Low-Rank Matrices

(aka Spectral Methods)

Sujay Sanghavi

ECE

University of Texas, Austin

Outline

1. Background
   - Low-rank Matrices, Eigenvalues, eigenvectors, etc.

2. Principal Components Analysis

3. Spectral Clustering

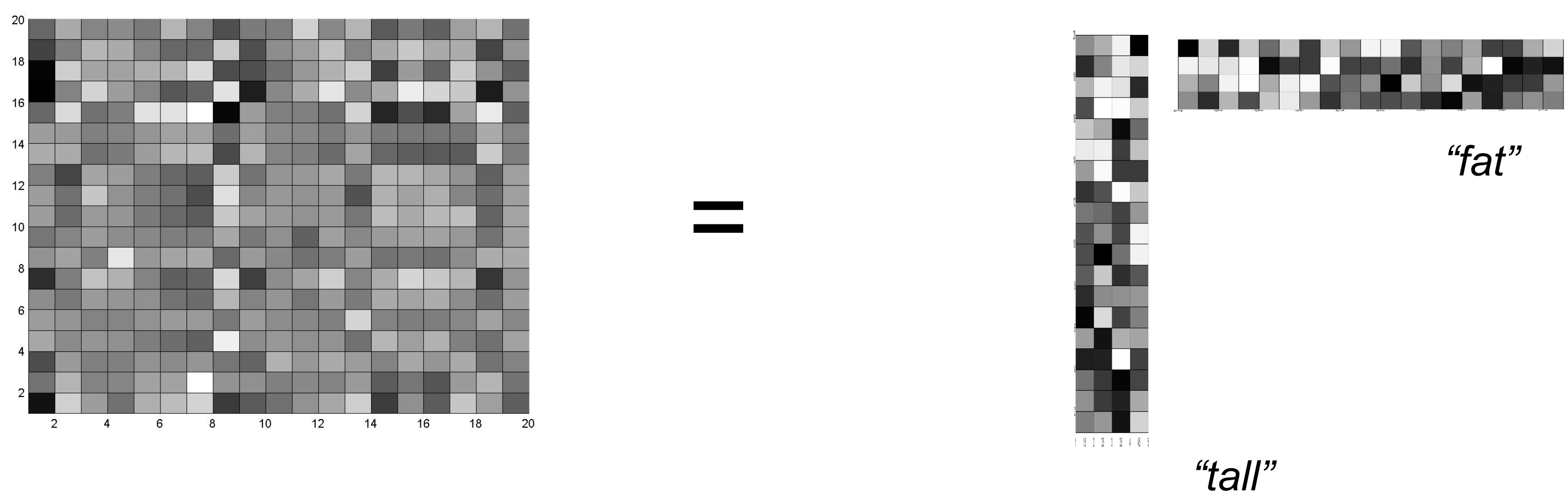
4. Predictions / collaborative filtering

5. Structured low-rank matrices: NNMF, Sparse PCA etc.

Low-Rank Matrices

Rank of a matrix = how many linearly independent rows (or columns) it has

A rank-r matrix can be written as product of smaller matrices:

