



RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY



Scalable Algorithms for Distributed Principal Component Analysis

July 25, 2022

ACM PODC 2022 Workshop on Principles of Distributed Learning

Waheed U. Bajwa

Department of Electrical and Computer Engineering

Rutgers University–New Brunswick



<http://www.inspirelab.us>





RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY



Alternative Talk Title

**Scalable Algorithms for Distributed Singular Value
Decomposition (SVD)**

July 25, 2022

ACM PODC 2022 Workshop on Principles of Distributed Learning

Waheed U. Bajwa

Department of Electrical and Computer Engineering

Rutgers University–New Brunswick



<http://www.inspirelab.us>



Collaborators and Selected Papers



Haroon Raja

1. H. Raja and W. U. Bajwa, “[Cloud K-SVD: A collaborative dictionary learning algorithm for big, distributed data](#),” *IEEE Transactions on Signal Processing*, vol. 64, no. 1, pp. 173–188, Jan. 2016, doi: 10.1109/TSP.2015.2472372.
2. H. Raja and W. U. Bajwa, “[Distributed stochastic algorithms for high-rate streaming principal component analysis](#),” *arXiv preprint arXiv:2001.01017*, Jan. 2020. Available: <http://arxiv.org/abs/2001.01017>
3. A. Gang, B. Xiang, and W. U. Bajwa, “[Distributed principal subspace analysis for partitioned big data: Algorithms, analysis, and implementation](#),” *IEEE Transactions on Signal and Information Processing over Networks*, vol. 7, pp. 699–715, Oct. 2021, doi: 10.1109/TSIPN.2021.3122297.
4. A. Gang and W. U. Bajwa, “[A linearly convergent algorithm for distributed principal component analysis](#),” *EURASIP Journal on Signal Processing*, vol. 193, p. 108408, Apr. 2022, doi: 10.1016/j.sigpro.2021.108408.
5. A. Gang and W. U. Bajwa, “[FAST-PCA: A fast and exact algorithm for distributed principal component analysis](#),” *arXiv preprint arXiv:2108.12373v2*, Feb. 2022, doi: 10.48550/arXiv.2108.12373.



Bingqing Xiang



Arpita Gang

Latest list of relevant papers from the INSPIRE Lab: [Google Scholar Page](#)

The Role of Feature Representation in Machine Learning

Success of any machine learning (ML) algorithm largely depends on the input data (or feature) 'representation'

A good feature representation method has one/some of the attributes:

- Tackles curse of dimensionality
- Uncorrelatedness/Disentanglement
- Abstraction
- Transferable
- Etc.

The Role of Feature Representation in Machine Learning

Success of any machine learning (ML) algorithm largely depends on the input data (or feature) 'representation'

A good feature representation method has one/some of the attributes:

- Tackles curse of dimensionality
- Uncorrelatedness/Disentanglement
- Abstraction
- Transferable
- Etc.

Raw data is high dimensional. But it usually lies in (or near) low-dimensional spaces.

- Finding the low-dimensional representations makes data easier to process

The Role of Feature Representation in Machine Learning

Success of any machine learning (ML) algorithm largely depends on the input data (or feature) 'representation'

A good feature representation method has one/some of the attributes:

- Tackles curse of dimensionality
- Uncorrelatedness/Disentanglement
- Abstraction
- Transferable
- Etc.

Raw data is high dimensional. But it usually lies in (or near) low-dimensional spaces.

- Finding the low-dimensional representations makes data easier to process

In parallel, learning uncorrelated feature representations are also known to help many downstream ML algorithms.^[1]

[1] Bengio Y, Courville A, Vincent P. Representation learning: a review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2013 Aug;35(8):1798-1828. DOI: 10.1109/tpami.2013.50. PMID: 23787338.

Feature Learning Methods

Principal Component Analysis (PCA): Pearson 1901

Low-Rank Matrix Factorization: Eckart, Young 1936

Linear Discriminant Analysis: Fisher 1936

Independent Component Analysis: Comon 1994

Dictionary Learning: Aharon et al. 2006

Autoencoders: Rumelhart et al. 1986

Many more ...

Feature Learning Methods

Principal Component Analysis (PCA): Pearson 1901

Low-Rank Matrix Factorization: Eckart, Young 1936

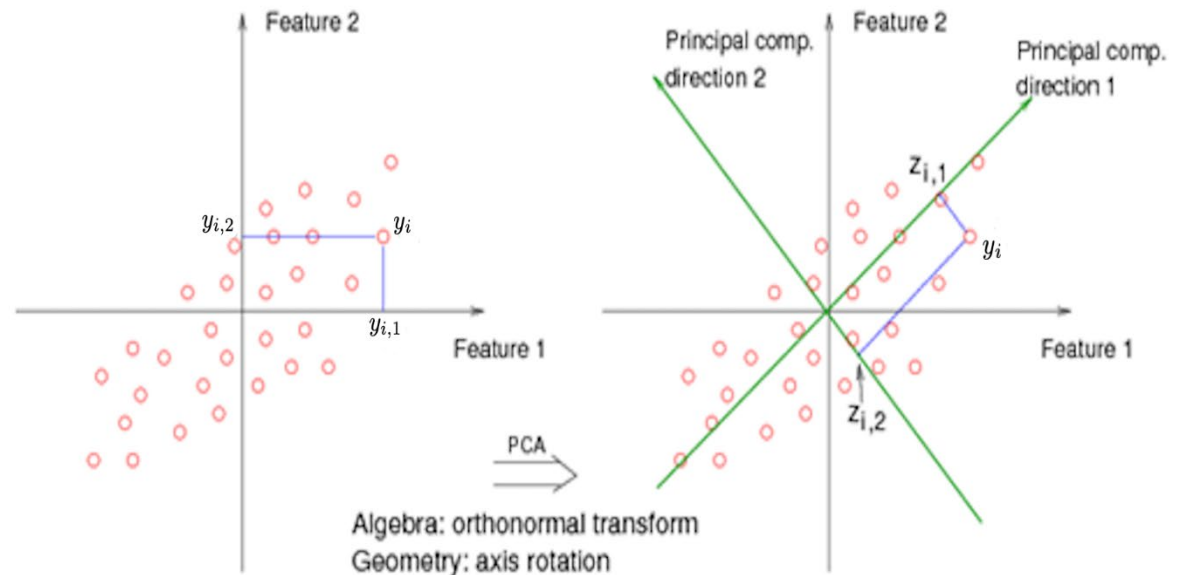
Linear Discriminant Analysis: Fisher 1936

Independent Component Analysis: Comon 1994

Dictionary Learning: Aharon et al. 2006

Autoencoders: Rumelhart et al. 1986

Many more ...



Feature Learning Methods

Principal Component Analysis (PCA): Pearson 1901

Low-Rank Matrix Factorization: Eckart, Young 1936

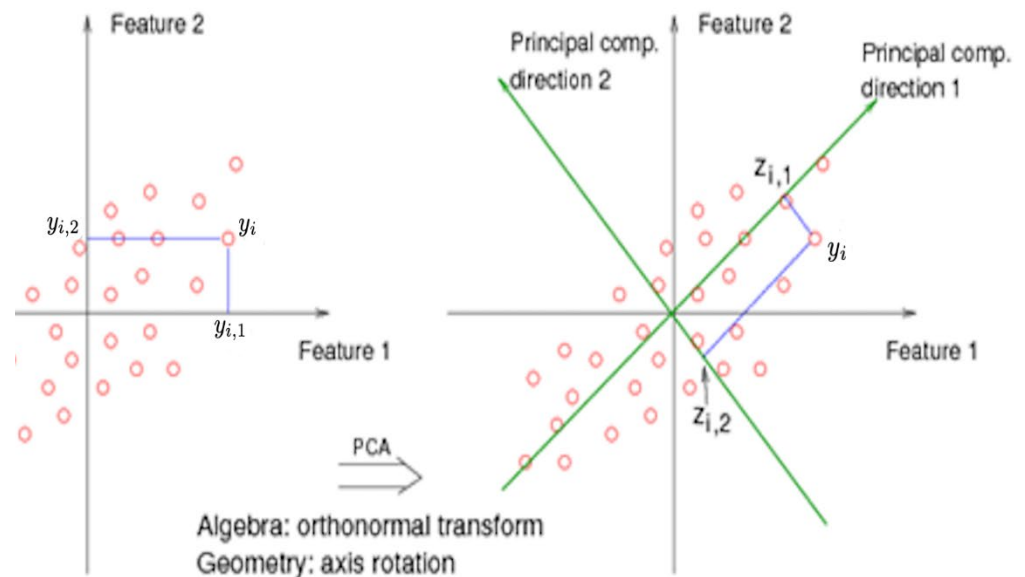
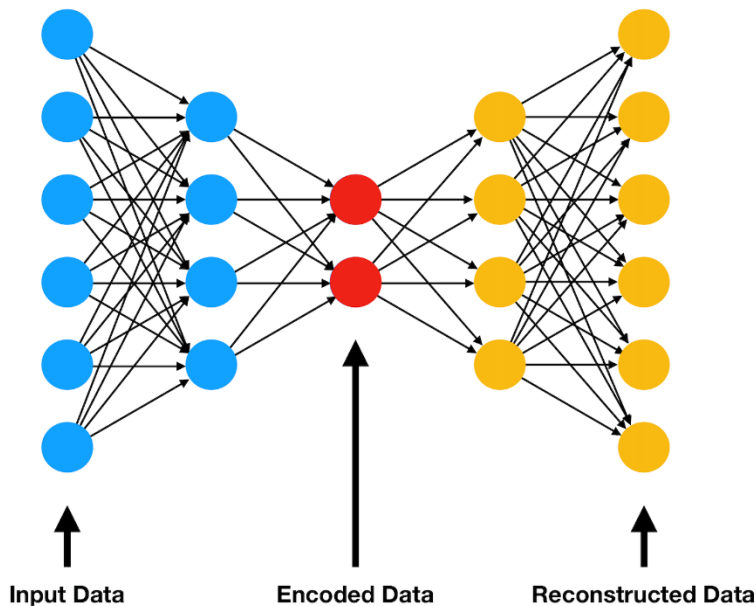
Linear Discriminant Analysis: Fisher 1936

Independent Component Analysis: Comon 1994

Dictionary Learning: Aharon et al. 2006

Autoencoders: Rumelhart et al. 1986

Many more ...



