Differentially-private Distributed Principal Component Analysis

Hafiz Imtiaz & Anand D. Sarwate

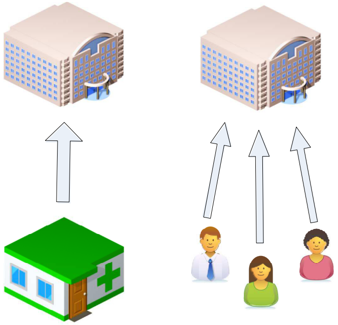
Department of Electrical and Computer Engineering
Rutgers, the State University of New Jersey

April 17, 2018

Outline

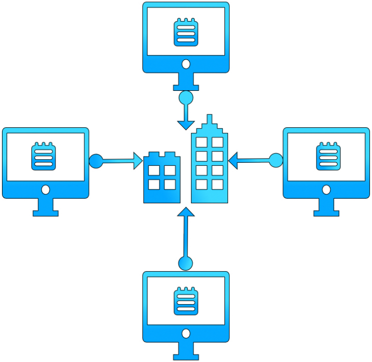
1. Motivation
2. Problem Formulation
3. Differential Privacy
4. Proposed Algorithm
5. Experimental Results
6. Conclusion

Why learn from private data?



- Much of private/sensitive data is being digitized
- Want to learn about population – using/reusing data
- Free and open sharing – ethical, legal, and technological obstacles

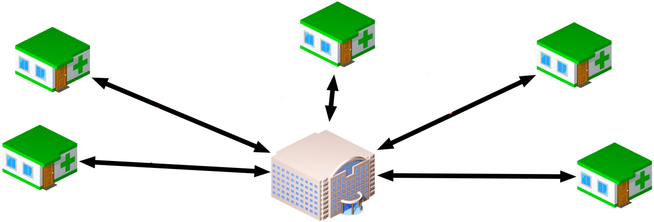
Why learn in distributed setting?



Good feature learning requires large sample sizes.

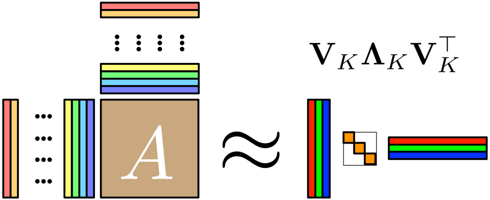
- Data at a single site may not be sufficient for statistical learning
- Pooling data in one location may not be possible

An example in neuroimaging



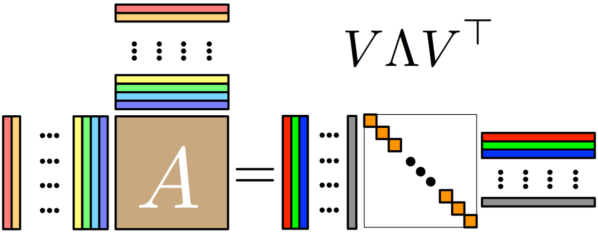
- Multiple fMRI collection centers
- Each has a moderate number of samples, at best
- **Goal:** find a way to reduce the sample dimension

We can perform principal component analysis (PCA)