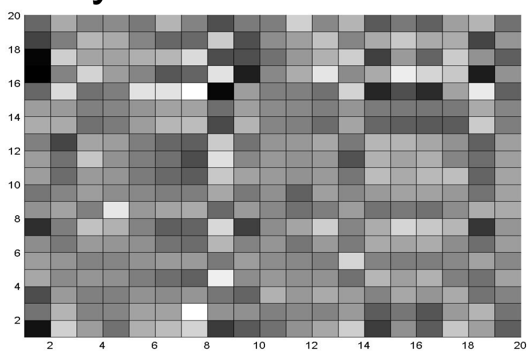


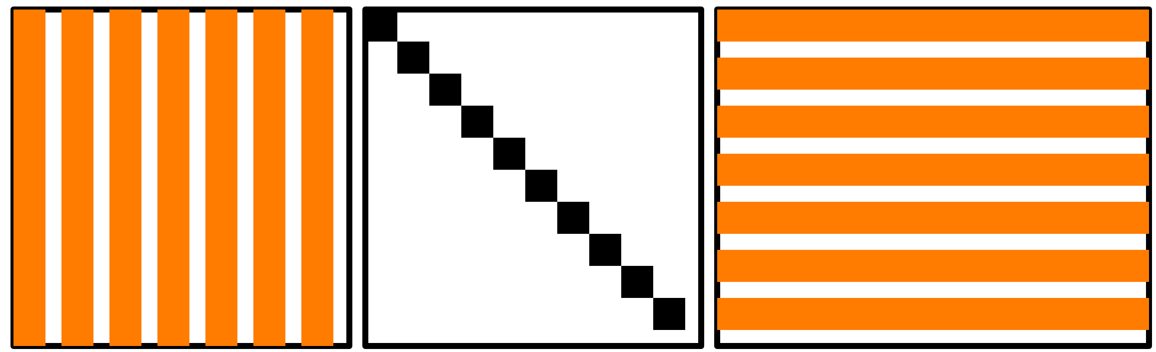
Only $2nr$ degrees of freedom.

In ML: prevents overfitting, reduces dimension, allows for generalization, reveals structure in data.

Singular Value (or Spectral) Decomposition

Any matrix





Ortho-normal Columns

“left” singular vectors

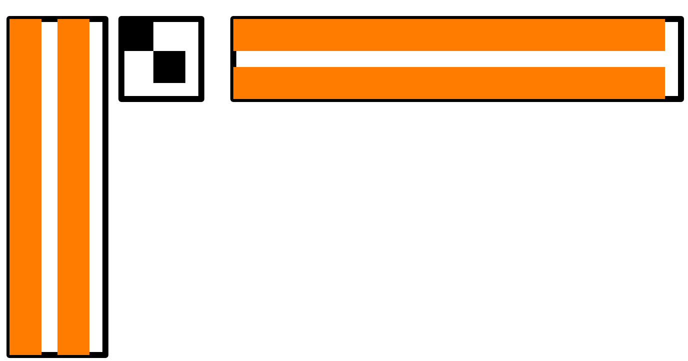
Diagonal, Positive

Singular values

Ortho-normal Rows

“right” singular vectors

IF matrix is low-rank

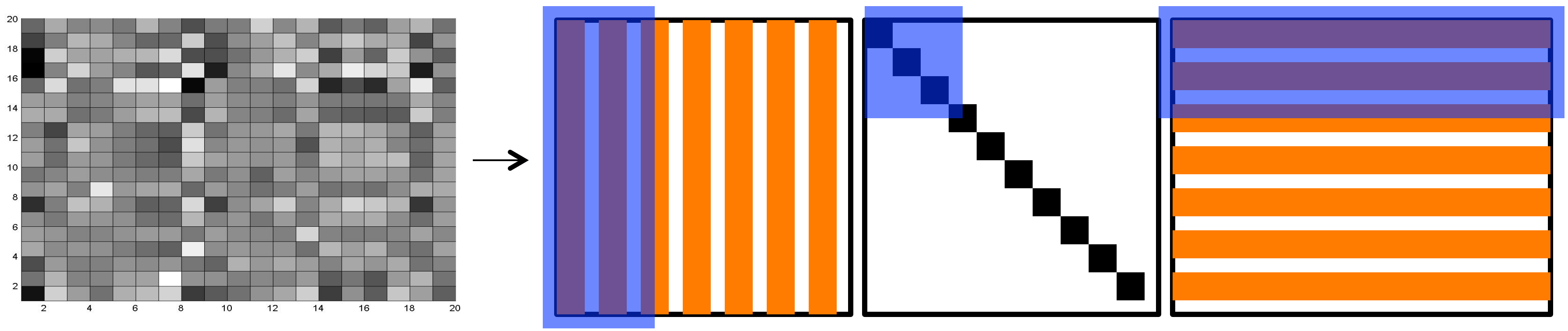


Also called eigenvectors, eigenvalues when matrix symmetric.

Singular values called spectrum of matrix.

Principal Components Analysis

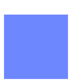
PCA: Approximating given matrix by a low-rank matrix