Principal Component Analysis (PCA)

An unsupervised learning technique that reduces a set of “raw” features representing a data point to a smaller set of uncorrelated features

Data setup

- Data points $\mathbf{y} \in \mathbb{R}^d$ are sampled from a distribution
- Assume zero mean and covariance $\Sigma = \mathbb{E}[\mathbf{y}\mathbf{y}^T]$

PCA finds a matrix $\mathbf{X} \in \mathbb{R}^{d \times K}$, $K \ll d$ such that

- It represents an orthogonal basis of the K -dimensional principal subspace
- It gives minimum reconstruction error representation of \mathbf{y} .
- It results in K uncorrelated features i.e., $\mathbb{E}[\mathbf{X}^T \mathbf{y}\mathbf{y}^T \mathbf{X}]$ is diagonal

The PCA problem can be formulated as:

$$\mathbf{X}^* = \underset{\mathbf{X} \in \mathbb{R}^{d \times K}}{\operatorname{arg\,min}} \quad \mathbb{E} [\|\mathbf{y} - \mathbf{X}\mathbf{X}^T \mathbf{y}\|_2^2] \quad \text{such that} \quad \forall l \neq q, \left(\mathbb{E} [\mathbf{X}^T \mathbf{y}\mathbf{y}^T \mathbf{X}] \right)_{lq} = 0.$$

Principal Component Analysis (PCA)

An unsupervised learning technique that reduces a set of “raw” features representing a data point to a smaller set of uncorrelated features

Data setup

- Data points $\mathbf{y} \in \mathbb{R}^d$ are sampled from a distribution
- Assume zero mean and covariance $\Sigma = \mathbb{E}[\mathbf{y}\mathbf{y}^T]$

PCA finds a matrix $\mathbf{X} \in \mathbb{R}^{d \times K}$, $K \ll d$ such that

- It represents an orthogonal basis of the K -dimensional principal subspace
- It gives minimum reconstruction error representation of \mathbf{y} .
- It results in K uncorrelated features i.e., $\mathbb{E}[\mathbf{X}^T \mathbf{y}\mathbf{y}^T \mathbf{X}]$ is diagonal

The PCA problem can be formulated as:

$$\mathbf{X}^* = \underset{\mathbf{X} \in \mathbb{R}^{d \times K}}{\operatorname{arg\,min}} \quad \mathbb{E} [\|\mathbf{y} - \mathbf{X}\mathbf{X}^T \mathbf{y}\|_2^2] \quad \text{such that} \quad \forall l \neq q, \left(\mathbb{E} [\mathbf{X}^T \mathbf{y}\mathbf{y}^T \mathbf{X}] \right)_{lq} = 0.$$

PCA Solution: Reduced Singular Value Decomposition (SVD) of the data matrix
(reduced Eigen Value Decomposition (EVD) of sample covariance)

PCA: A Visual Example



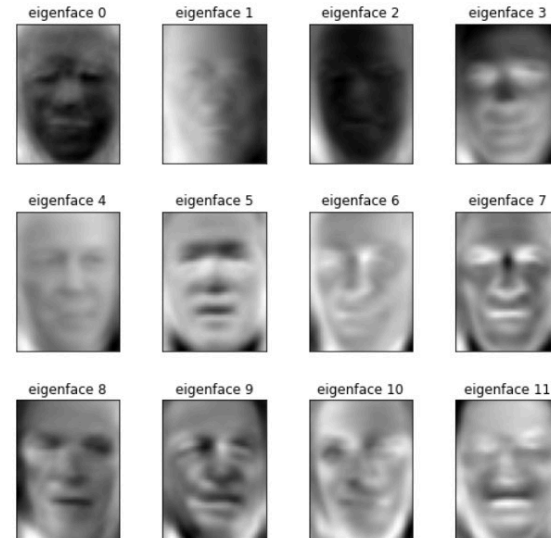
Training Image with True Label (LFW people's dataset)

Each image is 62 x 47 pixels, i.e., of dimension 2,914

PCA: A Visual Example



Training Image with True Label (LFW people's dataset)



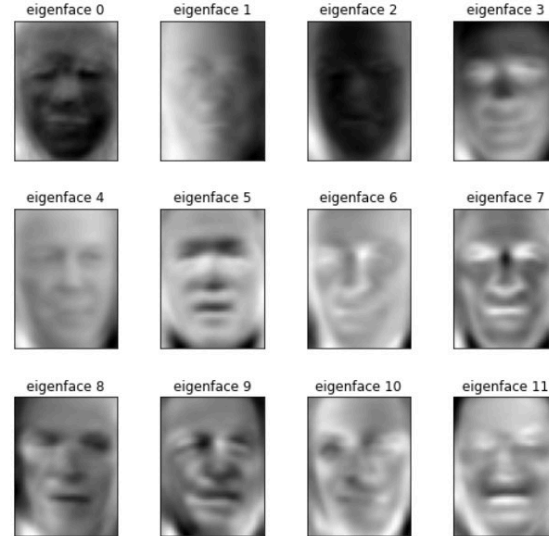
EigenFaces

Each image is 62 x 47 pixels, i.e., of dimension 2,914

PCA: A Visual Example

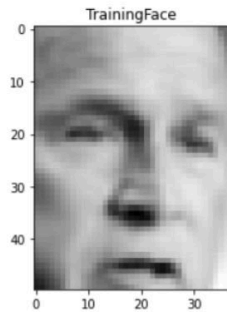


Training Image with True Label (LFW people's dataset)



EigenFaces

Each image is 62 x 47 pixels, i.e., of dimension 2,914



=

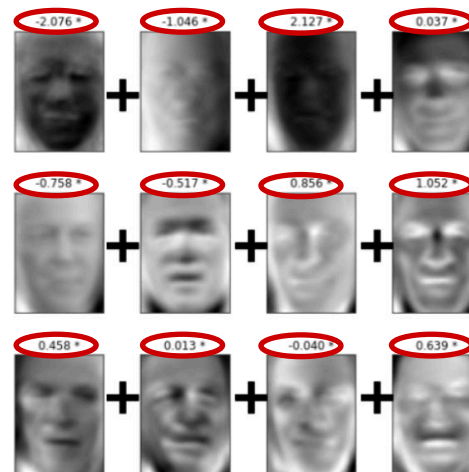


Image can be represented as a combination of 12 eigenfaces

Linear Combination of EigenFaces

Hebbian Learning Rule (Oja's Rule)

Neuroscientific rule trying to imitate the learning process of brain neurons [Hebb '49].
For a single neuron with stochastic inputs $\mathbf{y}_t, t = 1, 2, \dots$, the Hebbian update is as follows:

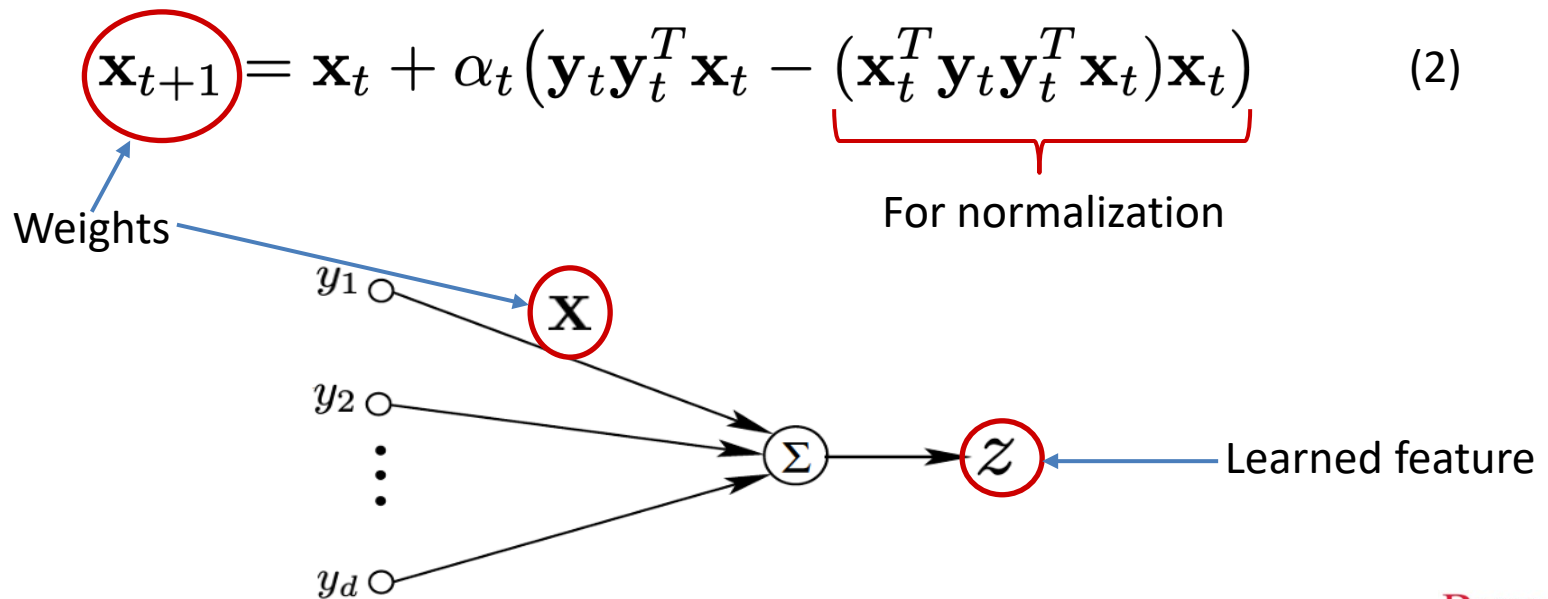
$$\begin{aligned}\mathbf{x}_{t+1} &= \mathbf{x}_t + \alpha_t \mathbf{y}_t \mathbf{y}_t^T \mathbf{x}_t \\ \mathbf{x}_{t+1} &= \frac{\mathbf{x}_{t+1}}{\|\mathbf{x}_{t+1}\|}\end{aligned}\tag{1}$$

Hebbian Learning Rule (Oja's Rule)

Neuroscientific rule trying to imitate the learning process of brain neurons [Hebb '49].
For a single neuron with stochastic inputs $\mathbf{y}_t, t = 1, 2, \dots$, the Hebbian update is as follows:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \alpha_t \mathbf{y}_t \mathbf{y}_t^T \mathbf{x}_t \quad (1)$$
$$\mathbf{x}_{t+1} = \frac{\mathbf{x}_{t+1}}{\|\mathbf{x}_{t+1}\|}$$

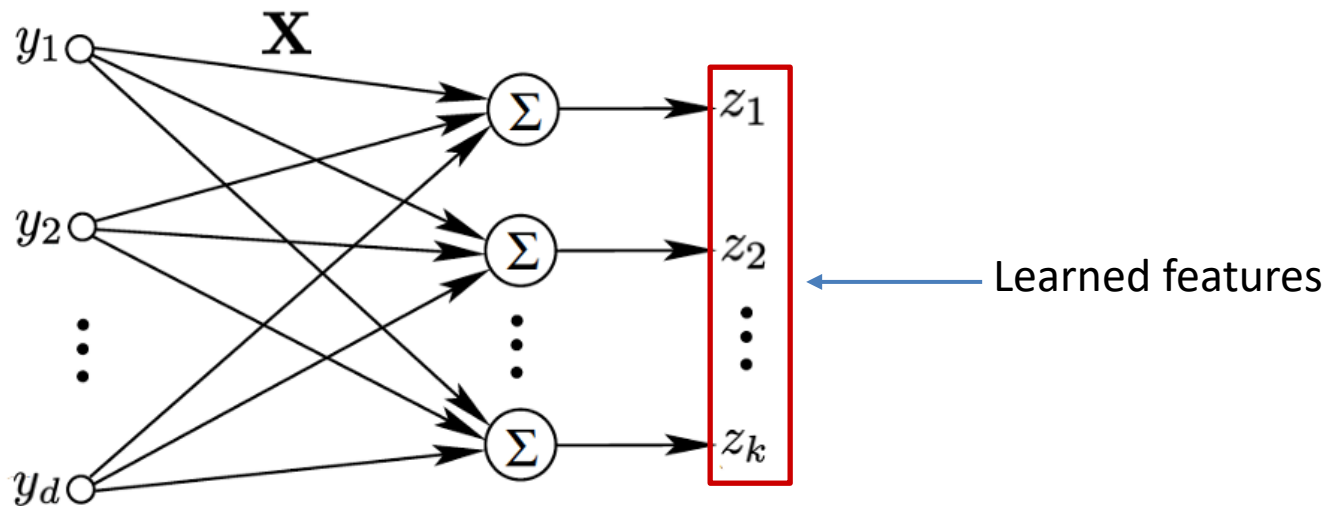
Adapted for use in autoencoders for data compression with an added normalization



Generalized Hebbian Learning Rule

First used by Oja^[1], this update rule was shown to converge to the dominant eigenvector of $\Sigma = \mathbb{E} [\mathbf{y}_t \mathbf{y}_t^T]$ asymptotically.

Later, Sanger^[2] combined the Hebbian update with Gram-Schmidt orthogonalization to give the generalized Hebbian algorithm (GHA) that can estimate multiple dominant eigenvectors.



[1] E. Oja and J. Karhunen, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," J. Math. Anal. Applicat., vol. 106, no. 1, pp. 69 – 84, 1985.

[2] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," Neural Netw., vol. 2, no. 6, pp. 459 – 473, 89.

Krasulina's Method^[1]

Another stochastic approximation for estimating the dominant eigenvector while processing one sample at a time.

