Streaming PCA For Markovian Data

Syamantak Kumar and Purnamrita Sarkar

UT Austin

PCA as an optimization problem

- X_1 ,....., X_n are IID mean zero random vectors in d dimensions with covariance matrix Σ.
  - *Σ has eigenvectors v_1 ,...., v_d and eigenvalues* $\lambda_1 > \lambda_2 \geq \ldots \geq \lambda_d$

- GOAL in PCA : estimate top eigenvector of Σ.

- We are optimizing $\widehat{w} := \underset{\|w\|_2=1}{\arg\max} \sum_{i=1}^{n} \left(X_i^T w\right)^2$

- Convergence measured in terms of the *sin*^{2} error - $1 - (\widehat{w}^T v_1)^2$

A Stochastic Gradient Descent (SGD) type algorithm[1]

learning rate - $\eta_t \propto 1/t$

$$w_{t+1} \leftarrow w_t + \eta_t \underbrace{(X_{t+1}^T w_t) X_{t+1}}$$

weight vector

Gradient computed on t+1th datapoint

$$w_{t+1} \leftarrow w_{t+1} / ||w_{t+1}||$$

No need to compute a d x d covariance matrix

Projection to the unit sphere

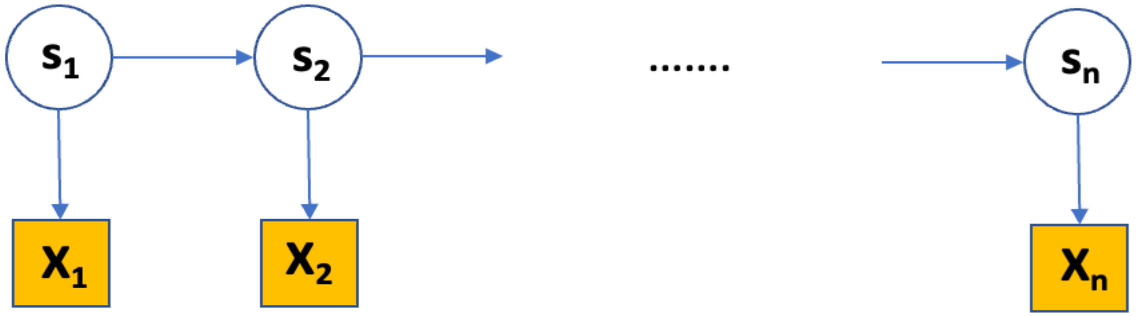
1. Oja, E. 1982

Our goal

- Existing results hold for the IID case
- Many real world scenarios have data coming from a Markov chain
- GOAL – Obtain similar results in the Markovian case

Setup

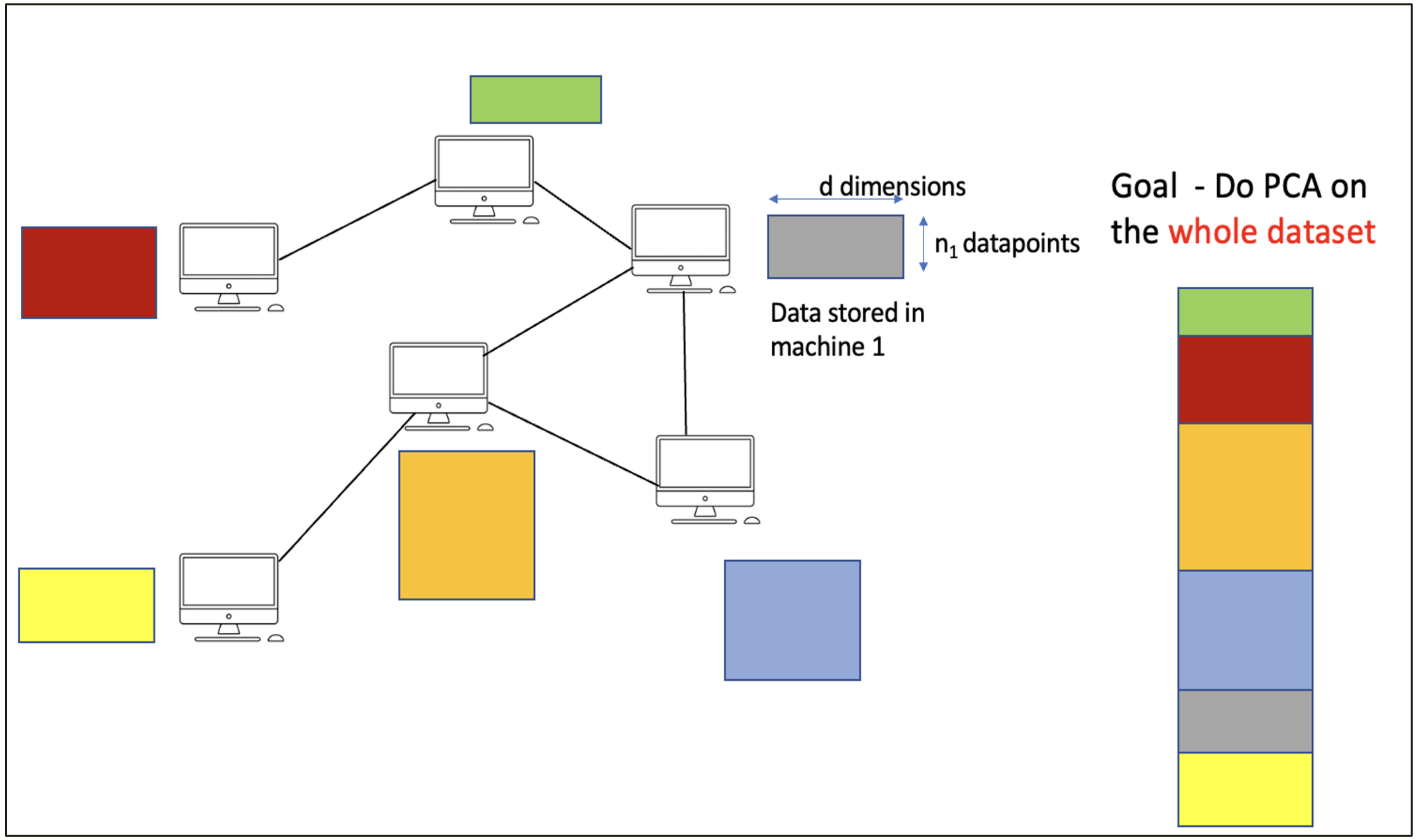
- Consider an irreducible, aperiodic and reversible* finite state space Markov Chain
  - *stationary distribution* π
  - *Transition matrix P*
  - *Second largest eigenvalue of P in magnitude* $\lambda_2(P)$



