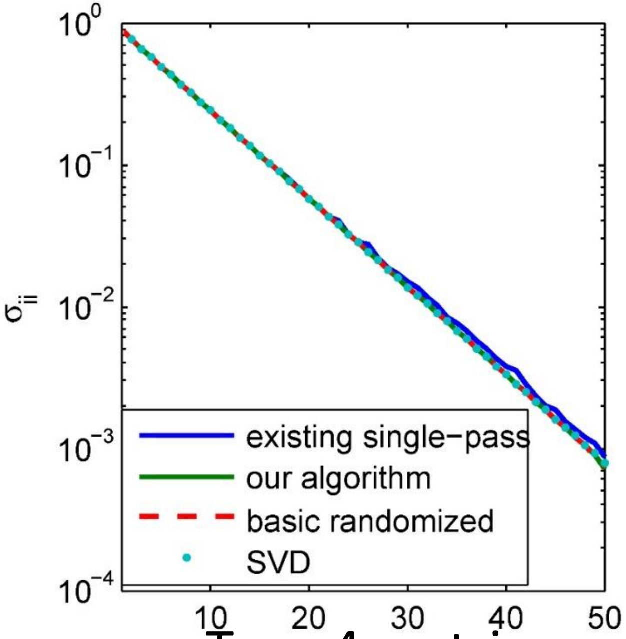
Experimental Results

- Five types of test matrices
  - Specified singular value spectrum with various decaying trend
  - For type 1 and 2 matrices, the singular value decays asymptotically slow
- Accuracy of computed singular value
  - A 3000x3000 matrix for each type

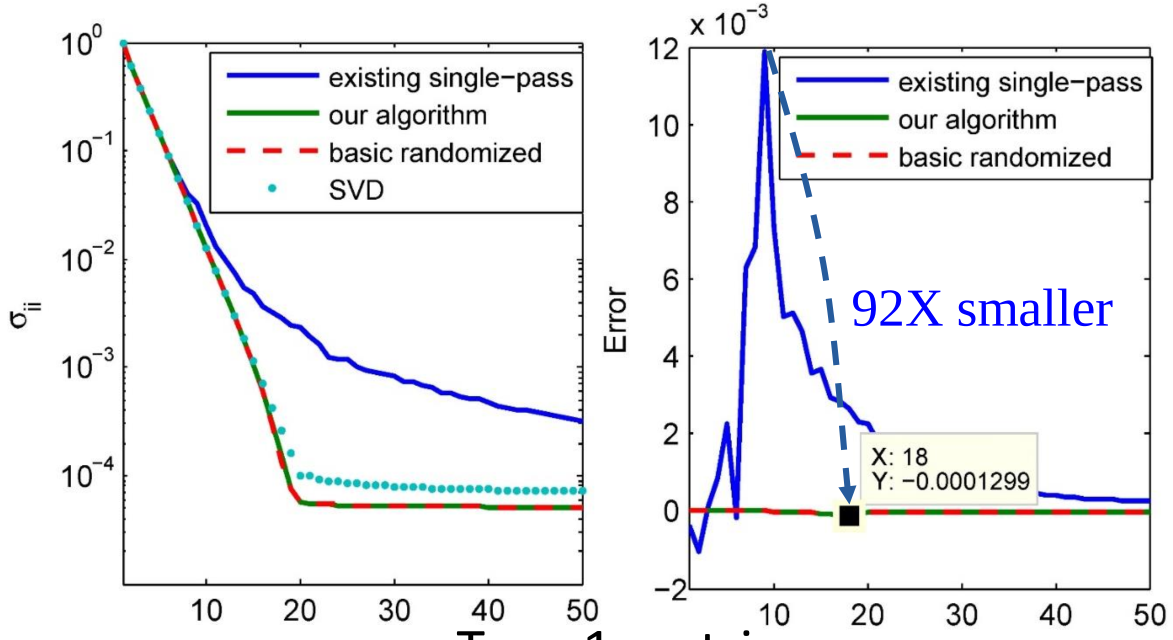




Type 2 matrix



Type 4 matrix



Type 1 matrix

Experimental Results

- Accuracy of the principal components
  - Test on type 1 matrix (with *slowly-decayed* σ_{ii}'s); smaller error for other matrices
  - For $\mathbf{v}_1$, only 2.8x10^{-5} difference in l_∞-norm
  - For the first 10 principal components, the correlation coefficients are calculated: 0.9993 ~1 even for the 10th component
- Runtime comparison (200,000x200,000 matrices)
  - Each matrix stored as a 149 GB hard-disk file
  - Alg. 4 is 2X faster than Alg. 1; more accurate than Alg. 2