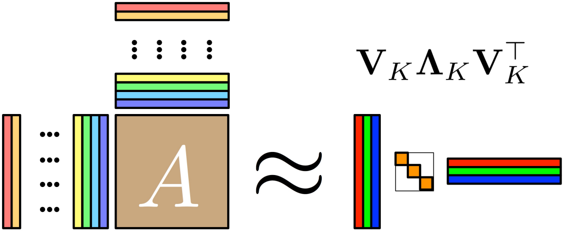
Principal Component Analysis

The PCA problem: pooled case



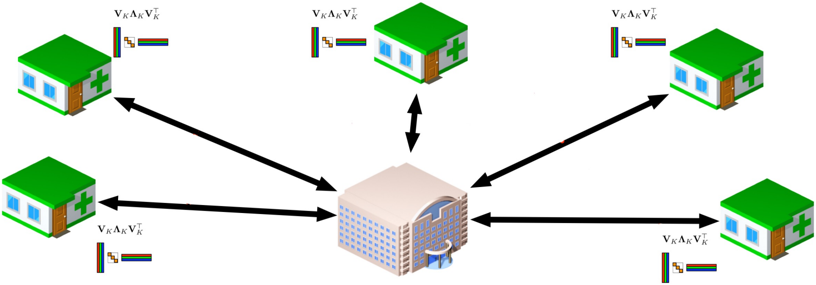
- Data matrix: $\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \ldots \ \mathbf{x}_N] \in \mathbb{R}^{D \times N}$
- Second-moment matrix: $\mathbf{A} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top$
- We can decompose $\mathbf{A}$ as: $\mathbf{A} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$
- Here, $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_D)$ and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D \geq 0$

The PCA problem: pooled case



- The best rank-K approximation of $\mathbf{A}$: $\mathbf{A}_K = \mathbf{V}_K \mathbf{\Lambda}_K \mathbf{V}_K^\top$
- The top-K PCA subspace is the span of the corresponding columns of $\mathbf{V}$: $\mathbf{V}_K(\mathbf{A})$

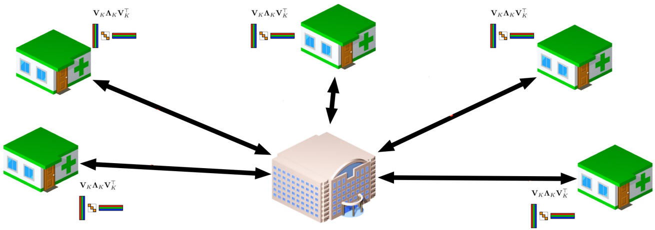
The PCA problem: distributed case



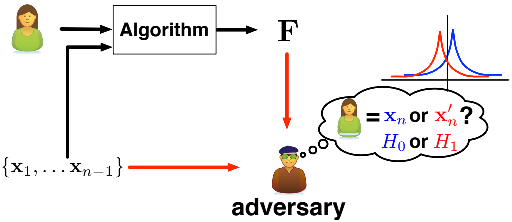
- One aggregator, S different sites with disjoint datasets
- Local data matrix: $\mathbf{X}_s = [\mathbf{x}_{s,1} \ldots \mathbf{x}_{s,N_s}] \in \mathbb{R}^{D \times N_s}$
- Local second-moment matrix: $\mathbf{A}_s = \frac{1}{N_s} \mathbf{X}_s \mathbf{X}_s^\top$
- All parties: "nice but curious"

How can we compute a **global** $\mathbf{V}_K$?

What are our options?

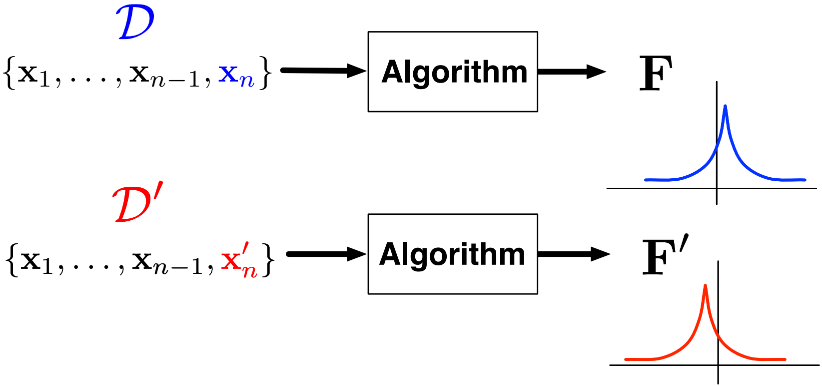


- Send $\mathbf{X}_s$ to aggregator
  - $\rightarrow$ huge communication cost
  - $\rightarrow$ privacy violation
- Compute $\mathbf{V}_K$ using local data
  - $\rightarrow$ poor quality of the subspace