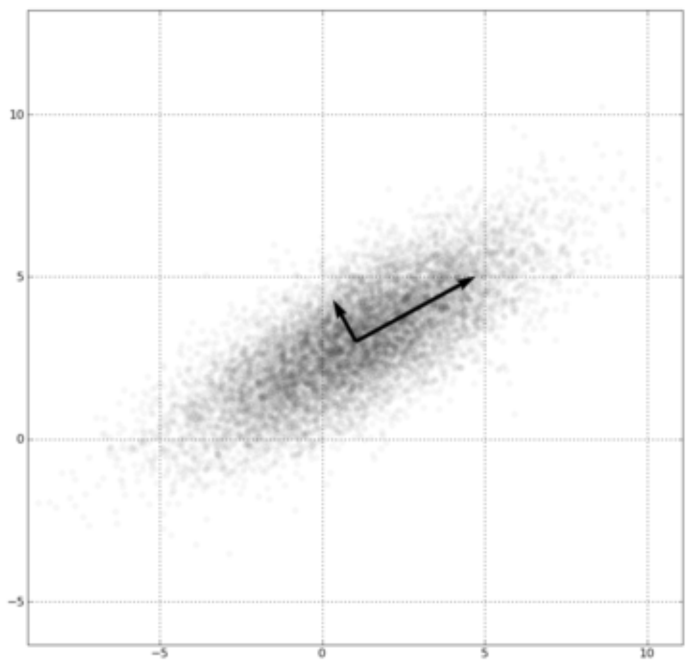
Ortho-normal columns

Diagonal, positive

Ortho-normal rows

 = “best” low-rank approximation

PCA: Geometry



First eigenvector: direction of largest variance

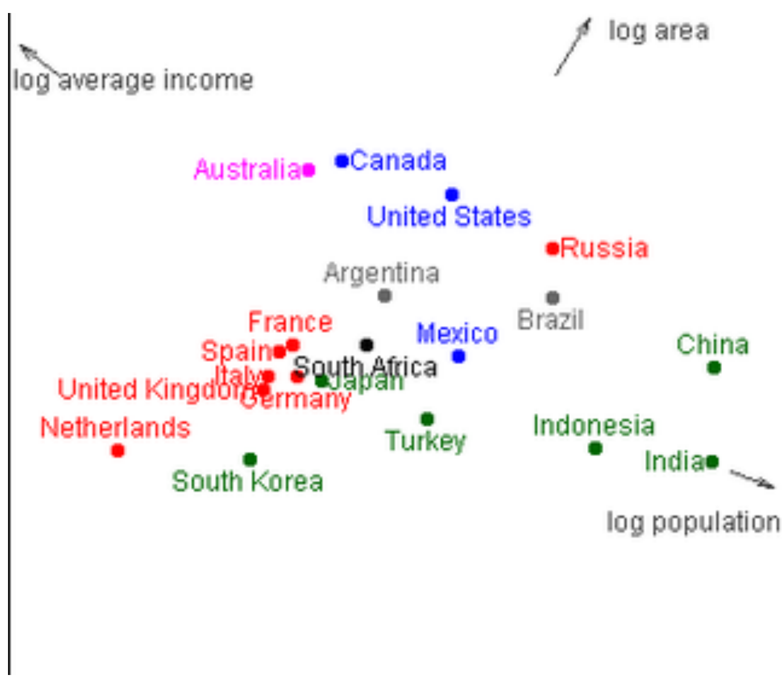
Each subsequent eigenvector: direction of largest residual variance

