# Single Pass PCA of Matrix Products

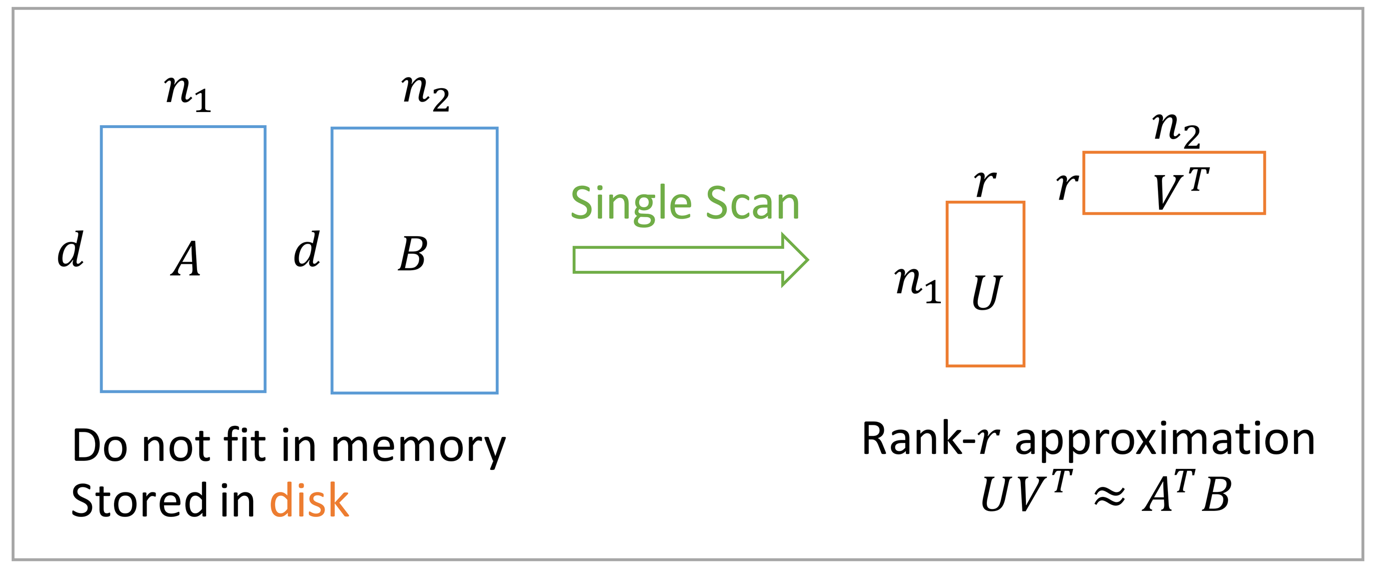
Shanshan Wu, Srinadh Bhojanapalli, Sujay Sanghavi, Alexandros G. Dimakis

The University of Texas at Austin

Toyota Technological Institute Chicago

## [ Problem ]

Given two large matrices $A$ and $B$ (assumed too large to fit in memory), find the rank-$r$ approximation of their product $A^TB$, using a single pass over the matrices.




Applications

- PCA is a special case when $A = B$
- Capture co-occurrence (e.g., $A$ user-by-query, $B$ user-by-ad) or cross-covariance relation (e.g., $A$ genotype, $B$ phenotype)

Why care about single-pass?