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \alpha_t \left(\mathbf{y}_t \mathbf{y}_t^T \mathbf{x}_t - \frac{\mathbf{x}_t^T \mathbf{y}_t \mathbf{y}_t^T \mathbf{x}_t}{\|\mathbf{x}_t\|^2} \mathbf{x}_t \right)$$

If estimating the dominant eigenvector is posed as the optimization problem

$$\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x}) = \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^d} \frac{-\mathbf{x}^T \mathbf{y}_t \mathbf{y}_t^T \mathbf{x}}{\|\mathbf{x}\|^2} \quad (1)$$

Then Krasulina's method looks similar to applying stochastic gradient descent to (1)

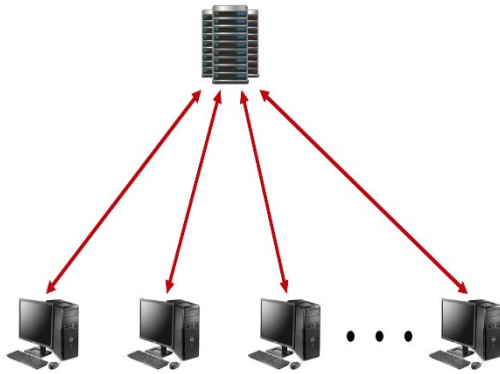
$$\nabla f(\mathbf{x}) = \frac{1}{\|\mathbf{x}\|^2} \left(-\mathbf{y}_t \mathbf{y}_t^T \mathbf{x} + \frac{\mathbf{x}^T \mathbf{y}_t \mathbf{y}_t^T \mathbf{x}}{\|\mathbf{x}\|^2} \mathbf{x} \right)$$

[1] T. P. Krasulina, "Method of stochastic approximation in the determination of the largest eigenvalue of the mathematical expectation of random matrices," *Autom. Remote Control*, vol. 1970, pp. 215–221, 1970.

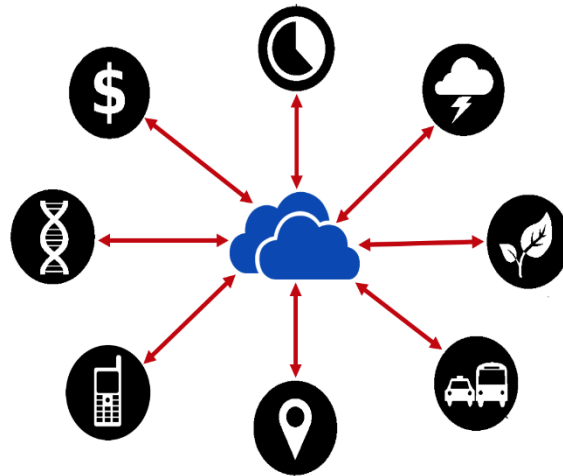
A Challenge: Data is Massive and Distributed

Modern (massive) datasets end up being distributed for various reasons

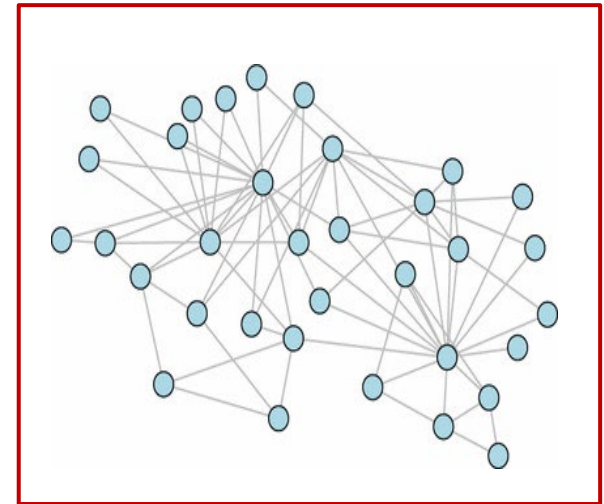
- **Parallel computing**—data gets distributed for storage reasons and computational speed ups
- **Federated systems**—multiple sensors collect data; system uses a central coordinating node
- **Distributed systems**—multiple sensors collect data; system lacks a central server



Parallel computing



Federated system
(e.g., sensor network)




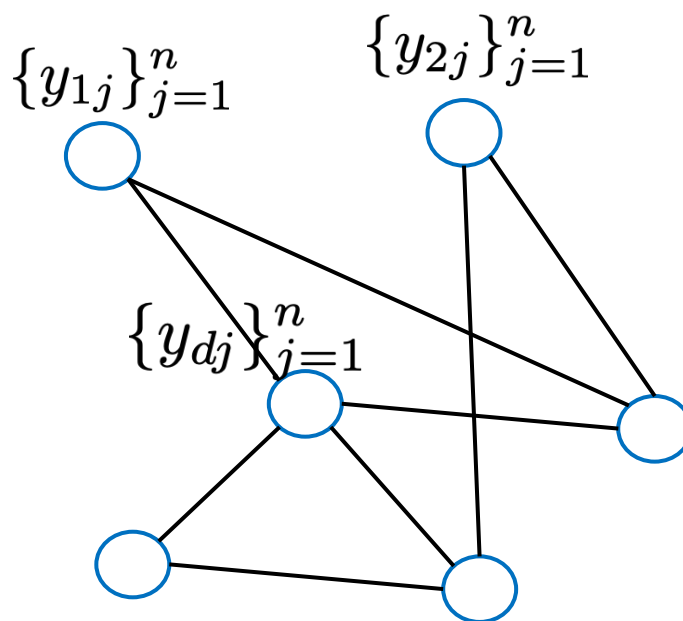
Distributed system
(e.g., IoT system)

Types of Data Distribution

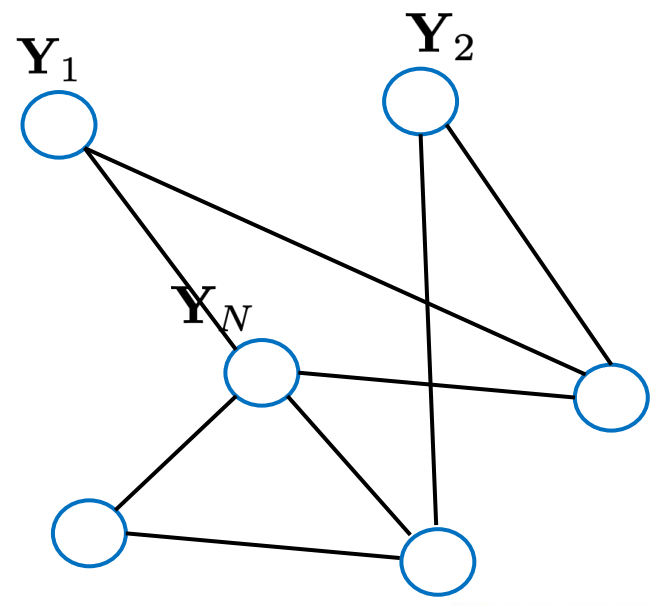
By **Features**: Each node has some features of the data samples; nodes then learn parts of the low-dimensional space

By **Samples**: Each node has some samples; each node learns the complete low dimensional space


$$\mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & \dots & y_{1n} \\ y_{21} & y_{22} & \dots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{d1} & y_{d2} & \dots & y_{dn} \end{bmatrix}$$



Feature-wise Split

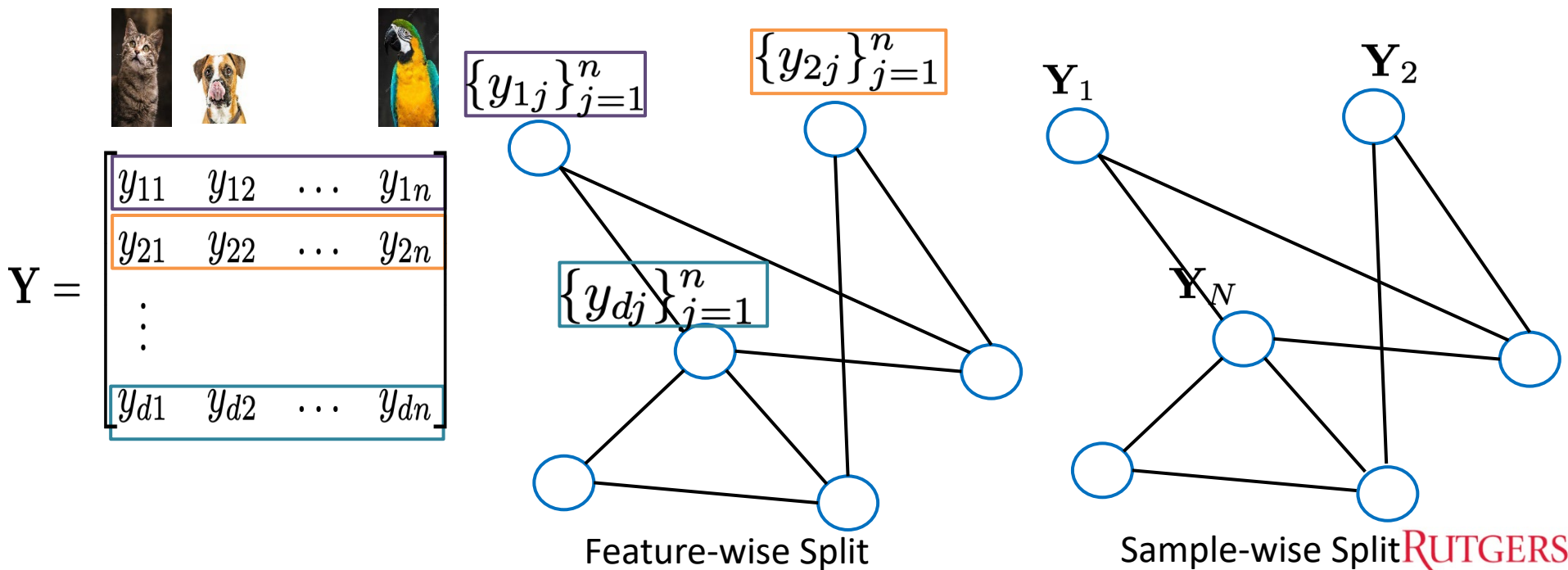


Sample-wise Split

Types of Data Distribution

By **Features**: Each node has some features of the data samples; nodes then learn parts of the low-dimensional space

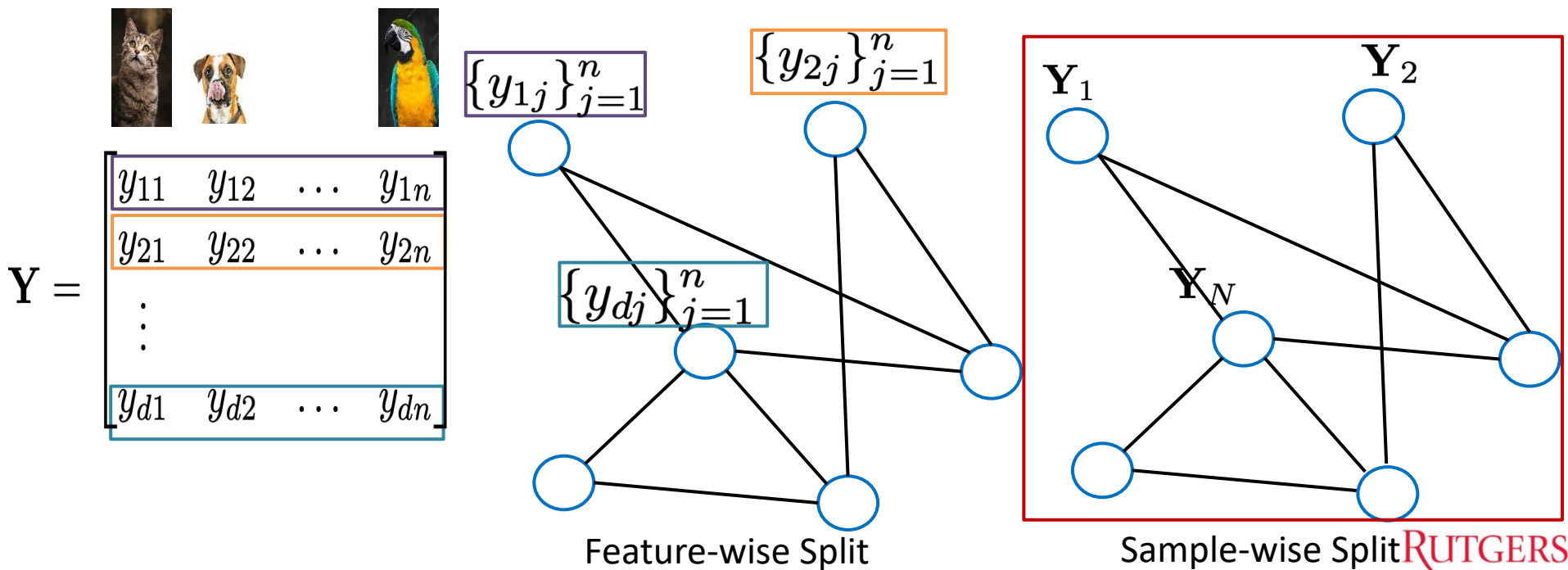
By **Samples**: Each node has some samples; each node learns the complete low dimensional space