PCA == retaining first few eigenvectors

== capturing data variation in a smaller dimension

From: wikipedia

Uses of PCA: Visualization



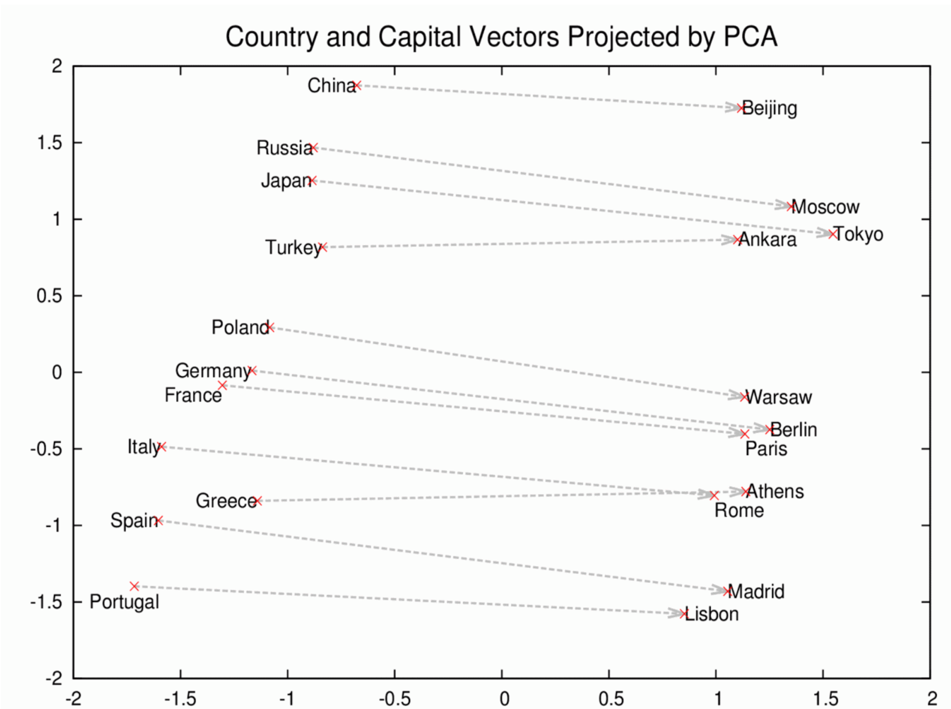
Each country a 3-d vector

(area, avg. income, population)

Then take top-2 PCA

From: wikipedia

Uses of PCA: Visualization



E.g.:

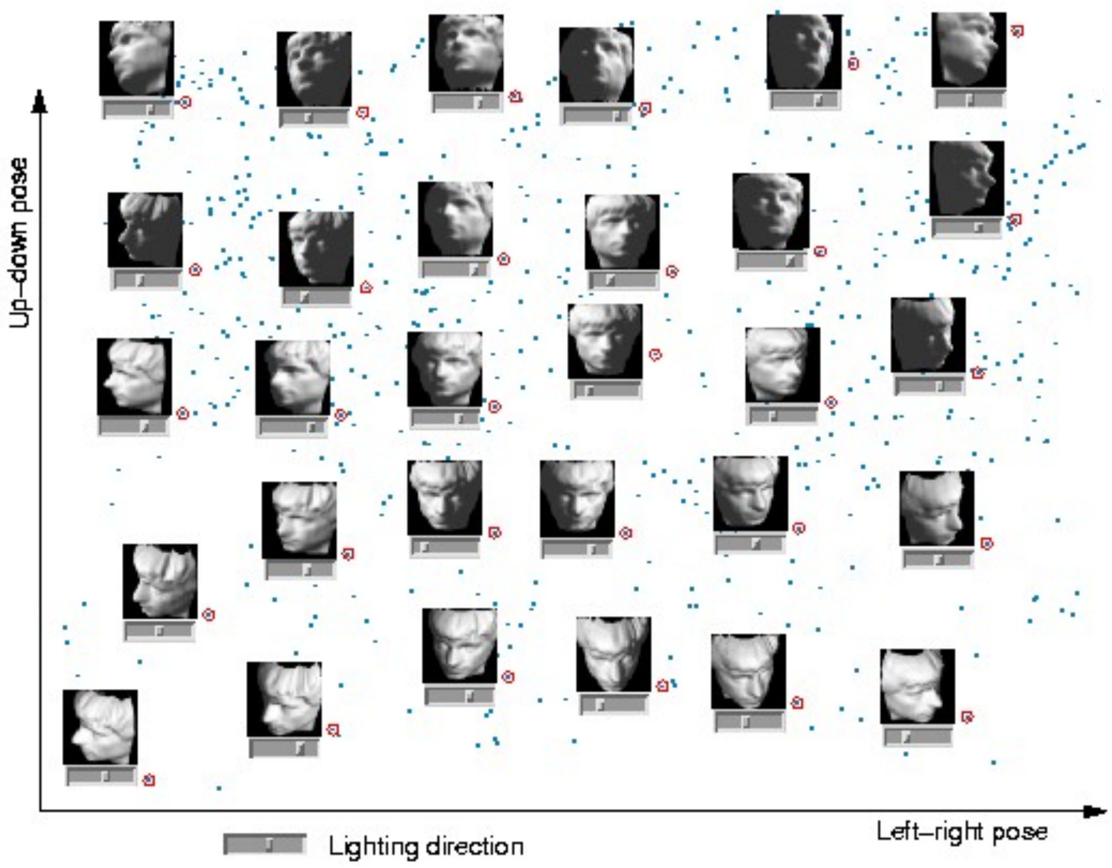
word2vec: associating a vector to
every word
vectors learnt from raw
text data

How do we interpret these vectors ?

A: PCA.