Differential Privacy

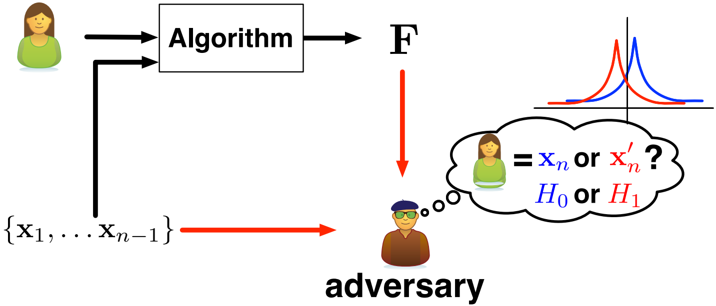
Differential privacy: a definition



[Dwork et al. 2006] An algorithm $\mathcal{A}$ is (ε, δ)-differentially private if for any set of outputs $\mathcal{F}$, and all $(\mathcal{D}, \mathcal{D}')$ differing in a single point,

$$\mathbb{P}\left(\mathcal{A}(\mathcal{D}) \in \mathcal{F}\right) \leq \exp(\varepsilon) \cdot \mathbb{P}\left(\mathcal{A}(\mathcal{D}') \in \mathcal{F}\right) + \delta$$

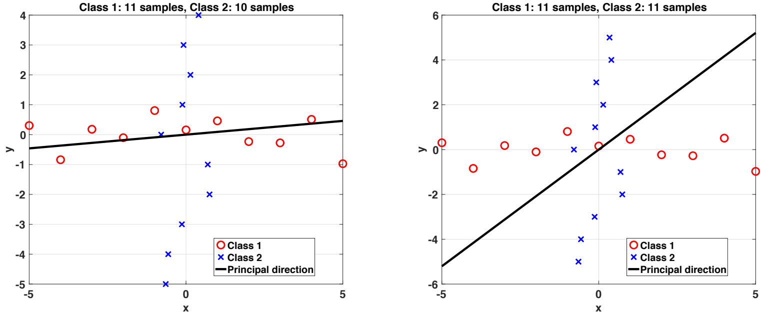
Differential privacy: hypothesis testing



$$\log \frac{\mathbb{P}\left(\mathcal{A}(\mathcal{D}) \in \mathcal{F}\right)}{\mathbb{P}\left(\mathcal{A}(\mathcal{D}') \in \mathcal{F}\right)} \leq \varepsilon$$

We want to design algorithms that satisfy differential privacy

Why we need privacy in PCA?



Changing one sample can significantly change the principal direction

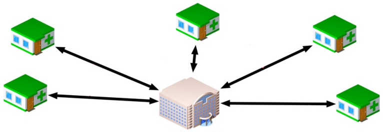
What are we trying to address?

Goal: compute an accurate $\mathbf{V}_K$

- want to exploit all samples across all sites
- want a lower communication cost
- want to preserve a formal privacy definition

Idea: send the differentially private partial square root of $\mathbf{A}_s$

- Compute $\mathbf{A}_s \leftarrow \frac{1}{N_s}\mathbf{X}_s\mathbf{X}_s^\top$
- Generate $D \times D$ symmetric matrix $\mathbf{E}$
- Compute $\hat{\mathbf{A}}_s \leftarrow \mathbf{A}_s + \mathbf{E}$
- Perform SVD $\hat{\mathbf{A}}_s = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top$
- Compute $\mathbf{P}_s \leftarrow \mathbf{U}_R\mathbf{\Sigma}_R^{\frac{1}{2}}$
- Send $\mathbf{P}_s$ to aggregator



- Compute $\mathbf{A}_c \leftarrow \frac{1}{S}\sum_{s=1}^{S}\mathbf{P}_s\mathbf{P}_s^\top$
- Perform SVD $\mathbf{A}_c = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$