  - *Mean zero* $E_\pi[X_i] = 0$
  - ***GOAL: estimate top eigenvector of*** $E_\pi[X_i X_i^T]$

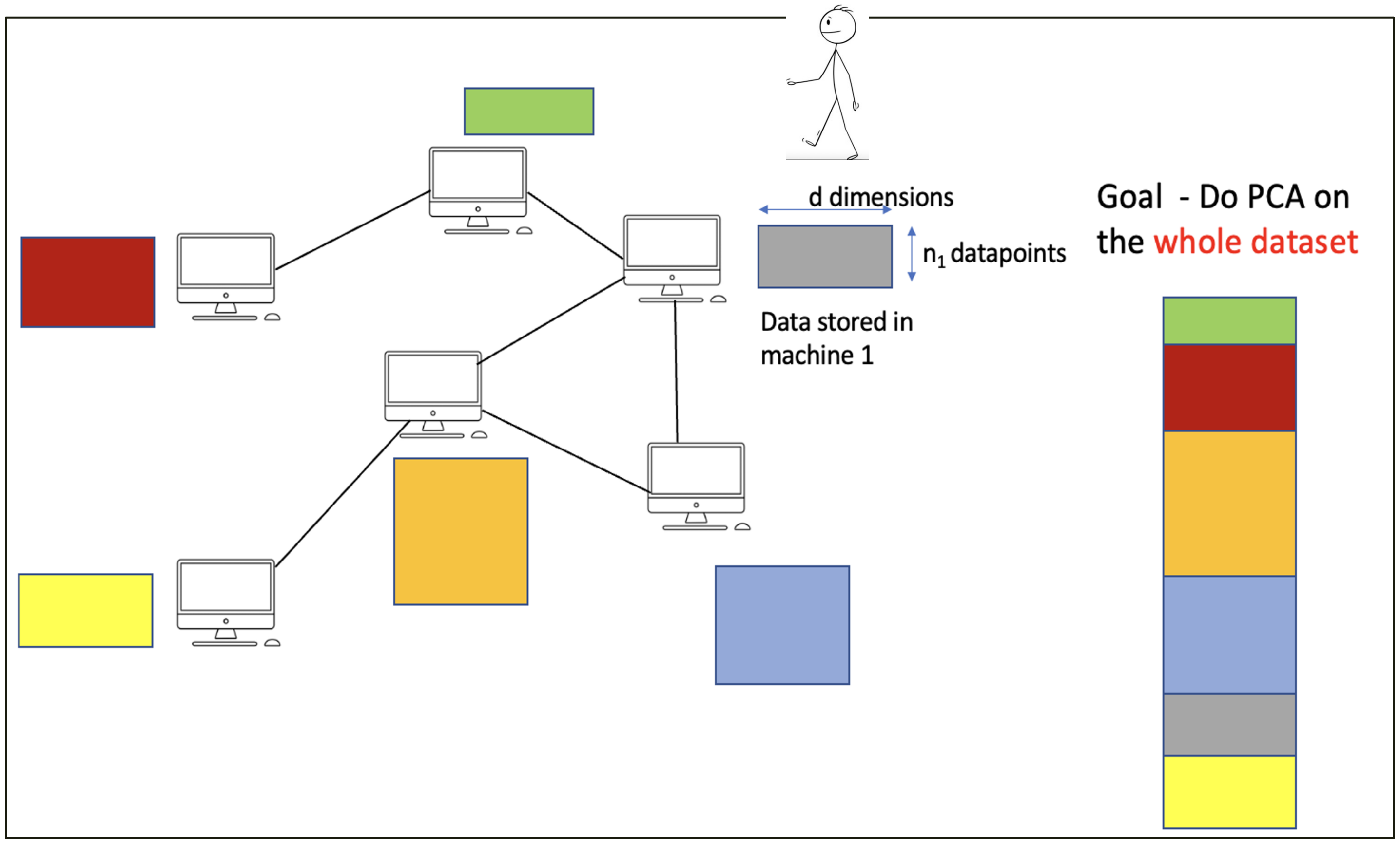
** This can be relaxed.*

Motivation

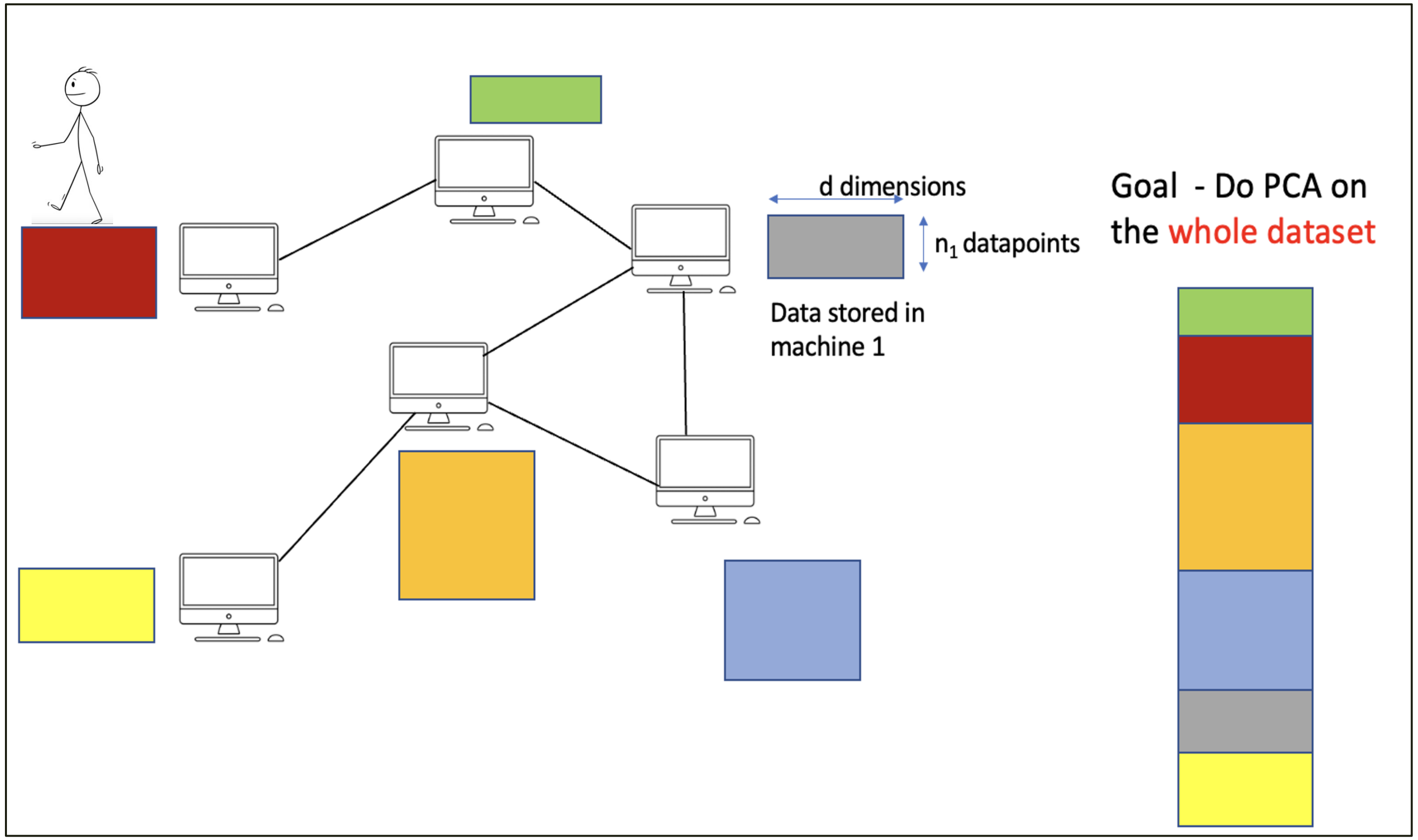


- Consider the classical federated learning setup with data distributed in a network of machines
- Aim : Decentralized algorithm for PCA on the global dataset
- Token Algorithms -
  - Construct Markov chain
  - Random walk with update at each timestep

