Distributed PCA and *k*-Means Clustering

Yingyu Liang, Maria-Florina Balcan, Vandana Kanchanapally

Presentation by Will Richardson

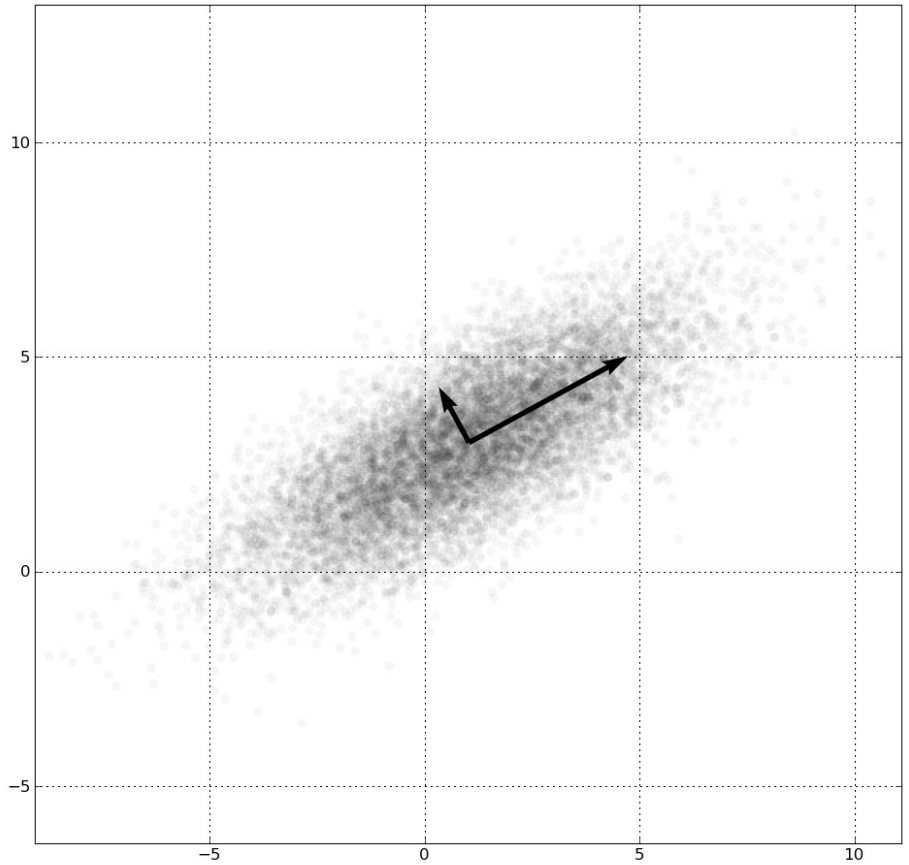
Organization

- What is PCA?
- Motivation
- Algorithm
- Theoretical Performance
- Empirical Performance
- Thoughts
- Questions

What is PCA?

- PCA: Principal Component Analysis
- How do we deal with high-dimensional data?
- Are all dimensions equally informative?
- How do we figure out which ones are significant?
- Hint: variance