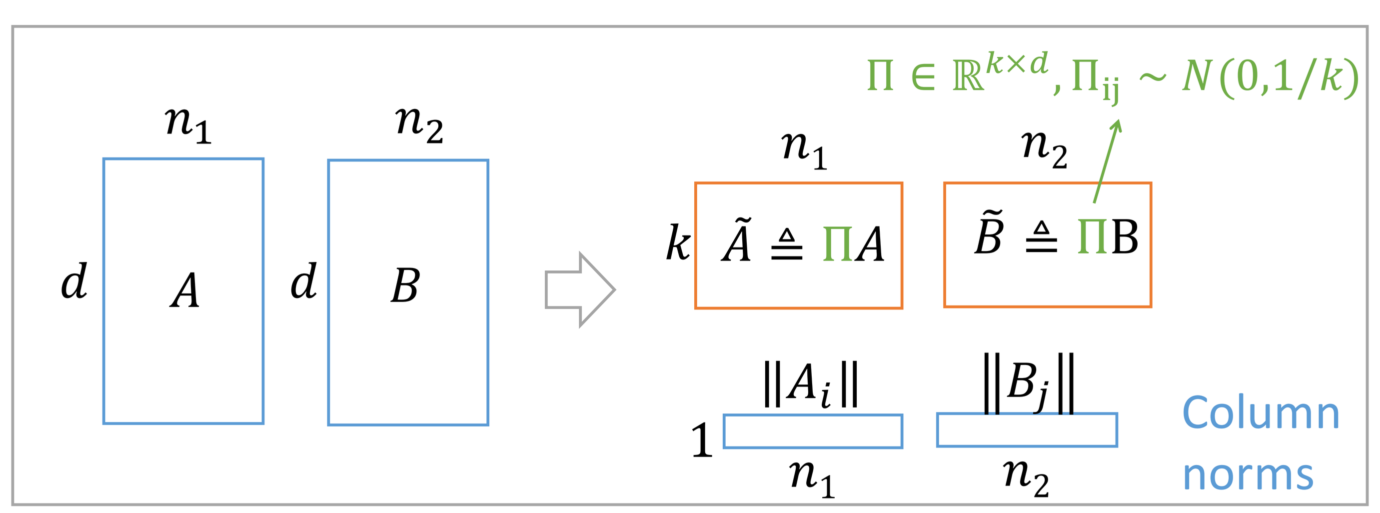
- Reduce disk I/O overhead
- Applicable when the data is streaming

| Algorithm | No. of Passes | Memory |
|---|---|---|
| $A^TB$ + SVD | 1 or more | $O(n_1n_2)$ |
| Direct power method on $A, B$ | $O(r)$ | $O(\max\{n_1,n_2\}r)$ |
| LELA [BJS15] | 2 | $O(\max\{n_1,n_2\}r^3/\epsilon^2)$ |
| SMP-PCA (Our Algorithm) | 1 | $O(\max\{n_1,n_2\}r^3/\epsilon^2)$ |

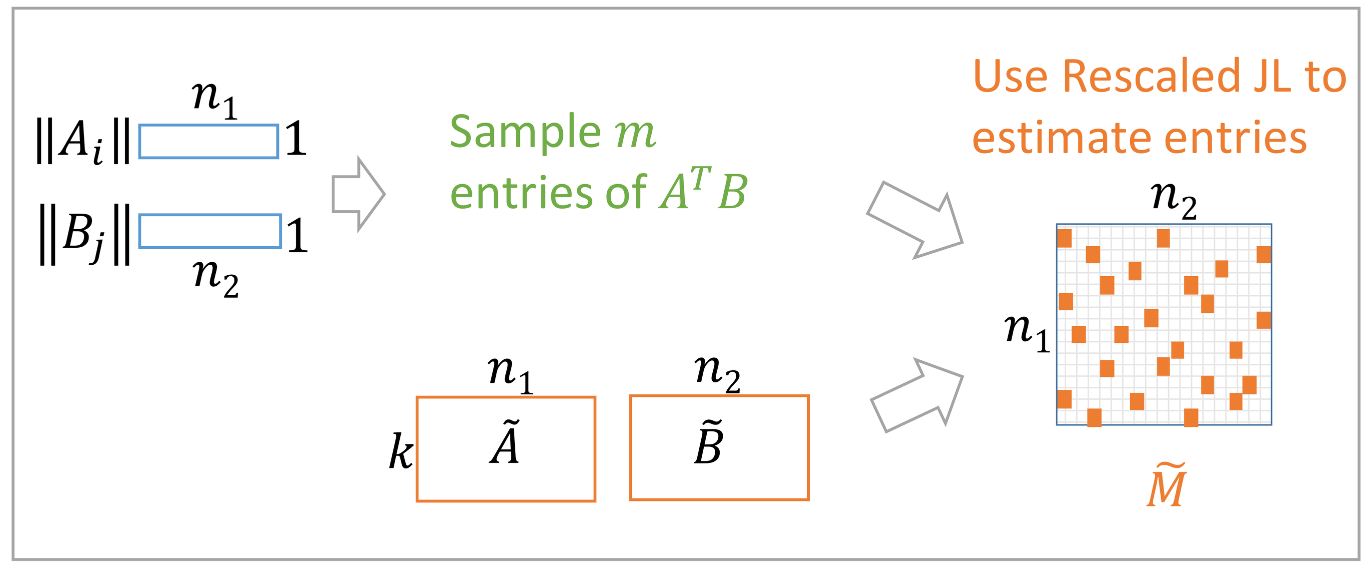
[BJS15] S. Bhojanapalli, P. Jain, and S. Sanghavi. Tighter low-rank approximation via sampling the leveraged element, SODA 2015.