Courtesy: word2vec

Uses of PCA: Visualization

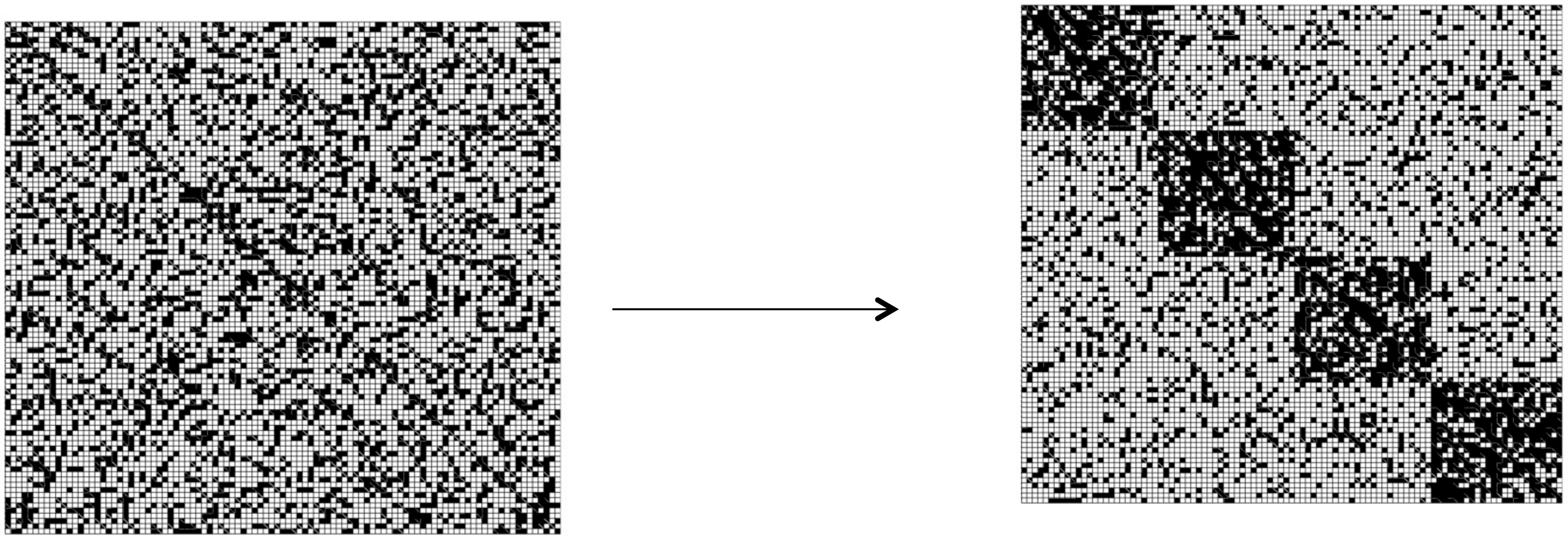


Courtesy: ISOMAP

Spectral Clustering

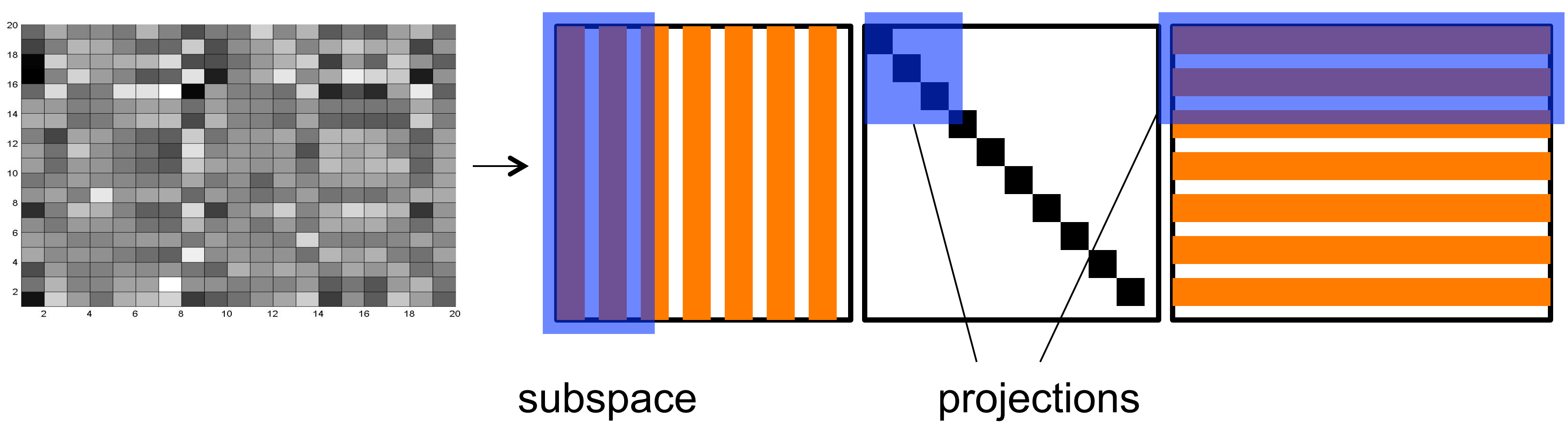
Singular vectors and values can (better) reveal cluster structure in data.

