Improved Distributed Principal Component Analysis

Maria-Florina Balcan†, Vandana Kanchanapally‡, Yingyu Liang◇, and David Woodruff*

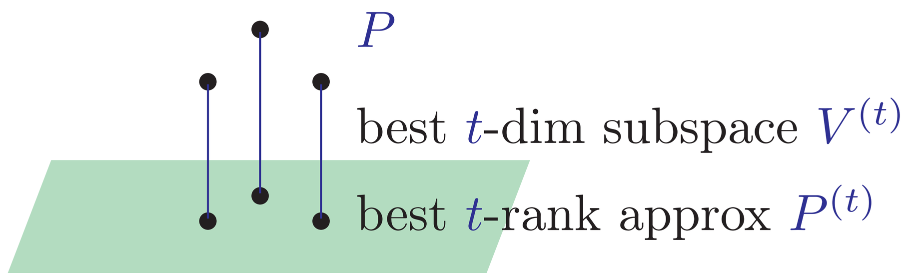
†ninamf@cs.cmu.edu, ‡vvandana@gatech.edu, ◇yingyul@cs.princeton.edu, *dpwoodru@us.ibm.com

Motivation

▷ **Principal Component Analysis**

Given: data matrix P with zero mean

Traditional Goal: to summarize/visualize the data in low dim,
find t-dim subspace V to min $\|P - PVV^\top\|_F^2$



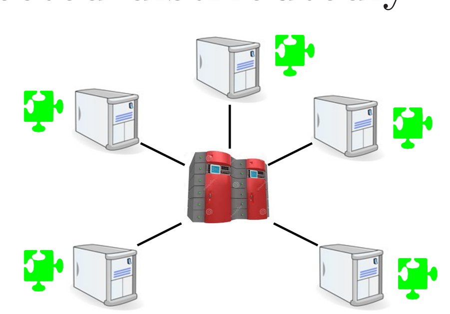
More Important: PCA pre-processes for downstream applications
e.g. k-means, regression, kernel methods, ...

Our Goal: find t-dim subspace V s.t. for downstream applications,
any α-approx for $PVV^\top$ is $(1+\epsilon)\alpha$-approx for P

▷ **Distributed Data Model**

PCA is most useful for large-scale data collected distributedly

- s servers linked to a coordinator
- Local datasets $P_1, P_2, \ldots, P_s$: points in dim d
- Global dataset $P = \bigcup_i P_i$



▷ **Challenges in Distributed PCA**

1. Arbitrary local datasets: no distributional or low rank assumption
2. Relative error guarantees for downstream applications
3. Low communication/computation cost

Our Results

▷ **disPCA** algorithm

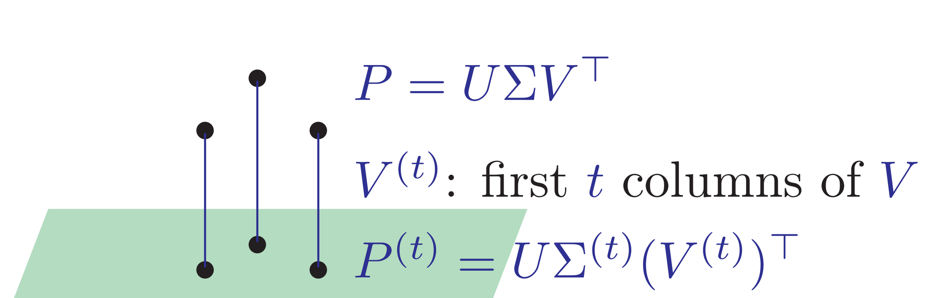
- Relative $(1+\epsilon)$ error for downstream applications
- Holds for arbitrary data and a wide family of problems: includes low rank approx, k-means, LDA, NNMF, ...
- Low communication cost: $O(skd/\epsilon^2)$ words

▷ **Fast disPCA** algorithm

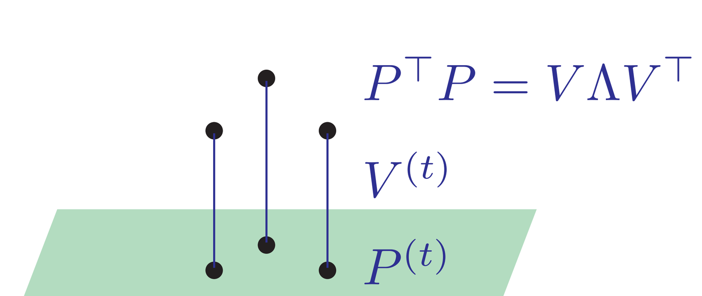
- Easy plug-in of randomized techniques
- Merely comprise the quality and communication
- Speedup of orders of magnitude