Types of Data Distribution

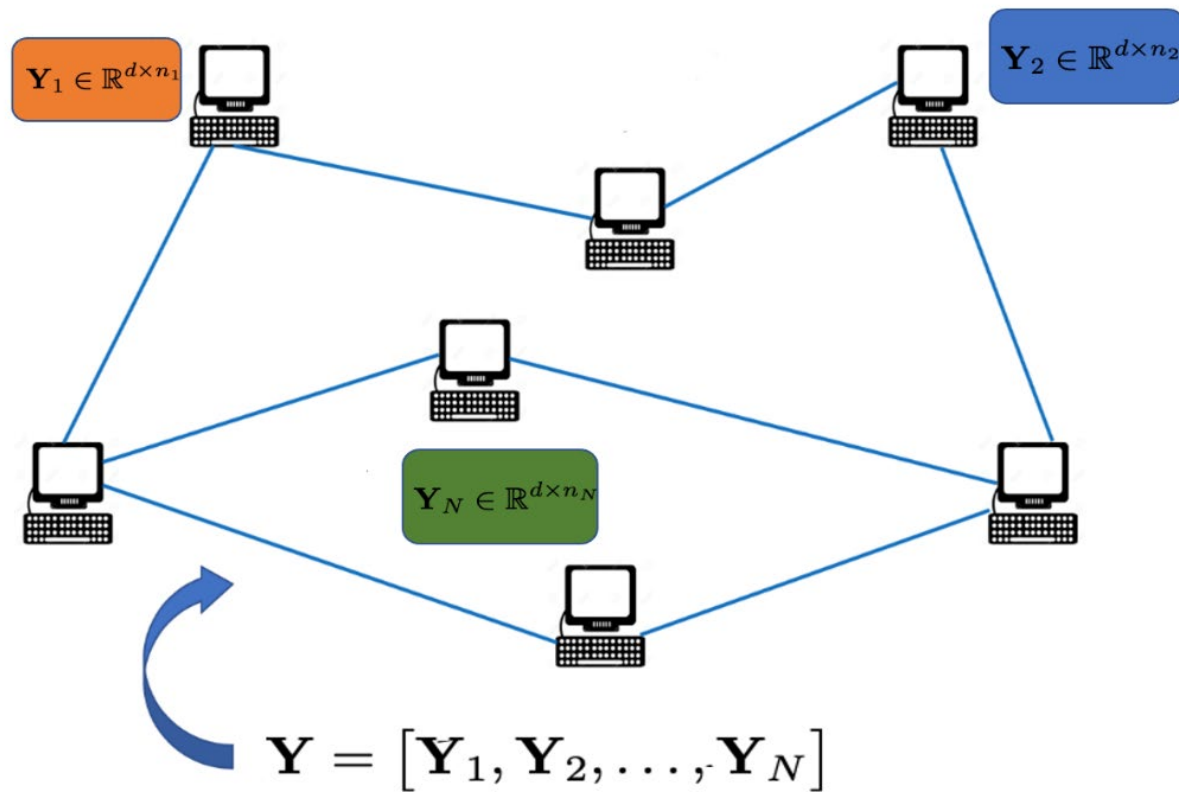
By **Features**: Each node has some features of the data samples; nodes then learn parts of the low-dimensional space

By **Samples**: Each node has some samples; each node learns the complete low dimensional space



Distributed PCA in Batch Settings

When data are *'massive, high-dimensional and sample-wise distributed'*



Distributed Setup and Goal

We assume the following setup:

- A set of N nodes connected in an arbitrary network.
- The graph underlying the network is undirected.
- Samples $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_n] \in \mathbb{R}^{d \times n}$ are scattered in the network.
- Each node i has n_i samples in $\mathbf{Y}_i \in \mathbb{R}^{d \times n_i}$ (Local covariance matrix $\mathbf{C}_i = \mathbf{Y}_i \mathbf{Y}_i^T$)

Nodes collaboratively learn the eigenvectors of $\mathbf{C} = \sum_i \mathbf{C}_i$

$$\underset{\mathbf{X}_i \in \mathbb{R}^{d \times K}, \mathbf{X}_i^T \mathbf{X}_i = \mathbf{I}}{\operatorname{argmin}} \sum_{i=1}^M \|\mathbf{Y}_i - \mathbf{X}_i \mathbf{X}_i^T \mathbf{Y}_i\|_F^2 \quad \text{s.t. } \forall j \in \mathcal{N}_i, \mathbf{X}_i = \mathbf{X}_j \quad \text{and} \quad \forall l \neq q, \left(\mathbf{X}_i^T \left(\sum_{i=1}^M \mathbf{Y}_i \mathbf{Y}_i^T \right) \mathbf{X}_i \right)_{lq} = 0$$

Recipe for a Good Distributed PCA Algorithm

Computationally inexpensive steps

- One that does not perform any computationally expensive steps

Communication efficient

- One that exchanges smaller sized messages between nodes
- One that does not require too many exchange of messages

Provable convergence

- One that converges to the true eigenvectors of the covariance matrix at a linear rate (exponentially fast) when the error metric is the angle between the true and estimated eigenvectors

Existing Approaches

Centralized solutions for PCA

- Power Method, Orthogonal Iteration: Golub '83
- Hebbian rule based methods: Oja '82, Sanger '89, APEX model
- Krasulina's Method: Krasulina' 70

} “Nice” convergence guarantees

BUT, we want distributed solutions!

Distributed solutions for PCA

- Early PCA: Fellus et al. '14
 - Involves doing SVD in each iteration, finds principal subspace
- Distributed Power Method: Raja-Bajwa '16, Wai et al. '17^[1]
 - Only finds dominant eigenvector, two-time step method
- Distributed Orthogonal Iterations: Gang-Xiang-Bajwa '21
 - Finds the principal subspace only, two-time step method
- DeEPCA: Ye-Zhang '21^[2]
 - Finds the principal subspace only, two-time step method

[1] H. Wai, A. Scaglione, J. Lafond, and E. Moulines, “Fast and privacy preserving distributed low-rank regression,” in Proc. IEEE Int. Conf. Acoustics, Speech and Signal Process., (ICASSP), 2017, pp. 4451–4455.

[2] H. Ye and T. Zhang, “DeEPCA: Decentralized exact PCA with linear convergence rate,” Journal of Machine Learning Research, pp. 1–27, 2021.

Summary of the 'Best' Existing Solutions

	Comm./iteration	Number of Iterations	Total Comm.
DistSeqPM (PCA)	$\mathcal{O}\left(\frac{K}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{K}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{K^2}{\log^2 \text{gap}_r^{-1}} \log^2 \frac{1}{\epsilon}\right)$
S-DOT (PSA)	$\mathcal{O}\left(\frac{1}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\log^2 \text{gap}_r^{-1}} \log^2 \frac{1}{\epsilon}\right)$
DeEPCA (PSA)	$\mathcal{O}\left(\log \frac{1}{\text{gap}}\right)$	$\mathcal{O}\left(\frac{1}{\text{gap}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\text{gap}} \log \frac{1}{\text{gap}} \log \frac{1}{\epsilon}\right)$

$$\text{gap}_r = \frac{\lambda_{K+1}}{\lambda_K} \quad \text{for PSA}$$

$$\text{gap}_r = \max_{k=1, \dots, K} \frac{\lambda_{k+1}}{\lambda_k} \quad \text{for PCA}$$

$$\text{gap} = \lambda_K - \lambda_{K+1} \quad \text{for PSA}$$

$$(\lambda_1, \lambda_2, \dots, \lambda_d \text{ are the eigenvalues of } \mathbf{C} = \sum_i \mathbf{C}_i)$$

DistSeqPM (Distributed Sequential Power Method): Extension of DePM (Raja-Bajwa '16, Wai et al. '17)

S-DOT (Sample-wise Distributed Orthogonal Iterations): Gang-Xiang-Bajwa '21

DeEPCA (Decentralized Exact PCA): Ye-Zhang '21

Summary of the 'Best' Existing Solutions

	Comm./iteration	Number of Iterations	Total Comm.
DistSeqPM (PCA)	$\mathcal{O}\left(\frac{K}{\log gap_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{K}{\log gap_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{K^2}{\log^2 gap_r^{-1}} \log^2 \frac{1}{\epsilon}\right)$
S-DOT (PSA)	$\mathcal{O}\left(\frac{1}{\log gap_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\log gap_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\log^2 gap_r^{-1}} \log^2 \frac{1}{\epsilon}\right)$
DeEPCA (PSA)	$\mathcal{O}\left(\log \frac{1}{gap}\right)$	$\mathcal{O}\left(\frac{1}{gap} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{gap} \log \frac{1}{gap} \log \frac{1}{\epsilon}\right)$

$$gap_r = \frac{\lambda_{K+1}}{\lambda_K} \quad \text{for PSA}$$

$$gap_r = \max_{k=1, \dots, K} \frac{\lambda_{k+1}}{\lambda_k} \quad \text{for PCA}$$

$$gap = \lambda_K - \lambda_{K+1} \quad \text{for PSA}$$

$$(\lambda_1, \lambda_2, \dots, \lambda_d \text{ are the eigenvalues of } \mathbf{C} = \sum_i \mathbf{C}_i)$$

DistSeqPM (Distributed Sequential Power Method): Extension of DePM (Raja-Bajwa '16, Wai et al. '17)

S-DOT (Sample-wise Distributed Orthogonal Iterations): Gang-Xiang-Bajwa '21

DeEPCA (Decentralized Exact PCA): Ye-Zhang '21

Summary of the 'Best' Existing Solutions

	Comm./iteration	Number of Iterations	Total Comm.
DistSeqPM (PCA)	$\mathcal{O}\left(\frac{K}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{K}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{K^2}{\log^2 \text{gap}_r^{-1}} \log^2 \frac{1}{\epsilon}\right)$
S-DOT (PSA)	$\mathcal{O}\left(\frac{1}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\log^2 \text{gap}_r^{-1}} \log^2 \frac{1}{\epsilon}\right)$
DeEPCA (PSA)	$\mathcal{O}\left(\log \frac{1}{\text{gap}}\right)$	$\mathcal{O}\left(\frac{1}{\text{gap}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\text{gap}} \log \frac{1}{\text{gap}} \log \frac{1}{\epsilon}\right)$

$$\text{gap}_r = \frac{\lambda_{K+1}}{\lambda_K} \quad \text{for PSA}$$

$$\text{gap}_r = \max_{k=1, \dots, K} \frac{\lambda_{k+1}}{\lambda_k} \quad \text{for PCA}$$

$$\text{gap} = \lambda_K - \lambda_{K+1} \quad \text{for PSA}$$

$$(\lambda_1, \lambda_2, \dots, \lambda_d \text{ are the eigenvalues of } \mathbf{C} = \sum_i \mathbf{C}_i)$$

DistSeqPM (Distributed Sequential Power Method): Extension of DePM (Raja-Bajwa '16, Wai et al. '17)