## [ Our Algorithm SMP-PCA (3 steps)]

**Step 1** Sketching




**Step 2** Sampling & estimating entries of $A^TB$




**Key Idea I** Entrywise Sampling

$$\text{Sample } (A^TB)_{ij} \quad \propto \quad q_{ij} = m\left(\frac{\|A_i\|^2}{2n_2\|A\|_F^2} + \frac{\|B_j\|^2}{2n_1\|B\|_F^2}\right)$$

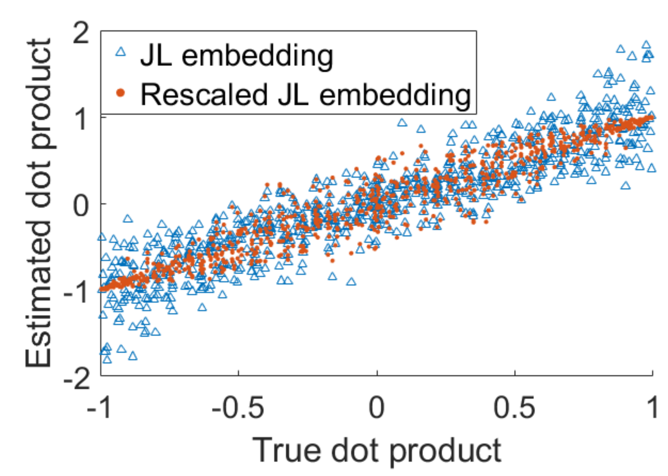
Higher weights are given to heavy rows and columns

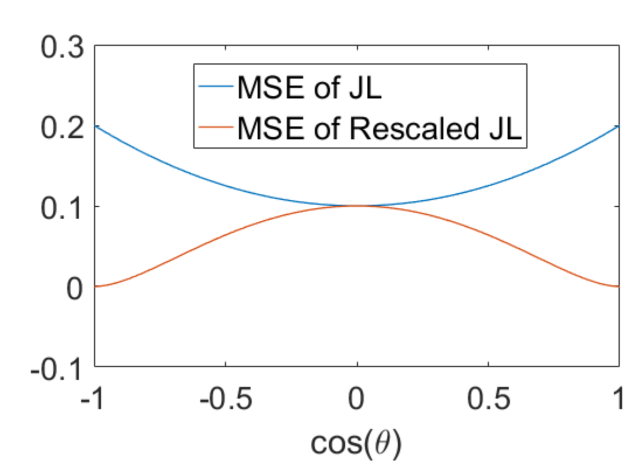
**Key Idea II** Rescaled JL

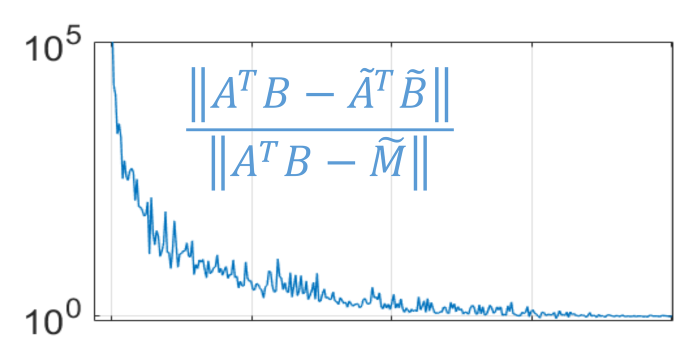
Rescaling reduces error from distorted norms

$$\tilde{M}_{ij} = \|A_i\| \cdot \|B_j\| \cdot \frac{\tilde{A}_i^T\tilde{B}_j}{\|\tilde{A}_i\| \cdot \|\tilde{B}_j\|}$$

**Observation** Rescaled JL ($\tilde{M}_{ij}$) has smaller mean squared error than standard JL ($\tilde{A}_i^T\tilde{B}_j$).




$\tilde{M}$ is more accurate than $\tilde{A}^T\tilde{B}$, when the column vectors are more correlated.