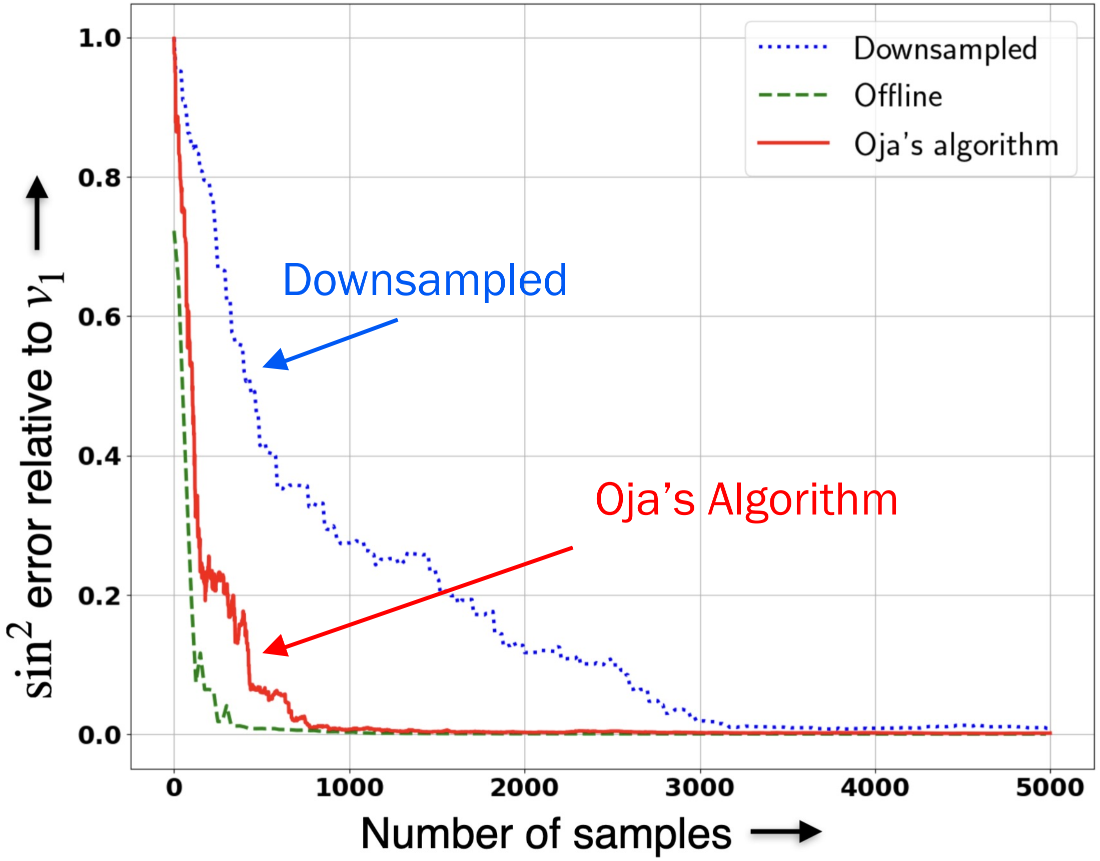
Streaming PCA on Markovian Data



Streaming PCA on Markovian Data



Downsample?



Downsampled Oja [Chen et al, 2018] performs **significantly worse** than Oja's algorithm on the whole data

Our contribution

	Offline	Online
IID	$O\left(\frac{\mathcal{V}\log(d)}{(\lambda_1-\lambda_2)^2 n}\right)$ Matrix Bernstein Jain et al. (2016)	$O\left(\frac{\mathcal{V}}{(\lambda_1-\lambda_2)^2 n}\right)$ Oja's algorithm Jain et al. (2016)
Markovian	$O\left(\frac{\mathcal{V}\log(d)}{(\lambda_1-\lambda_2)^2 n(1-\lambda_2(P))}\right)$ Neeman et al. (2023)	$O\left(\frac{\mathcal{V}}{(\lambda_1-\lambda_2)^2 n(1-\lambda_2(P))}\right)$ Our work

Table: $\sin^2$ error rate

Thank you!

Wednesday, Dec 13, 10:45AM

Poster Session 3