Review: Classic Non-Distributed PCA Algorithm

1. SVD on Data



2. Factorize Covariance $S = P^\top P$



For a matrix A, let $A^{(t)}$ denote the first t columns of A

Distributed PCA Algorithm

▷ Algorithm **disPCA**

1. Each server: SVD $P_i = U_i \Sigma_i V_i^\top$;
   send $\Sigma_i^{(t)}$ and $V_i^{(t)}$ to coordinator
2. Coordinator: estimate $S = \sum_i V_i^{(t)} \Sigma_i^{(t)} \Sigma_i^{(t)} \left(V_i^{(t)}\right)^\top$;
   factorize $S = V\Lambda V^\top$ and get $V^{(t)}$

▷ Equivalent Form of Algorithm **disPCA**

Let $Y_i = \Sigma_i^{(t)} \left(V_i^{(t)}\right)^\top$, then $S = \sum_i Y_i^\top Y_i$.

Factorizing S is equivalent to SVD on $Y = [Y_1; Y_2; \ldots; Y_s]$.

$$P = \begin{bmatrix} P_1 \\ \vdots \\ P_s \end{bmatrix} \begin{matrix} \xrightarrow{\text{Local PCA}} \\ \vdots \\ \xrightarrow{\text{Local PCA}} \end{matrix} \begin{bmatrix} \Sigma_1^{(t)} \left(V_1^{(t)}\right)^\top \\ \vdots \\ \Sigma_s^{(t)} \left(V_s^{(t)}\right)^\top \end{bmatrix} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_s \end{bmatrix} = Y \xrightarrow{\text{Global PCA}} V^{(t)}$$

Guarantee of disPCA

▷ **Main Theorem** (for k-means) : Suppose $t = O(k/\epsilon^2)$.
Then any α-approx for $\widetilde{P} = PV^{(t)}(V^{(t)})^\top$ is $(1+\epsilon)\alpha$-approx for P.

- When $t = O(rk/\epsilon^2)$, the same holds for r-Subspace k-Clustering
  $\min_{\mathcal{L}} \sum_{p \in P} \min_{L \in \mathcal{L}} d^2(p, L)$
  where $\mathcal{L} = \{L_j\}_{j=1}^k$ is a set of k linear/affine subspaces of dim r.

Analysis

▷ **I.** Close Projection Property of SVD:

Let $A = U\Sigma W^\top$ be its SVD, and $\widehat{A} = AW^{(t)}\left(W^{(t)}\right)^\top$.

For any k-dim subspace X, $AXX^\top$ and $\widehat{A}XX^\top$ are close.

▷ **II.** Close Projection Property of **disPCA**:

For any k-dim subspace X, $PXX^\top$ and $\widetilde{P}XX^\top$ are close.

$$P \xrightarrow{\text{project to } V^{(t)}} \widetilde{P}$$

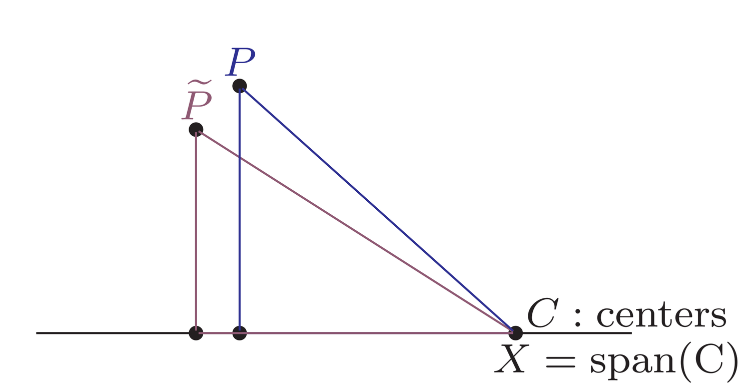
$$P_i \xrightarrow{\text{project to } V_i^{(t)}} \widehat{P}_i \xrightarrow{\text{project to } V^{(t)}} \bar{P}_i$$

$$\|(P-\widetilde{P})XX^\top\|_F^2 \le \sum_i \|(P_i - \widehat{P}_i)XX^\top\|_F^2 + 2\|(\widehat{P} - \bar{P})XX^\top\|_F^2 + 2\|(\bar{P} - \widetilde{P})XX^\top\|_F^2$$

each term can be bounded by **I** or its generalization

▷ **III.** Cost Decomposition:

Decompose the costs of $\widetilde{P}$ and P and show that they are close



$$\left.\begin{aligned} \mathrm{cost}(P) &= d^2(P, PXX^\top) + d^2(PXX^\top, C) \\ \mathrm{cost}(\widetilde{P}) &= d^2(\widetilde{P}, \widetilde{P}XX^\top) + d^2(\widetilde{P}XX^\top, C) \end{aligned}\right\}$$ difference in each part bounded by **II** and weak triangle inequality