E.g.: graph clustering

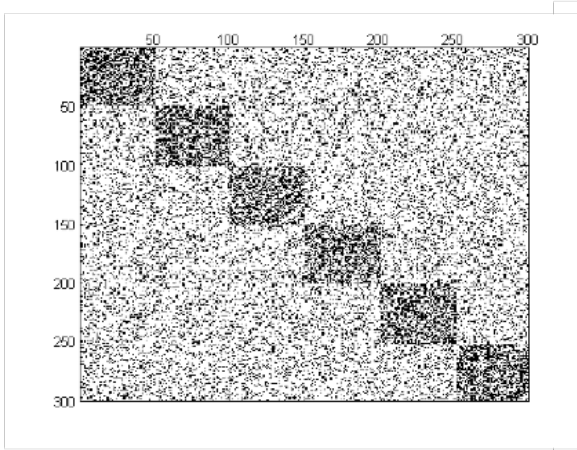


Spectral Clustering

1. Organize data into matrix: each column is one data point
2. Find top r “left” singular vectors. This is the top-r subspace.
3. Represent each data point as a vector in $\mathbb{R}^r$ by projecting onto this subspace.
4. Do some “naïve” clustering of these vectors in $\mathbb{R}^r$

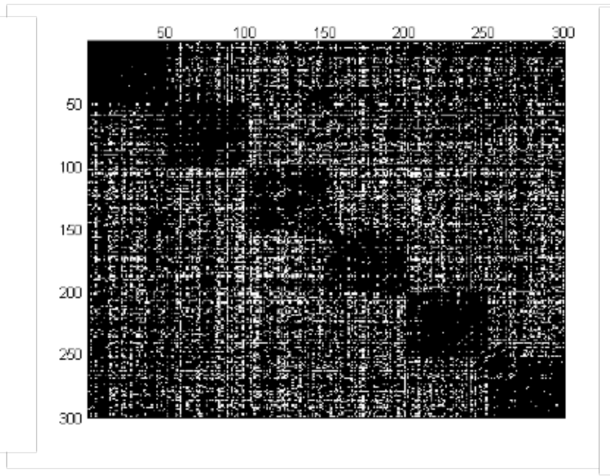


Spectral Clustering



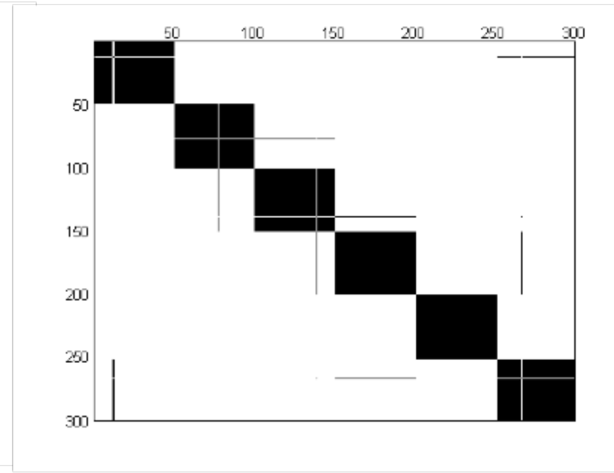
Input graph

(shown with
“correct” ordering)



Naïve Algorithm:

Two nodes in same
cluster if they have
“big enough” common
neighborhood



Spectral clustering:

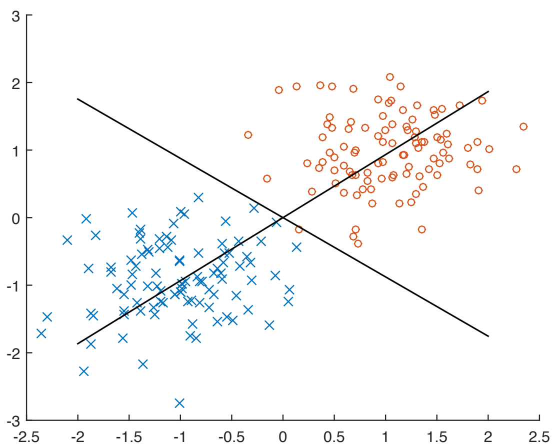
Take top-8 PCA of
Input matrix.

Represent every node by
a vector in $\mathbb{R}^8$

Do k-means++ in $\mathbb{R}^8$

Spectral Clustering

Gaussian mixture models



Given: *Unlabeled* points from several different Gaussian distributions

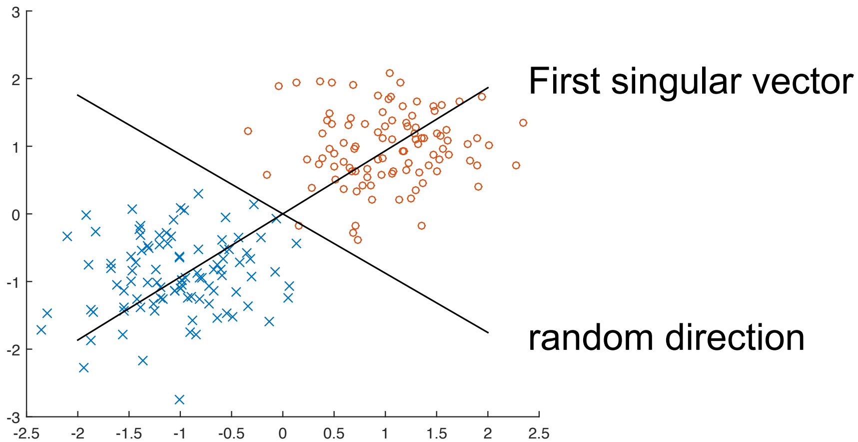
Task: Find means of these Gaussians.

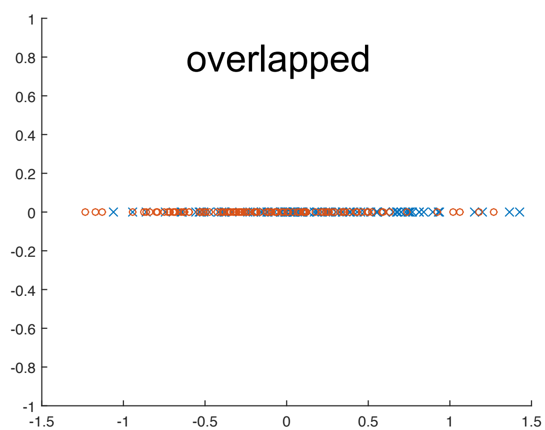
Spectral clustering:

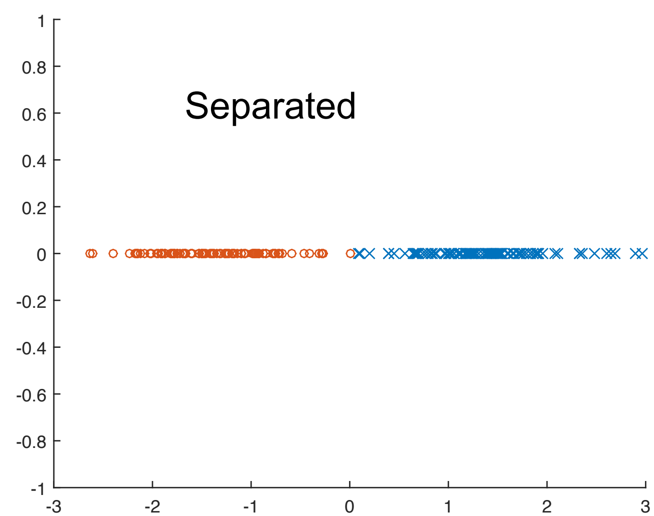
Retains "direction of separation", removes other directions (and noise therein)