S-DOT (Sample-wise Distributed Orthogonal Iterations): Gang-Xiang-Bajwa '21

DeEPCA (Decentralized Exact PCA): Ye-Zhang '21

Summary of the 'Best' Existing Solutions

	Comm./iteration	Number of Iterations	Total Comm.
DistSeqPM (PCA)	$\mathcal{O}\left(\frac{K}{\log gap_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{K}{\log gap_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{K^2}{\log^2 gap_r^{-1}} \log^2 \frac{1}{\epsilon}\right)$
S-DOT (PSA)	$\mathcal{O}\left(\frac{1}{\log gap_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\log gap_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\log^2 gap_r^{-1}} \log^2 \frac{1}{\epsilon}\right)$
DeEPCA (PSA)	$\mathcal{O}\left(\log \frac{1}{gap}\right)$	$\mathcal{O}\left(\frac{1}{gap} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{gap} \log \frac{1}{gap} \log \frac{1}{\epsilon}\right)$

$$gap_r = \frac{\lambda_{K+1}}{\lambda_K} \quad \text{for PSA}$$

$$gap_r = \max_{k=1, \dots, K} \frac{\lambda_{k+1}}{\lambda_k} \quad \text{for PCA}$$

$$gap = \lambda_K - \lambda_{K+1} \quad \text{for PSA}$$

$$(\lambda_1, \lambda_2, \dots, \lambda_d \text{ are the eigenvalues of } \mathbf{C} = \sum_i \mathbf{C}_i)$$

DistSeqPM (Distributed Sequential Power Method): Extension of DePM (Raja-Bajwa '16, Wai et al. '17)

S-DOT (Sample-wise Distributed Orthogonal Iterations): Gang-Xiang-Bajwa '21

DeEPCA (Decentralized Exact PCA): Ye-Zhang '21

Summary of the 'Best' Existing Solutions

	Comm./iteration	Number of Iterations	Total Comm.
DistSeqPM (PCA)	$\mathcal{O}\left(\frac{K}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{K}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{K^2}{\log^2 \text{gap}_r^{-1}} \log^2 \frac{1}{\epsilon}\right)$
S-DOT (PSA)	$\mathcal{O}\left(\frac{1}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\log^2 \text{gap}_r^{-1}} \log^2 \frac{1}{\epsilon}\right)$
DeEPCA (PSA)	$\mathcal{O}\left(\log \frac{1}{\text{gap}}\right)$	$\mathcal{O}\left(\frac{1}{\text{gap}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\text{gap}} \log \frac{1}{\text{gap}} \log \frac{1}{\epsilon}\right)$

$$\text{gap}_r = \frac{\lambda_{K+1}}{\lambda_K} \quad \text{for PSA}$$

$$\text{gap}_r = \max_{k=1, \dots, K} \frac{\lambda_{k+1}}{\lambda_k} \quad \text{for PCA}$$

$$\text{gap} = \lambda_K - \lambda_{K+1} \quad \text{for PSA}$$

$$(\lambda_1, \lambda_2, \dots, \lambda_d \text{ are the eigenvalues of } \mathbf{C} = \sum_i \mathbf{C}_i)$$

DistSeqPM (Distributed Sequential Power Method): Extension of DePM (Raja-Bajwa '16, Wai et al. '17)

S-DOT (Sample-wise Distributed Orthogonal Iterations): Gang-Xiang-Bajwa '21

DeEPCA (Decentralized Exact PCA): Ye-Zhang '21

Distributed PCA: The Road Ahead

Can we obtain solutions that provably converge to the true eigenvectors?

- We desire PCA solutions and not just the PSA solutions.

Can we reduce dependence of the complexity on the eigen gap and the final error?

- Can we reduce or do away with extra communication steps? Is there a faster solution?

We need a distributed PCA solution that is both fast and exact

Distributed PCA: The Road Ahead

Can we obtain solutions that provably converge to the true eigenvectors?

- We desire PCA solutions and not just the PSA solutions.

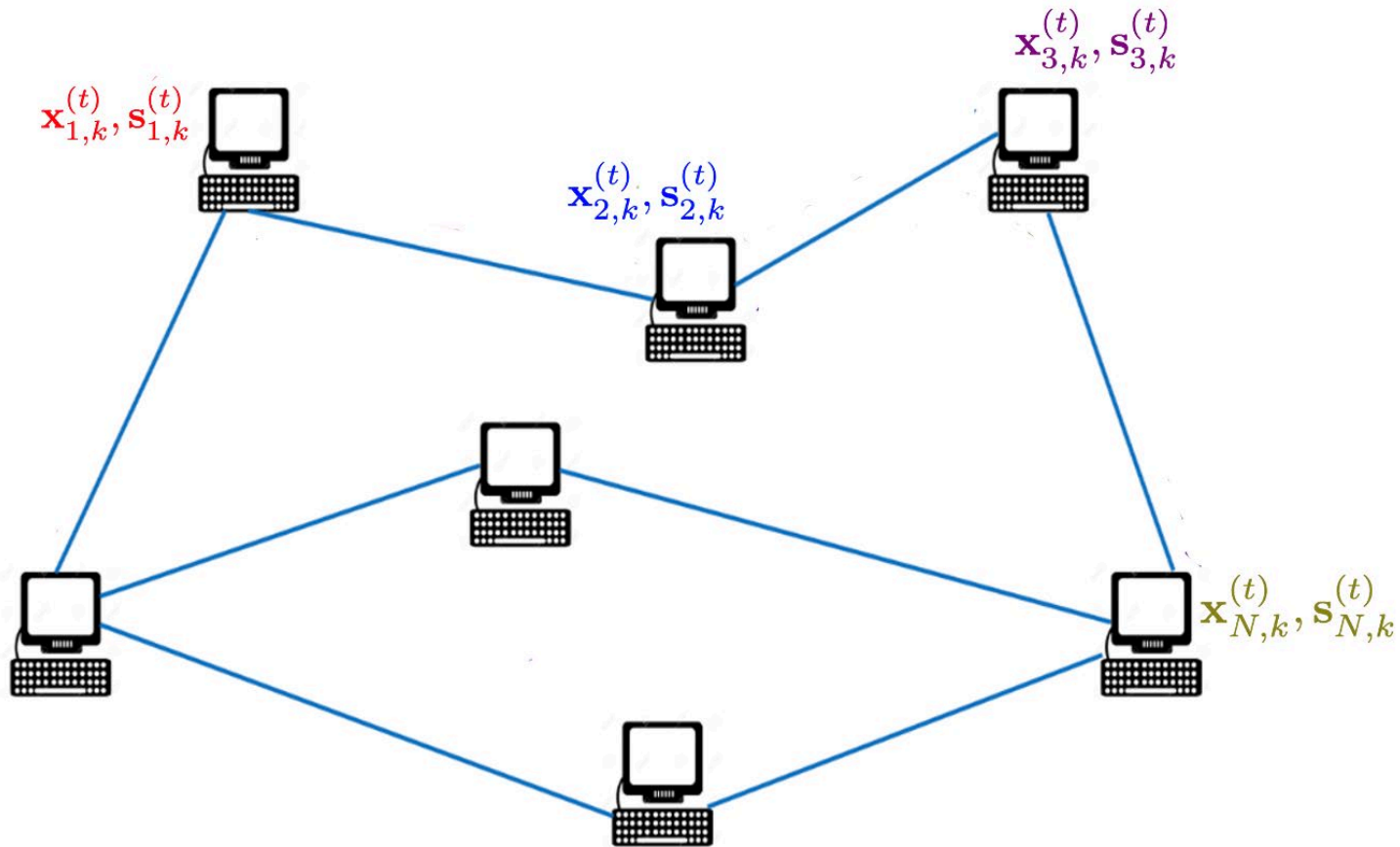
Can we reduce dependence of the complexity on the eigen gap and the final error?

- Can we reduce or do away with extra communication steps? Is there a faster solution?

We need a distributed PCA solution that is both fast and exact

Idea: A 'gradient tracking'-based solution

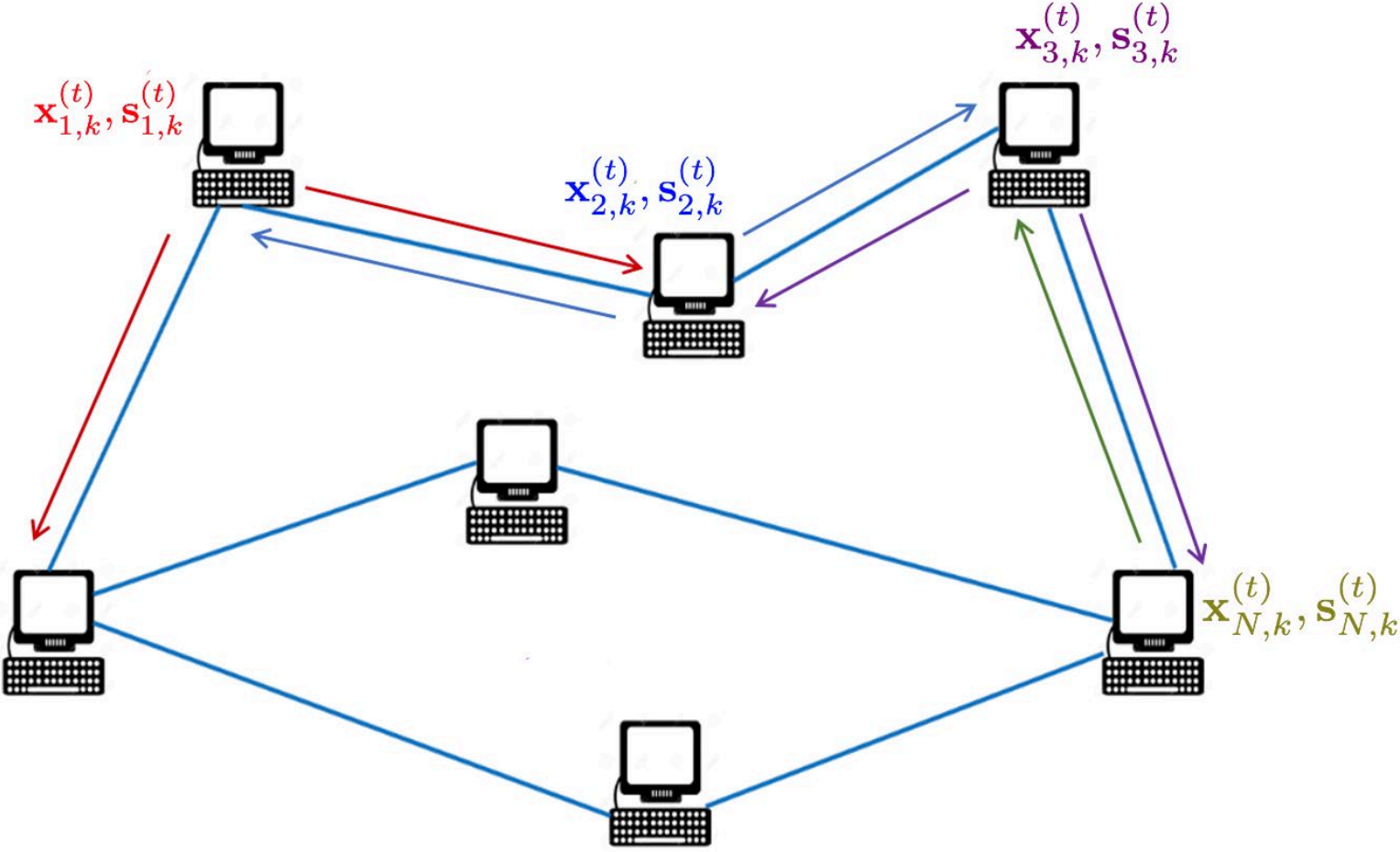
Solution: Fast and exAct diSTributed PCA (FAST-PCA)



$\mathbf{x}_{i,k}^{(t)}$: estimate of the k^{th} eigenvector at node i after t iterations.