PCA Visualized

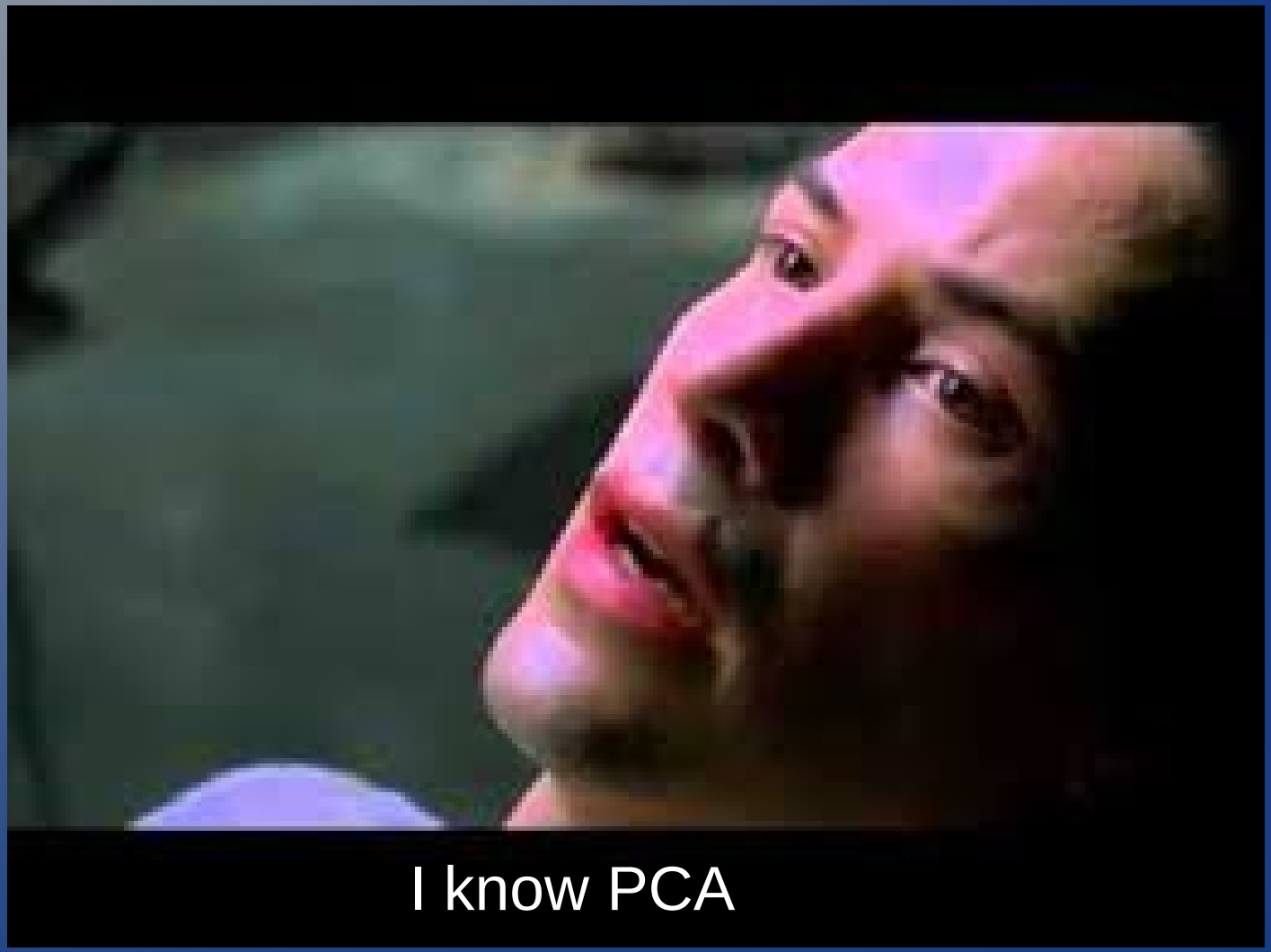


"**GaussianScatterPCA**" by — Ben FrantzDale (talk) (Transferred by ILCyborg) - PNG version of gigantic SVGOwn work (Original caption: “I created this work entirely by myself. (Originally uploaded on en.wikipedia) -”). Licensed under CC BY-SA 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:GaussianScatterPCA.png#mediaviewer/File:GaussianScatterPCA.png

How Do I PCA?

- Linear Algebra Review
- Eigenvalues and Eigenvectors
- SVD: Singular Value Decomposition

Got it?



Another core concept

- This paper is an extension of another:
- *Distributed k-Means and k-Median Clustering on General Topologies*
- Finding coresets in a distributed way
- What's a coreset?

Distributed PCA

- Suppose we have too much data to hold in main memory
- How can we find its principal components?
- How can we minimize noisy traffic over a distributed system?
- Computation nodes $\mathbf{V = \{ v_1, v_2, \ldots v_n\}}$
- Data set: $\mathbf{P = \{P_1, P_2, \ldots P_n\}}$

Round 1: Local PCA

- Perform SVD:
  - $\mathbf{P}_i = \mathbf{U}_i\mathbf{D}_i(\mathbf{E}_i)^T$
- Take the first *t* diagonal entries of Di. Put them in a matrix $\mathbf{D}_i^{(t)}$ with all other values set to zero
  - What *is* $\mathbf{D}_i^{(t)}$?
- Calculate $\mathbf{P}_i^{(t)} = \mathbf{U}_i\mathbf{D}_i^{(t)}(\mathbf{E}_i)^T$
  - What is $\mathbf{P}_i^{(t)}$?
- Broadcast $\mathbf{D}_i^{(t)}$ and $\mathbf{E}_i^{(t)}$ across V

Round 2: Global PCA

- Compute $\mathbf{S}_i^{(t)} = (\mathbf{P}_i^{(t)})^T \mathbf{P}_i^{(t)}$
- Calculate the sum over $\mathbf{S}_i^{(t)}$: $\mathbf{S}^{(t)}$
- Compute the eigenvectors for $\mathbf{S}^{(t)} = \mathbf{E}\Lambda\mathbf{E}^T$
- Take the first *t* columns of **E** : $\mathbf{E}^{(t)}$
- Project $\mathbf{P}_i^{(t)}$ onto $\mathbf{E}^{(T)}$: $\mathbf{P}_i^{(t)} \mathbf{E}^{(t)} (\mathbf{E}^{(t)})^T$
- Concatenate all $\mathbf{P}_i^{(t)}$

Distributed Clustering

- The final result differs from the original data:
  - We discarded less informative (translated) dimensions, as per usual with PCA
  - Distributed PCA is an approximation of the centralized algorithm
- How do we ensure that the translated data still resembles the original data points?