**Step 3** Low rank approx. from the sampled entries


WAltMin: optimize over $U(V)$ assuming $V(U)$ is fixed.

$$\min_{U,V} \sum_{(i,j)\in\Omega} \frac{\left(e_i^TUV^Te_j - \tilde{M}_{ij}\right)^2}{\hat{q}_{ij}}$$

Weights ensure unbiasedness

## [ Theoretical Guarantee]

**Theorem** (Informal) Let

- $\rho$ be the condition number of $(A^TB)_r$
- $\tilde{r}$ be the maximum stable rank of $A$ and $B$

If the input parameters satisfy

- Sketch size $k = O\left(\frac{r^3\rho^2\tilde{r}\log n}{\epsilon^2}\right)$
- The number of samples $m = O\left(\frac{nr^3\rho^2\tilde{r}^2T^2\log n}{\gamma\,\epsilon^2}\right)$
- The number of WAltMin iterations $T = O\left(\log\frac{1}{\zeta}\right)$

Then w.p. $\geq 1-\gamma$, we get

$$\|(A^TB)_r - UV^T\| \leq \epsilon\|A^TB - (A^TB)_r\|_F + \zeta + \epsilon\sigma_r^*$$

Vanishes if $A^TB$ is exactly rank-$r$

Captures the error caused by sketching

Computation Complexity:

$$O(\text{nnz}(A)k + \text{nnz}(B)k + mk + mr^2T)$$

Sketching — Sampling — Alternating least squares

## [ Numerical Experiments]

**Spark Implementation**

Source code *https://github.com/wushanshan/MatrixProductPCA*

Flow chat of SMC-PCA in Spark-1.6.2:

$A, B$: RDD [(Int, (BDM[Double], BDM[Double]))], StorageLevel.DISK_ONLY

⇩ Sketching

$\tilde{A}, \tilde{B}$: RDD [(Int, (BDM[Double], BDM[Double]))]
$\|A_i\|, \|B_j\|$: Array[Double]

⇨ Sampling

Sampled Matrix: RDD[Rating]

⇩ Run ALS.train

Low-rank factors: RDD[(Int, Array[Double])]

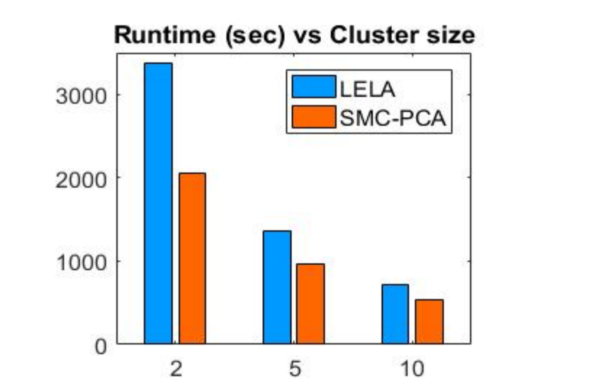
**Synthetic Dataset**

- Spark-1.6.2, m3.2xlarge EC2 instances
- $A, B$: 100k-by-100k dense matrices (150GB)
- Use SRHT as the sketching step

| Algorithm | Error | Runtime [1] |
|---|---|---|
| Exact SVD[2] | 0.0271 | 23 hrs |
| LELA | 0.0274 | 56 mins |
| SMP-PCA | 0.0280 | 34 mins |

[1] Runtime on a cluster with two m3.2xlarge instances
[2] Perform power method directly on $A$, $B$

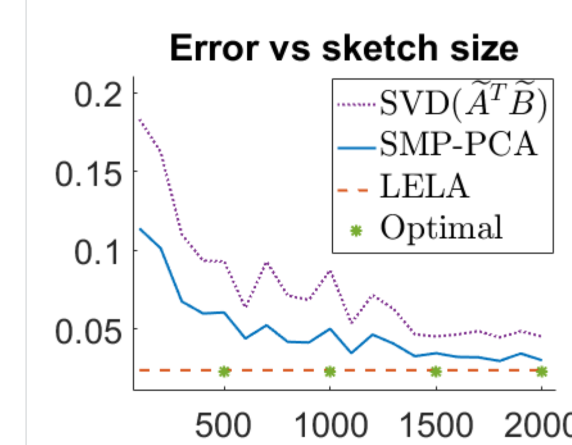
Runs faster with smaller loss of accuracy




Scales as the cluster size grows

**Real Dataset**

- SMP-PCA outperforms SVD($\tilde{A}^T\tilde{B}$) because of Rescaled JL
- SMP-PCA achieves similar error as the two-pass LELA




| Dataset | Optimal | LELA | SMP-PCA |
|---|---|---|---|
| URL-malicious Size: 792k-by-10k | 0.0163 | 0.0182 | 0.0188 |
| URL-benign Size: 1603k-by-10k | 0.0103 | 0.0105 | 0.0117 |