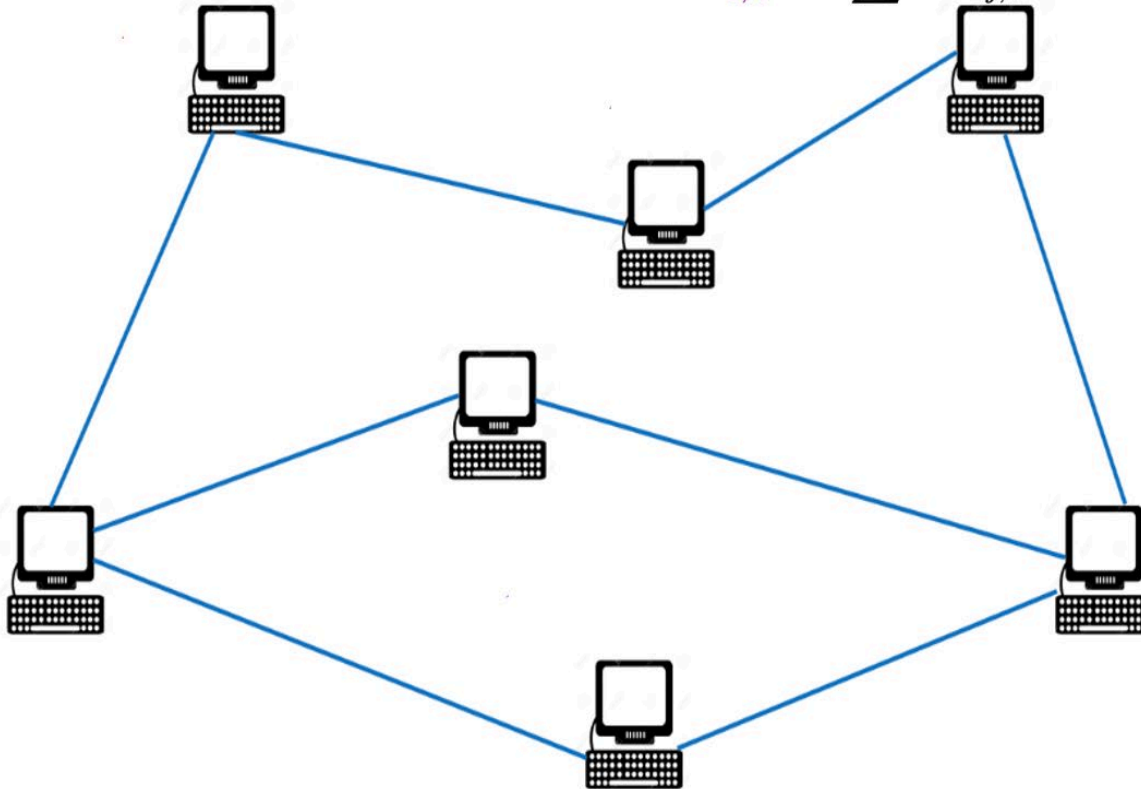
$\mathbf{s}_{i,k}^{(t)}$: estimate of the average of pseudo gradients $\mathbf{h}_i(\mathbf{x}_{i,k}^{(t)})$ at node i after t iterations.

FAST-PCA: Communication with Neighbors



FAST-PCA: Update equation

$$\mathbf{x}_{3,k}^{(t+1)} = \sum w_{3j} \mathbf{x}_{j,k}^{(t)} + \alpha \mathbf{s}_{3,k}^{(t)}$$
$$\mathbf{s}_{3,k}^{(t+1)} = \sum w_{3j} \mathbf{s}_{j,k}^{(t)} + \mathbf{h}_3(\mathbf{x}_{3,k}^{(t+1)}) - \mathbf{h}_3(\mathbf{x}_{3,k}^{(t)})$$



w_{ij} are averaging weights. Depend on the number of neighbors.

Assumptions

Underlying graph of N nodes is undirected and connected.

Weight matrix $\mathbf{W} = [w_{ij}]$ is doubly stochastic, hence

$$\beta = \max\{|\lambda_2(\mathbf{W})|, |\lambda_N(\mathbf{W})|\} < 1$$

The eigenvalues λ_i of covariance matrix \mathbf{C} has distinct first K eigenvalues, the following holds:

$$\lambda_1 > \lambda_2 \dots \lambda_K > \lambda_{K+1} \geq \dots \geq \lambda_d$$

FAST-PCA-O: First Variant

$$\mathbf{h}_i(\mathbf{x}_{i,k}^{(t)}) = \mathbf{C}_i \mathbf{x}_{i,k}^{(t)} - (\mathbf{x}_{i,k}^{(t)})^T \mathbf{C}_i \mathbf{x}_{i,k}^{(t)} \mathbf{x}_{i,k}^{(t)} - \sum_{p=1}^{k-1} (\mathbf{x}_{i,p}^{(t)})^T \mathbf{C}_i \mathbf{x}_{i,k}^{(t)} \mathbf{x}_{i,p}^{(t)}$$

Theorem

Suppose the estimate $\mathbf{x}_{i,p}^{(t)}$ from FAST-PCA-O remains bounded for $p = 1, \dots, k$, i.e., $\|\mathbf{x}_{i,p}^{(t)}\|^2 \leq \mu$, $\alpha \leq \frac{\min_{k=1, \dots, K} (\lambda_k - \lambda_{k+1})}{2\lambda_1^2 (1 + 2\mu + k\mu)^2} \left(\frac{1 - \beta}{9}\right)^2$ where λ_k, λ_{k+1} are the k^{th} and $(k+1)^{\text{th}}$ largest eigenvalues of \mathbf{C} and $\beta = \max\{|\lambda_2(\mathbf{W})|, |\lambda_N(\mathbf{W})|\}$, $\mathbf{q}_k^T \mathbf{x}_{i,k}^{(0)} \neq 0$, and the graph underlying the network is connected. Then the estimate $\mathbf{x}_{i,k}^{(t)}$ from FAST-PCA-O converges to the eigenvector $\pm \mathbf{q}_k$ corresponding to the largest eigenvalue λ_k of \mathbf{C} at each node $i = 1, \dots, N$ at a linear rate.

FAST-PCA-O: First Variant

$$\mathbf{h}_i(\mathbf{x}_{i,k}^{(t)}) = \underbrace{\mathbf{C}_i \mathbf{x}_{i,k}^{(t)} - (\mathbf{x}_{i,k}^{(t)})^T \mathbf{C}_i \mathbf{x}_{i,k}^{(t)} \mathbf{x}_{i,k}^{(t)}}_{\text{Based On Hebbian (Oja's) rule}} - \underbrace{\sum_{p=1}^{k-1} (\mathbf{x}_{i,p}^{(t)})^T \mathbf{C}_i \mathbf{x}_{i,k}^{(t)} \mathbf{x}_{i,p}^{(t)}}_{\text{Gram-Schmidt orthogonalization}}$$

Based On Hebbian (Oja's) rule

Gram-Schmidt orthogonalization

Theorem

Suppose the estimate $\mathbf{x}_{i,p}^{(t)}$ from FAST-PCA-O remains bounded for $p = 1, \dots, k$, i.e., $\|\mathbf{x}_{i,p}^{(t)}\|^2 \leq \mu$,

$\alpha \leq \frac{\min_{k=1, \dots, K} (\lambda_k - \lambda_{k+1})}{2\lambda_1^2 (1 + 2\mu + k\mu)^2} \left(\frac{1 - \beta}{9}\right)^2$ where λ_k, λ_{k+1} are the k^{th} and $(k+1)^{\text{th}}$ largest

eigenvalues of \mathbf{C} and $\beta = \max\{|\lambda_2(\mathbf{W})|, |\lambda_N(\mathbf{W})|\}$, $\mathbf{q}_k^T \mathbf{x}_{i,k}^{(0)} \neq 0$, and the graph underlying the network is connected. Then the estimate $\mathbf{x}_{i,k}^{(t)}$ from FAST-PCA-O converges to the eigenvector $\pm \mathbf{q}_k$ corresponding to the largest eigenvalue λ_k of \mathbf{C} at each node $i = 1, \dots, N$ at a linear rate.

FAST-PCA-O: First Variant

$$\mathbf{h}_i(\mathbf{x}_{i,k}^{(t)}) = \underbrace{\mathbf{C}_i \mathbf{x}_{i,k}^{(t)} - (\mathbf{x}_{i,k}^{(t)})^T \mathbf{C}_i \mathbf{x}_{i,k}^{(t)} \mathbf{x}_{i,k}^{(t)}}_{\text{Based On Hebbian (Oja's) rule}} - \underbrace{\sum_{p=1}^{k-1} (\mathbf{x}_{i,p}^{(t)})^T \mathbf{C}_i \mathbf{x}_{i,k}^{(t)} \mathbf{x}_{i,p}^{(t)}}_{\text{Gram-Schmidt orthogonalization}}$$

Based On Hebbian (Oja's) rule

Gram-Schmidt orthogonalization

Theorem

Suppose the estimate $\mathbf{x}_{i,p}^{(t)}$ from FAST-PCA-O remains bounded for $p = 1, \dots, k$, i.e., $\|\mathbf{x}_{i,p}^{(t)}\|^2 \leq \mu$,

$\alpha \leq \frac{\min_{k=1, \dots, K} (\lambda_k - \lambda_{k+1})}{2\lambda_1^2 (1 + 2\mu + k\mu)^2} \left(\frac{1 - \beta}{9}\right)^2$ where λ_k, λ_{k+1} are the k^{th} and $(k+1)^{\text{th}}$ largest

eigenvalues of \mathbf{C} and $\beta = \max\{|\lambda_2(\mathbf{W})|, |\lambda_N(\mathbf{W})|\}$, $\mathbf{q}_k^T \mathbf{x}_{i,k}^{(0)} \neq 0$, and the graph underlying the network is connected. Then the estimate $\mathbf{x}_{i,k}^{(t)}$ from FAST-PCA-O converges to the eigenvector $\pm \mathbf{q}_k$ corresponding to the largest eigenvalue λ_k of \mathbf{C} at each node $i = 1, \dots, N$ at a linear rate.

FAST-PCA-K: Second Variant

$$\mathbf{h}_i(\mathbf{x}_{i,k}^{(t)}) = \mathbf{C}_i \mathbf{x}_{i,k}^{(t)} - \frac{(\mathbf{x}_{i,k}^{(t)})^T \mathbf{C}_i \mathbf{x}_{i,k}^{(t)}}{\|\mathbf{x}_{i,k}^{(t)}\|^2} \mathbf{x}_{i,k}^{(t)} - \sum_{p=1}^{k-1} \frac{(\mathbf{x}_{i,p}^{(t)})^T \mathbf{C}_i \mathbf{x}_{i,k}^{(t)}}{\|\mathbf{x}_{i,p}^{(t)}\|^2} \mathbf{x}_{i,p}^{(t)}$$

Theorem

Suppose $\alpha \leq \frac{\min_{k=1,\dots,K}(\lambda_k - \lambda_{k+1})}{2\lambda_1^2(K+5)^2} \left(\frac{1-\beta}{9}\right)^2$ where λ_k, λ_{k+1} are the k^{th} and $(k+1)^{\text{th}}$ largest eigenvalues of \mathbf{C} , $\beta = \max\{|\lambda_2(\mathbf{W})|, |\lambda_N(\mathbf{W})|\}$ and $\mathbf{q}_k^T \mathbf{x}_{i,k}^{(0)} \neq 0$ and the graph underlying the network is connected, then the estimate $\mathbf{x}_{i,k}^{(t)}$ from FAST-PCA-K converges to a multiple of the eigenvector $\pm c_k \mathbf{q}_k$ corresponding to the largest eigenvalue λ_k of \mathbf{C} at each node $i = 1, \dots, N$ at a linear rate.