Output: Differentially-private rank-K subspace $\mathbf{V}_K$

Input: Data matrix $\mathbf{X}_s$ for $s \in [S]$; privacy parameters ϵ, δ; intermediate dimension R; reduced dimension K

Proposed Algorithm

Differentially-private Distributed PCA (DPdisPCA)

Input: Data matrix $\mathbf{X}_s$ for $s \in [S]$; privacy parameters ϵ, δ; intermediate dimension R; reduced dimension K

→ for $s = 1, 2, \ldots, S$ do :

- Compute $\mathbf{A}_s \leftarrow \frac{1}{N_s}\mathbf{X}_s\mathbf{X}_s^\top$
- Generate $D \times D$ symmetric matrix $\mathbf{E}$ where $\{\mathbf{E}_{ij} : i \in [D], j \leq i\}$ drawn i.i.d. $\sim \mathcal{N}(0, \Delta_{\epsilon,\delta}^2)$ and $\Delta_{\epsilon,\delta} = \frac{1}{N_s\epsilon}\sqrt{2\log(\frac{1.25}{\delta})}$
- Compute $\hat{\mathbf{A}}_s \leftarrow \mathbf{A}_s + \mathbf{E}$
- Perform SVD $\hat{\mathbf{A}}_s = \mathbf{U\Sigma U}^\top$
- Compute $\mathbf{P}_s \leftarrow \mathbf{U}_R\mathbf{\Sigma}_R^{\frac{1}{2}}$; send $\mathbf{P}_s$ to aggregator

→ Compute $\mathbf{A}_c \leftarrow \frac{1}{S}\sum_{s=1}^{S}\mathbf{P}_s\mathbf{P}_s^\top$

→ Perform SVD $\mathbf{A}_c = \mathbf{V\Lambda V}^\top$

Output: Differentially-private rank-K subspace $\mathbf{V}_K$

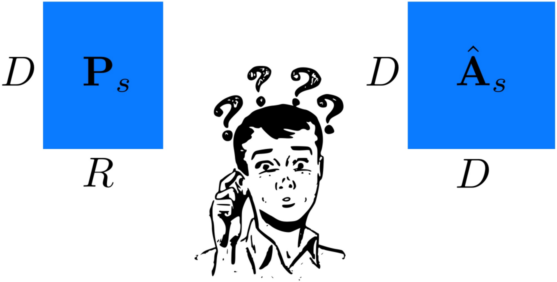
Privacy guarantee of DPdisPCA

Theorem (Privacy of DPdisPCA Algorithm)

DPdisPCA *computes an* (ϵ, δ) *differentially private approximation to the optimal subspace* $\mathbf{V}_K(\mathbf{A})$.

- $\mathcal{L}_2$ sensitivity of $\mathbf{A}_s$ is $\frac{1}{N_s}$
- By AG [Dwork et al. 2014] algorithm: computation of $\hat{\mathbf{A}}_s$ is (ϵ, δ) differentially private
- Differential-privacy is invariant to post-processing: computation of $\mathbf{V}_K$ also satisfies (ϵ, δ) differential privacy

Some comments on DPdisPCA



- $\mathbf{P}_s$ is $D \times R$: communication cost is proportional to $S \times D \times R$
- If we send $\hat{\mathbf{A}}_s$, the cost would be proportional to $S \times D^2$. Typically, $K < R < D$
- Sending $\mathbf{P}_s$ instead of $\hat{\mathbf{A}}_s$ does introduce some errors – cost of cheaper communication

Experimental Results

Datasets

- *Synthetic* dataset ($D = 200,\ K = 50$) generated with zero mean and a pre-determined covariance matrix
- *MNIST* dataset ($D = 784,\ K = 50$) (MNIST)
- *Covertype* dataset ($D = 54,\ K = 10$) (COVTYPE)

The trade-offs

We are interested to find out:

- how performance varies with "privacy risk" ϵ
- how performance varies with sample size N_s