Distributed *k*-Means Clustering

- **Theorem 1** implies that the [squared] distances to a lower-dimension sub-space are approximatelly equal when there are enough components taken with respect to that space
- **Theorem 2** places bounds on how much the new data can differ from the old in terms of its relation to arbitrary centers
- **Theorem 3** places bounds on the amount of communication over the network (it's independent of data set size/dimensionality)

Experiments

- Having put theoretical bounds on the performance, how well does distributed PCA actually work?

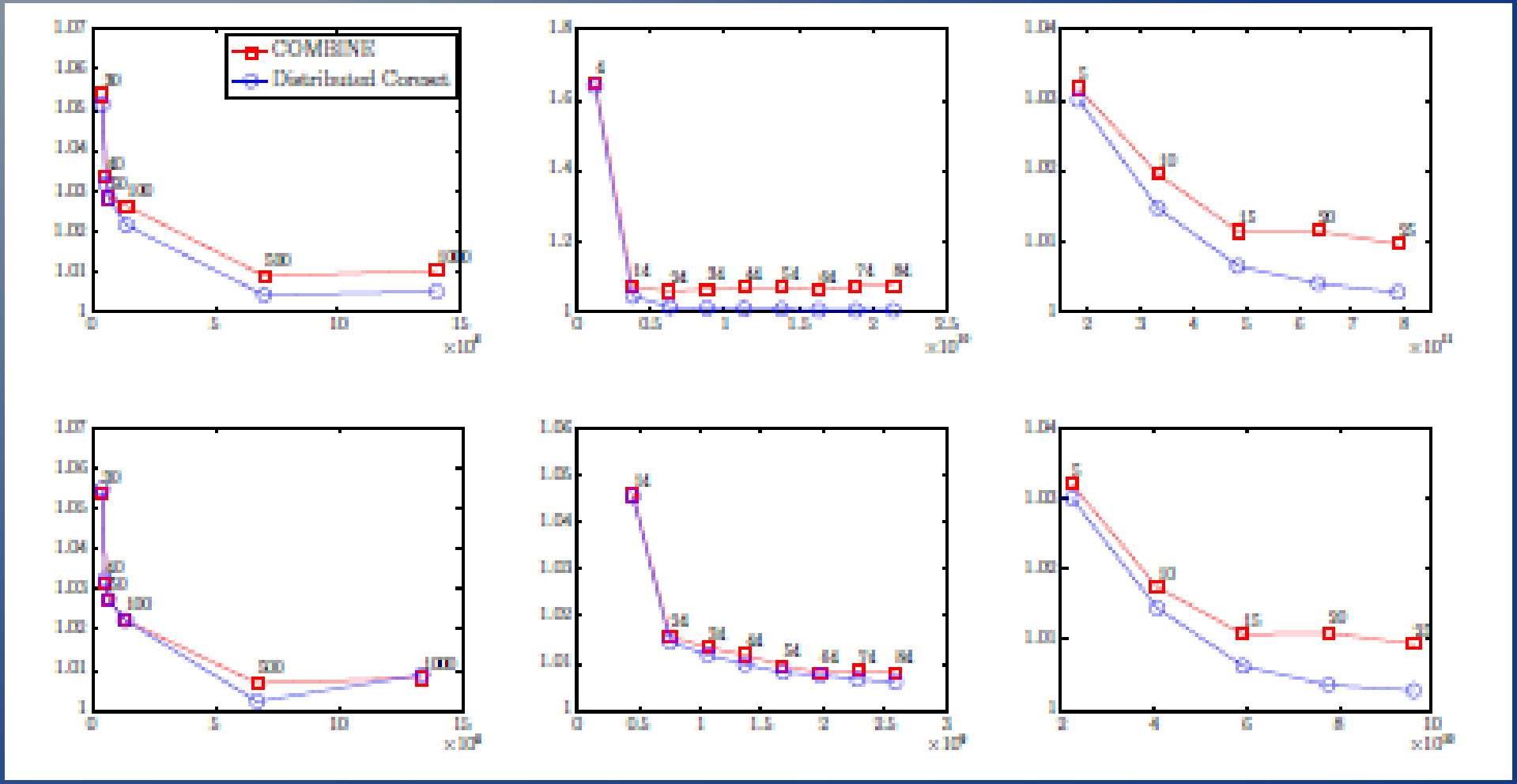
Data Sets

Data Set	Number of Points	Dimensionality
Daily and Sports Activities	9,210	5,625
MNIST handwritten digits	70,000	784
NYTimes Bag of Words	300,000	102,660

Experimental Setup

1) Generate a communication graph linking the computation nodes (square grid pattern)
2) Construct a set of clusters using either a distributed coreset algorithm (earlier paper) or COMBINE on each projection dimensionality
3) Run Lloyd's algorithm with the centers/coresets to partition the projected data
4) Run Lloyd's algorithm on the original data
5) Compare the results of 3 and 4

Results



Thoughts

- This paper has wide applicability
- By necessity, this paper is dense, but it remains reasonably accessible
- The experiments fail to demonstrate any comparative advantage of the PCA component

Questions