FAST-PCA-K: Second Variant

$$\mathbf{h}_i(\mathbf{x}_{i,k}^{(t)}) = \underbrace{\mathbf{C}_i \mathbf{x}_{i,k}^{(t)} - \frac{(\mathbf{x}_{i,k}^{(t)})^T \mathbf{C}_i \mathbf{x}_{i,k}^{(t)}}{\|\mathbf{x}_{i,k}^{(t)}\|^2} \mathbf{x}_{i,k}^{(t)}}_{\text{Based On Krasulina's rule}} - \underbrace{\sum_{p=1}^{k-1} \frac{(\mathbf{x}_{i,p}^{(t)})^T \mathbf{C}_i \mathbf{x}_{i,k}^{(t)}}{\|\mathbf{x}_{i,p}^{(t)}\|^2} \mathbf{x}_{i,p}^{(t)}}_{\text{Gram-Schmidt orthogonalization}}$$

Based On Krasulina's rule

Gram-Schmidt orthogonalization

Theorem

Suppose $\alpha \leq \frac{\min_{k=1,\dots,K}(\lambda_k - \lambda_{k+1})}{2\lambda_1^2(K+5)^2} \left(\frac{1-\beta}{9}\right)^2$ where λ_k, λ_{k+1} are the k^{th} and $(k+1)^{\text{th}}$ largest eigenvalues of \mathbf{C} , $\beta = \max\{|\lambda_2(\mathbf{W})|, |\lambda_N(\mathbf{W})|\}$ and $\mathbf{q}_k^T \mathbf{x}_{i,k}^{(0)} \neq 0$ and the graph underlying the network is connected, then the estimate $\mathbf{x}_{i,k}^{(t)}$ from FAST-PCA-K converges to a multiple of the eigenvector $\pm c_k \mathbf{q}_k$ corresponding to the largest eigenvalue λ_k of \mathbf{C} at each node $i = 1, \dots, N$ at a linear rate.

FAST-PCA-K: Second Variant

$$\mathbf{h}_i(\mathbf{x}_{i,k}^{(t)}) = \underbrace{\mathbf{C}_i \mathbf{x}_{i,k}^{(t)} - \frac{(\mathbf{x}_{i,k}^{(t)})^T \mathbf{C}_i \mathbf{x}_{i,k}^{(t)}}{\|\mathbf{x}_{i,k}^{(t)}\|^2} \mathbf{x}_{i,k}^{(t)}}_{\text{Based On Krasulina's rule}} - \underbrace{\sum_{p=1}^{k-1} \frac{(\mathbf{x}_{i,p}^{(t)})^T \mathbf{C}_i \mathbf{x}_{i,k}^{(t)}}{\|\mathbf{x}_{i,p}^{(t)}\|^2} \mathbf{x}_{i,p}^{(t)}}_{\text{Gram-Schmidt orthogonalization}}$$

Based On Krasulina's rule

Gram-Schmidt orthogonalization

Theorem

Suppose $\alpha \leq \frac{\min_{k=1,\dots,K} (\lambda_k - \lambda_{k+1})}{2\lambda_1^2 (K+5)^2} \left(\frac{1-\beta}{9}\right)^2$ where λ_k, λ_{k+1} are the k^{th} and $(k+1)^{\text{th}}$ largest eigenvalues of \mathbf{C} , $\beta = \max\{|\lambda_2(\mathbf{W})|, |\lambda_N(\mathbf{W})|\}$ and $\mathbf{q}_k^T \mathbf{x}_{i,k}^{(0)} \neq 0$ and the graph underlying the network is connected, then the estimate $\mathbf{x}_{i,k}^{(t)}$ from FAST-PCA-K converges to a multiple of the eigenvector $\pm c_k \mathbf{q}_k$ corresponding to the largest eigenvalue λ_k of \mathbf{C} at each node $i = 1, \dots, N$ at a linear rate.

Communication and Iteration Complexity

	Comm./iteration	Iterations	Total Comm.
DistSeqPM (PCA)	$\mathcal{O}\left(\frac{K}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{K}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{K^2}{\log^2 \text{gap}_r^{-1}} \log^2 \frac{1}{\epsilon}\right)$
S-DOT (PSA)	$\mathcal{O}\left(\frac{1}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\log^2 \text{gap}_r^{-1}} \log^2 \frac{1}{\epsilon}\right)$
DeEPCA (PSA)	$\mathcal{O}\left(\log \frac{1}{\text{gap}}\right)$	$\mathcal{O}\left(\frac{1}{\text{gap}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\text{gap}} \log \frac{1}{\text{gap}} \log \frac{1}{\epsilon}\right)$
DSA (PCA)	1	$\mathcal{O}\left(\frac{1}{\log(1 + \alpha \text{gap})} \log \frac{1}{\epsilon}\right)$ (upto $\epsilon = \mathcal{O}(\alpha)$)	$\mathcal{O}\left(\frac{1}{\log(1 + \alpha \text{gap})} \log \frac{1}{\epsilon}\right)$
FAST-PCA (PCA)	1	$\mathcal{O}\left(\frac{1}{\log(1 + \alpha \text{gap})} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\log(1 + \alpha \text{gap})} \log \frac{1}{\epsilon}\right)$

$$\text{gap}_r = \frac{\lambda_{K+1}}{\lambda_K} \quad \text{for PSA}$$

$$\text{gap}_r = \max_{k=1, \dots, K} \frac{\lambda_{k+1}}{\lambda_k} \quad \text{for PCA}$$

$$\text{gap} = \lambda_K - \lambda_{K+1} \quad \text{for PSA}$$

$$\text{gap} = \min_{k=1, \dots, K} (\lambda_k - \lambda_{k+1}) \quad \text{for PCA}$$

DistSeqPM (Distributed Sequential Power Method): Extension of DePM (Raja-Bajwa '16, Wai et al. '17)

S-DOT (Sample-wise Distributed Orthogonal Iterations): Gang-Xiang-Bajwa '21

DeEPCA (Decentralized Exact PCA): Ye-Zhang '21

DSA (Distributed Sanger's Algorithm): Gang-Bajwa '21

FAST-PCA: Gang-Bajwa '21, '22 (submitted; EUSIPCO)

Communication and Iteration Complexity

	Comm./iteration	Iterations	Total Comm.
DistSeqPM (PCA)	$\mathcal{O}\left(\frac{K}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{K}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{K^2}{\log^2 \text{gap}_r^{-1}} \log^2 \frac{1}{\epsilon}\right)$
S-DOT (PSA)	$\mathcal{O}\left(\frac{1}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\log \text{gap}_r^{-1}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\log^2 \text{gap}_r^{-1}} \log^2 \frac{1}{\epsilon}\right)$
DeEPCA (PSA)	$\mathcal{O}\left(\log \frac{1}{\text{gap}}\right)$	$\mathcal{O}\left(\frac{1}{\text{gap}} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\text{gap}} \log \frac{1}{\text{gap}} \log \frac{1}{\epsilon}\right)$
DSA (PCA)	1	$\mathcal{O}\left(\frac{1}{\log(1 + \alpha \text{gap})} \log \frac{1}{\epsilon}\right)$ (upto $\epsilon = \mathcal{O}(\alpha)$)	$\mathcal{O}\left(\frac{1}{\log(1 + \alpha \text{gap})} \log \frac{1}{\epsilon}\right)$
FAST-PCA (PCA)	1	$\mathcal{O}\left(\frac{1}{\log(1 + \alpha \text{gap})} \log \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\log(1 + \alpha \text{gap})} \log \frac{1}{\epsilon}\right)$

$$\text{gap}_r = \frac{\lambda_{K+1}}{\lambda_K} \quad \text{for PSA}$$

$$\text{gap}_r = \max_{k=1, \dots, K} \frac{\lambda_{k+1}}{\lambda_k} \quad \text{for PCA}$$

$$\text{gap} = \lambda_K - \lambda_{K+1} \quad \text{for PSA}$$

$$\text{gap} = \min_{k=1, \dots, K} (\lambda_k - \lambda_{k+1}) \quad \text{for PCA}$$

DistSeqPM (Distributed Sequential Power Method): Extension of DePM (Raja-Bajwa '16, Wai et al. '17)

S-DOT (Sample-wise Distributed Orthogonal Iterations): Gang-Xiang-Bajwa '21

DeEPCA (Decentralized Exact PCA): Ye-Zhang '21

DSA (Distributed Sanger's Algorithm): Gang-Bajwa '21

FAST-PCA: Gang-Bajwa '21, '22 (submitted; EUSIPCO)

Simulations: Performance Metric and Setup

We use the angles between our estimates $\mathbf{x}_{i,k}^{(t)}$ and true eigenvectors \mathbf{q}_k of the sample covariance matrix \mathbf{C} as the performance metric.

Average error for estimating K eigenvectors at N nodes after t iterations:

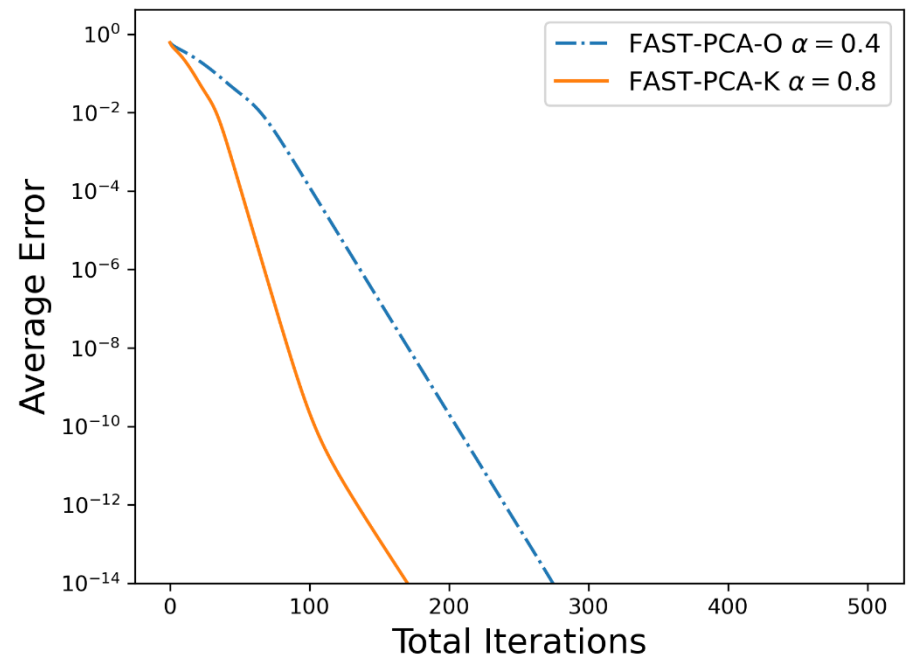
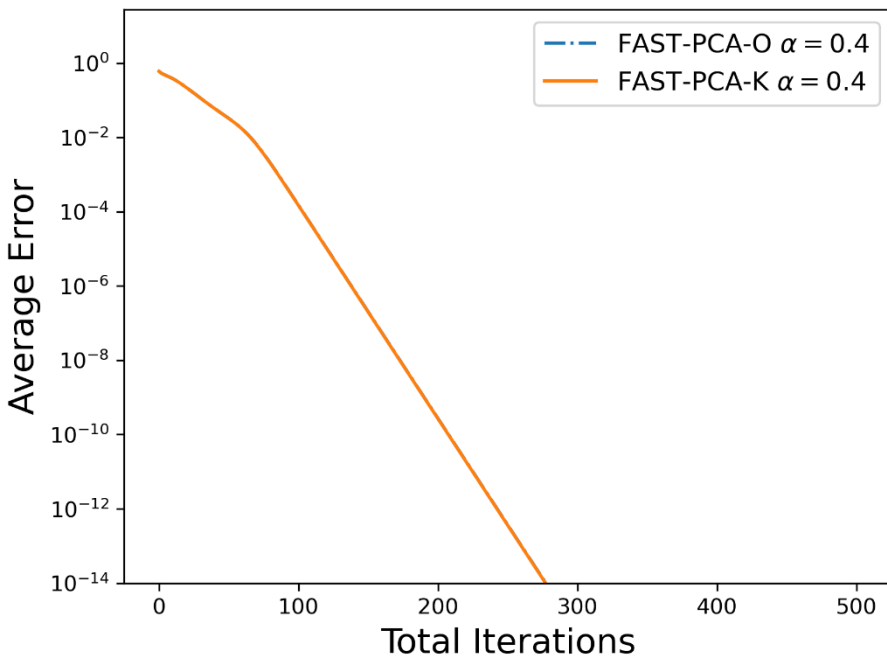
$$\mathcal{E} = \frac{1}{NK} \sum_{i=1}^N \sum_{k=1}^K \sin^2(\mathbf{q}_k^T \mathbf{x}_{i,k}^{(t)})$$

Setup:

- Generated an Erdos-Renyi graph of N nodes with connectivity prob. $p = 0.5$.
- Samples are generated identically and independently from a zero mean Gaussian distribution.
- Samples are equally divided among the nodes.
- 10 Monte-Carlo trials for each synthetic data experiments.