Performance measures

Table: Notation of performance measures

Algorithm / Setting	Performance Index
Pooled Data	q_{pooled}
DPdisPCA	q_{DPdisPCA}
Local Data	q_{local}
Sending $\hat{\mathbf{A}}_s$	q_{full}

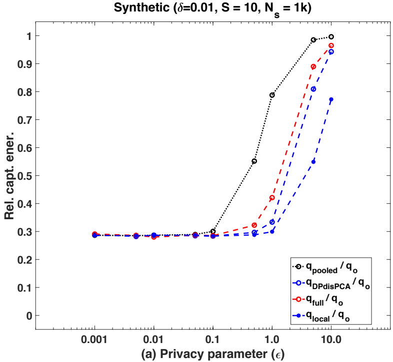
- Quality of a subspace $\mathbf{V}$: *captured energy* of $\mathbf{A}$

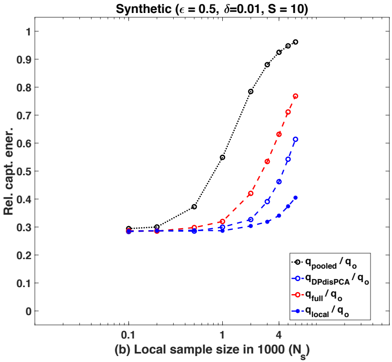
$$q(\mathbf{V}) = \operatorname{tr}(\mathbf{V}^\top \mathbf{A} \mathbf{V})$$

- We plot the ratio of these quantities with respect to the true captured energy q_o

Performance variation

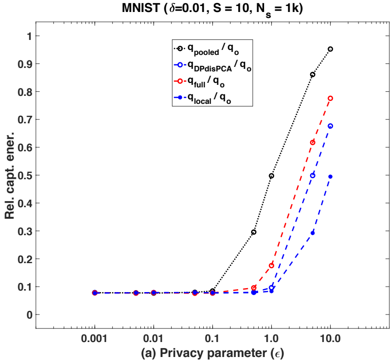
For synthetic data

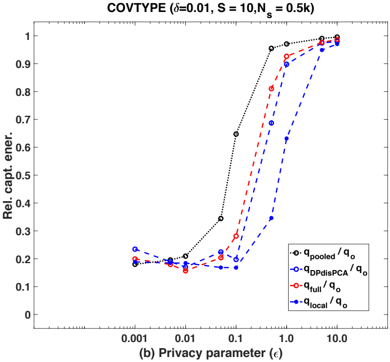




Performance variation

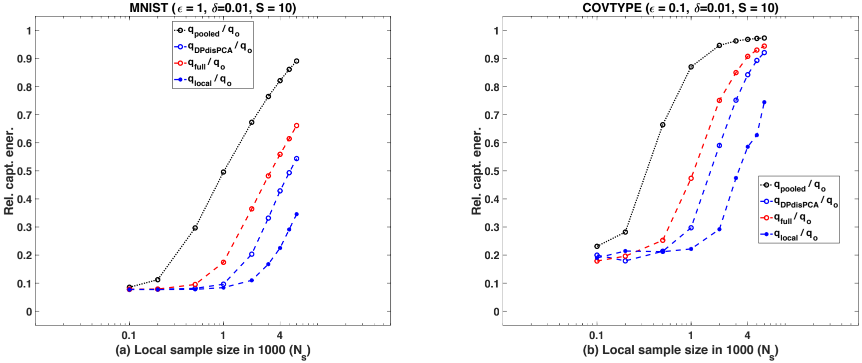
with ϵ





Performance variation

with N_s



Concluding Remarks

Concluding remarks

- The distributed algorithm clearly outperforms the local PCA algorithm
- Increasing ϵ improves performance at the cost of lower privacy
- Datasets with lower D allows smaller ϵ for achieving the same utility
- Increasing N_s improves performance for a fixed privacy level
- The cost of sending $\mathbf{P}_s$ instead of $\hat{\mathbf{A}}_s$ is noticeable in all datasets

Questions

Thank you