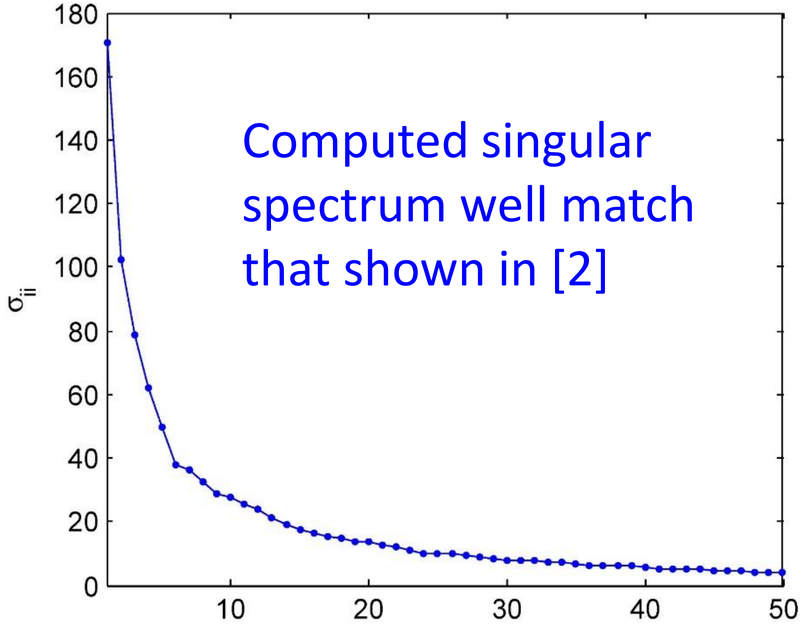
Matrix	k	Algorithm 1			Algorithm 2			Algorithm 4		
		t_{read}	t_{PCA}	max_err	t_{read}	t_{PCA}	max_err	t_{read}	t_{PCA}	max_err
Type1	16	2390	2607	1.7e-3	1186	1404	2.2e-2	1206	1426	1.8e-3
Type1	20	2420	2616	9e-4	1198	1380	1.6e-1	1217	1413	1.2e-3
Type1	24	2401	2593	1e-3	1216	1400	1.5e-1	1216	1414	1.2e-3
Type2	12	2553	2764	5e-4	1267	1477	3e-2	1276	1490	5e-4
Type3	24	2587	2777	1e-5	1312	1500	1.7e-3	1310	1502	2e-5

Experiment on a computer with two 12-core Xeon CPUs and 32GB memory

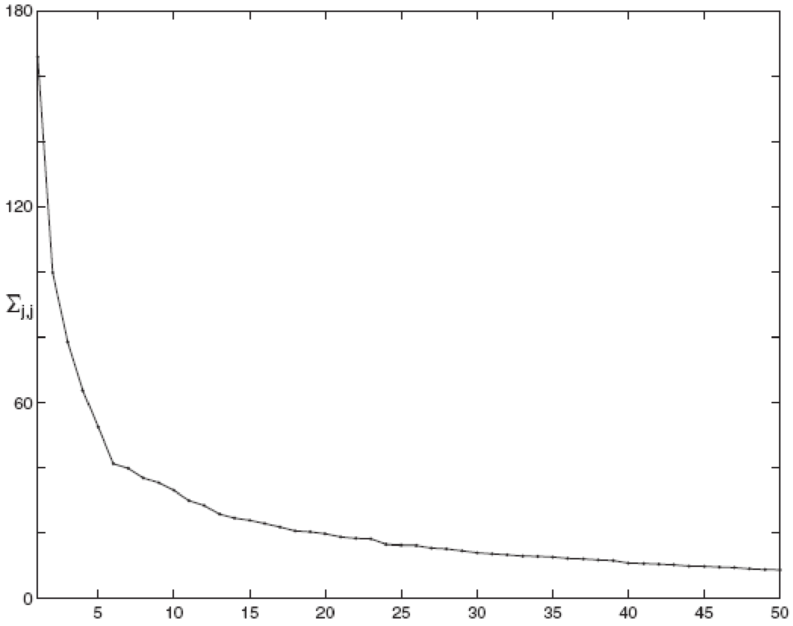
Test on a 10^4x10^4 matrix: Our Alg. 4 is over 300X faster than svd/svds

Experimental Results

- A test of real data
  - Face images from the FERET [4]
  - As in [2], constuct a 102,042x392,216 matrix (*150GB file on hard disk*)
  - Compute 50 eigenfaces on the machine with 24 CPU cores
  - Runtime of our algorithm: ~ 24 minutes



Computed singular spectrum well match that shown in [2]



Four eigenfaces

[2] **N Halko**, **et al.**, "An algorithm for the principal component analysis of large data sets," ***SISC***, 2011

[4] **P J Phillips**, **et al.**, "The FERET evaluation methodology for face-recognition algorithms," ***T-PAMI***, 2000

Conclusion

- **A single-pass PCA algorithm for large and high-dimensional data**
- **Only one pass over data matrix, providing that the matrix is stored in a row-major format**
- **Comparable accuracy to existing randomized algorithm; much less error than an existing single-pass algorithm**
- **Experiments demonstrate the algorithm's effectiveness for large-size high-dimensional data (~150 GB disk file), in terms of runtime and memory usage**

The codes of the proposed algorithm and experimental data are shared on: https://github.com/WenjianYu/rSVD-single-pass

Thank You !