Fast Distributed PCA

▷ **Techniques:** subspace embedding; randomized SVD

▷ **Subspace Embedding**

Property of embedding matrix H:

- With prob $\ge 99/100$, for any y
  $$\|HPy\| = (1 \pm \epsilon)\|Py\|$$
  such that **disPCA**(HP) almost equivalent to **disPCA**(P)
- HP much smaller than P, computed in time $O(\mathrm{nnz}(P))$; particularly suitable for sparse data
- HP can be computed locally

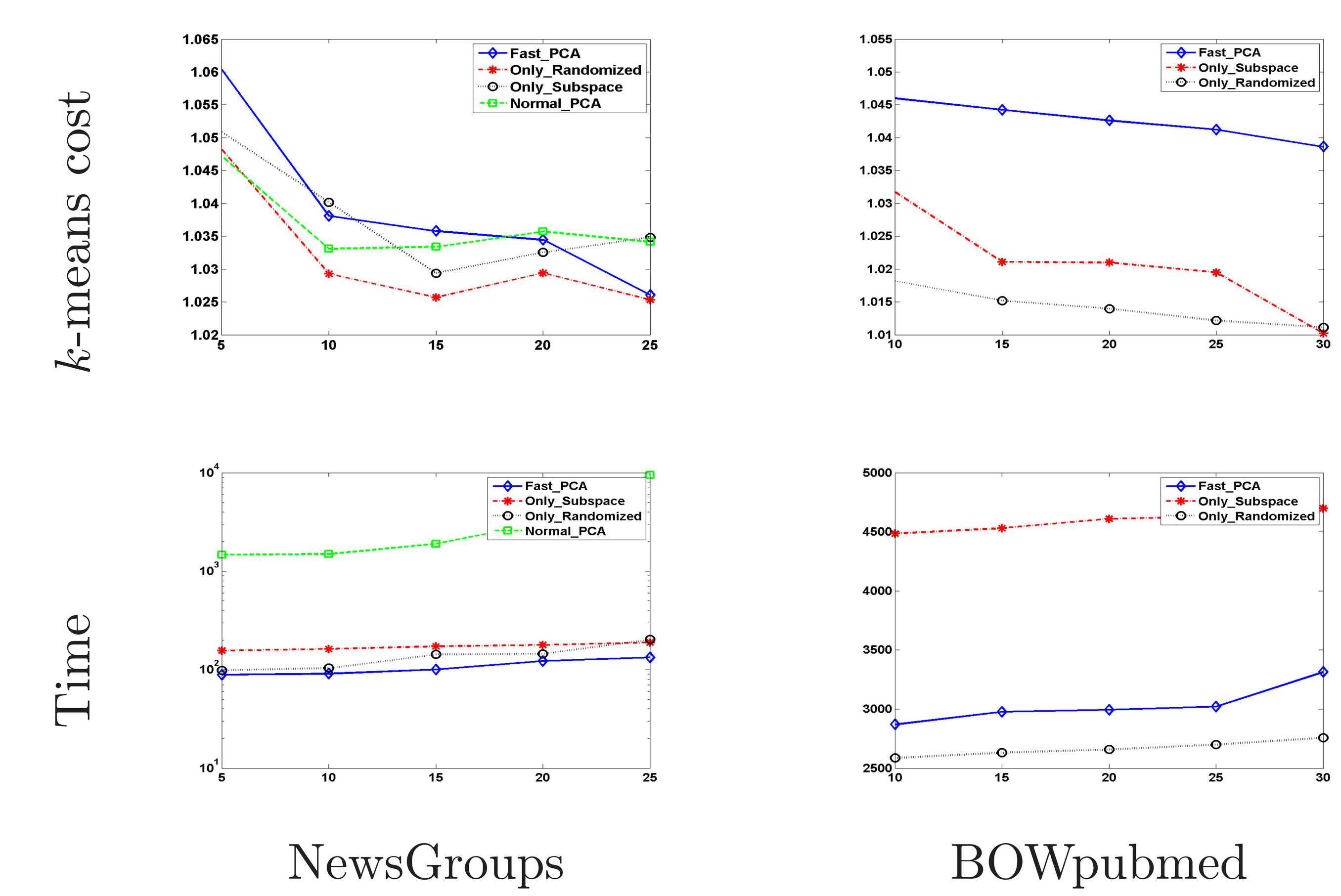
Challenge: boost the constant success prob to high prob

- Solved by our new cross validation style technique
- Dimension and sparsity of HP independent of the success prob

Experimental Results

▷ **Performance:** cost increase $< 5\%$; $\times 10$ to $\times 100$ speedup

▷ **k-Means Clustering**: k-means cost/time vs dimension



▷ **Principal Component Regression**: PCR cost/time vs dimension