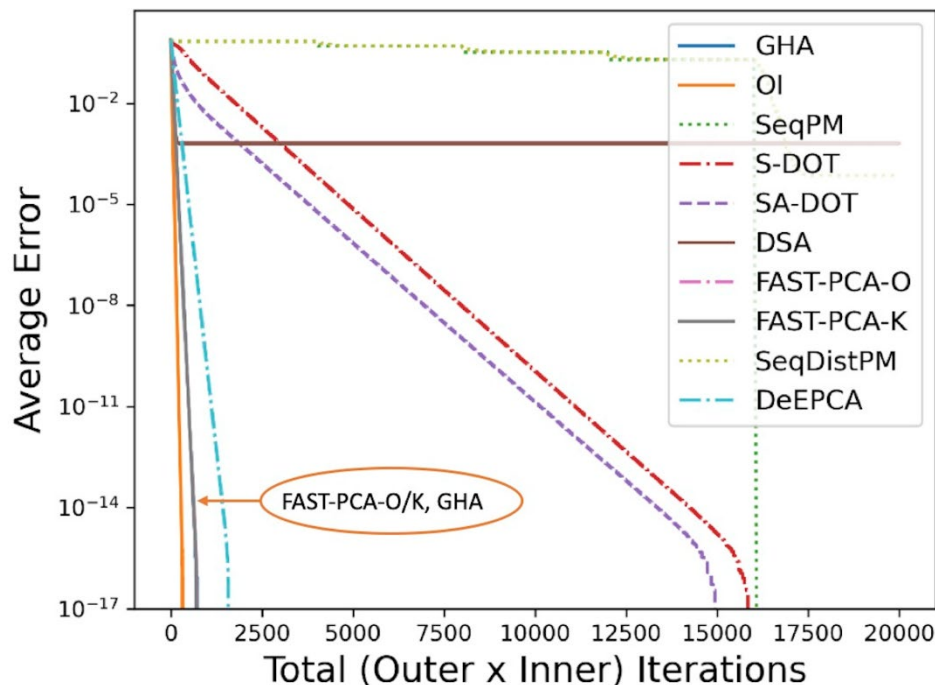
FAST-PCA-O vs FAST-PCA-K

$d = 20, N = 20, n = 200000, K = 2, gap = 0.15$

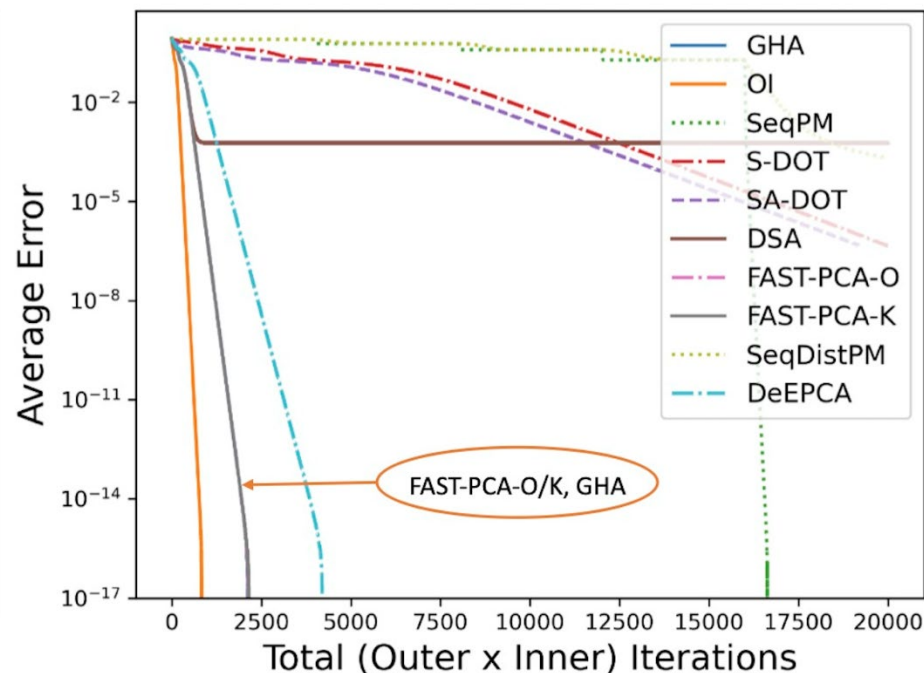


Synthetic Data: Performance Comparison

$$d = 20, K = 5, N = 20, n = 100000$$



$\text{gap}_r = 0.8$



$\text{gap}_r = 0.97,$

GHA (Generalized Hebbian Algorithm): Sanger'82

OI (Orthogonal Iteration): Golub'83

SeqPM (Sequential Power Method): Ext. of Golub'83

SeqDistPM (Sequential Distributed Power Method): Ext. of Raja-Bajwa '16, Wai et al. '17

S-DOT, SA-DOT (Sample-wise Distributed Orthogonal Iterations): Gang-Xiang-Bajwa '21

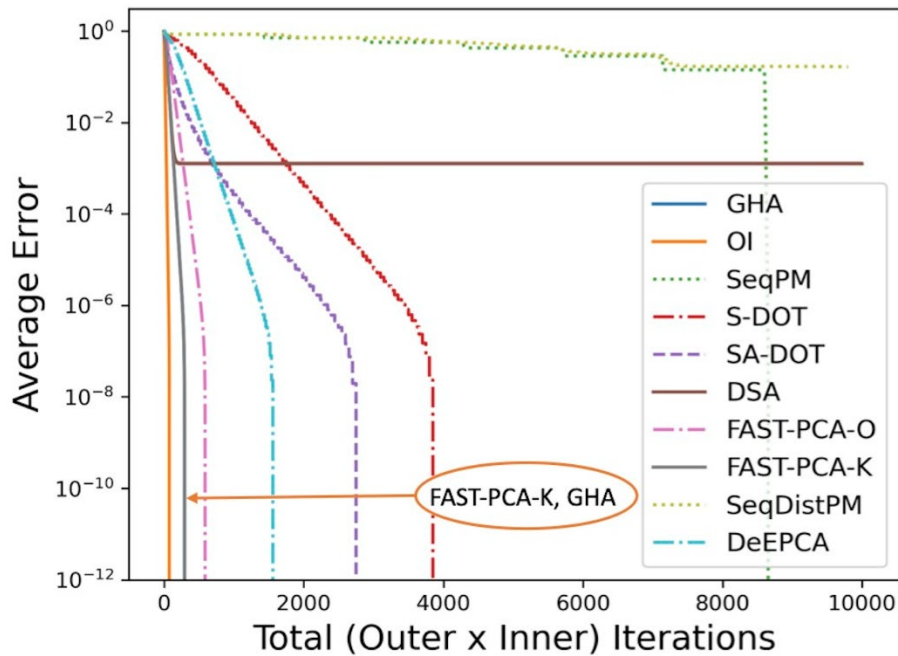
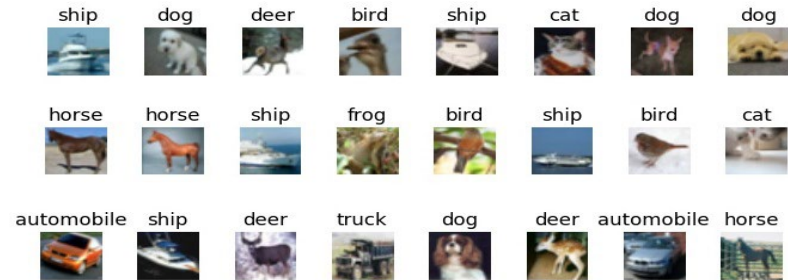
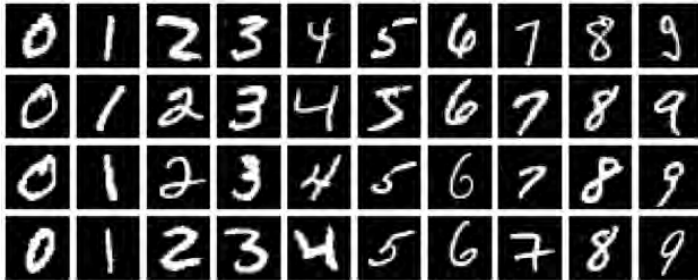
DeEPCA (Decentralized Exact PCA): Ye-Zhang '21

DSA (Distributed Sanger's Algorithm): Gang-Bajwa '21

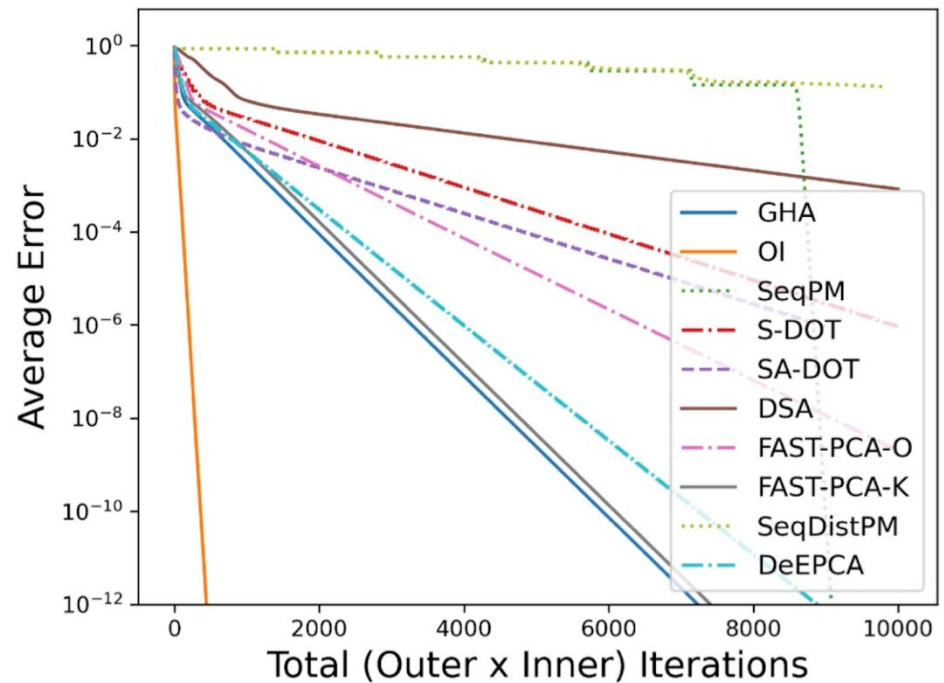
FAST-PCA-O/K: Gang-Bajwa '22 (submitted)

Real Data

$K = 7, N = 20, n = 60000$ (evenly distributed)



MNIST



CIFAR10

Conclusion and Open Questions

This talk dealt with the distributed PCA problem when the data samples are split across an arbitrary network

- **Distributed Batch Setting:** Two algorithms based on the neural network formulation were developed and analyzed
 - The algorithms linearly converge to the true eigenvectors of the sample covariance
 - The iteration / communications complexity of the algorithms only depends logarithmically on the eigen gap
 - The Krasulina variant of the algorithm allows for a larger step size, compared to the Oja variant, which can lead to faster convergence in some settings

Open Questions

- Is the dependence on the eigen gap optimal, or can it be improved?
- How do these algorithms behave in the case of time-varying networks, directed networks, and/or asynchronous networks?
- What kind of algorithms can be developed in the case of both feature- and sample-wise partitioning of data in the case of massively large datasets?
- And many more ...