Spectral Clustering

Gaussian mixture models

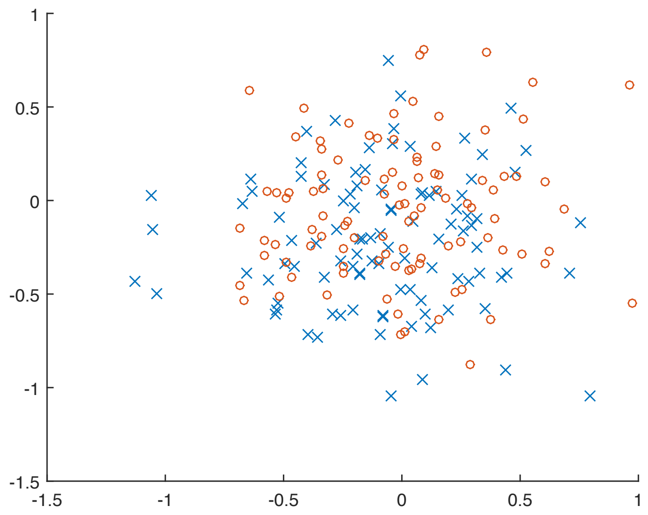




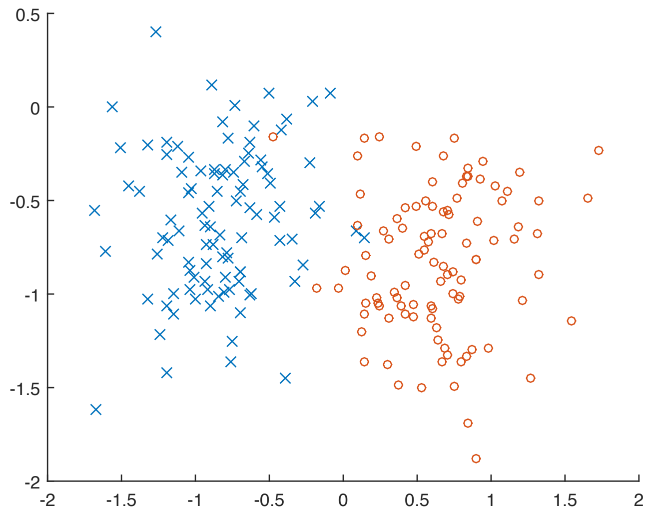


Spectral Clustering

Example: two clusters in $\mathbb{R}^{50}$, projected down to $\mathbb{R}^2$



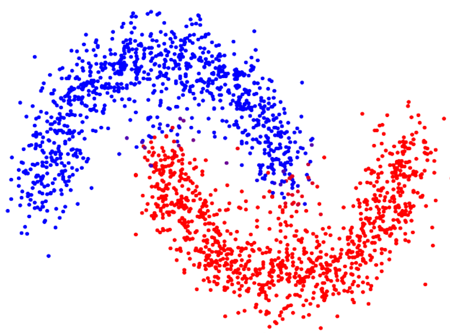
Random 2-d subspace

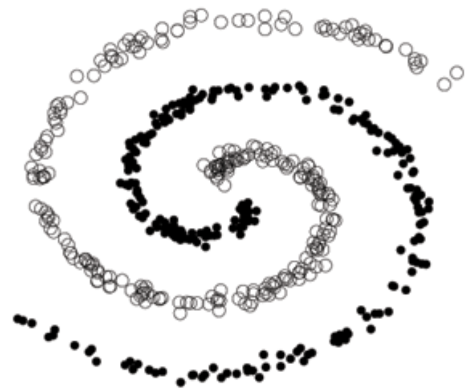


First two singular vectors

Spectral Clustering

More general settings:





$$s_{ij} = e^{-d_{ij}}$$

"kernel trick"

1. Make **pairwise similarity matrix**
2. Find top r singular vectors of similarity matrix
3. Represent each point as a vector in $\mathbb{R}^r$
4. Run k-means or other simple method

Matrix Completion

Task: given few elements of a matrix, find the remaining elements

NOT possible in general.

MAY be possible for low-rank matrix – because few degrees of freedom.

Applications: in a couple of slides …

$$
\begin{array}{cc}
 & \boxed{\begin{array}{cccc} 7 & 5 & 2 & 1 \end{array}} \\
\boxed{\begin{array}{c} 1 \\ 10 \\ 3 \\ 4 \end{array}} & \boxed{\begin{array}{cccc} 7 & 5 & 2 & 1 \\ 70 & 50 & 20 & 10 \\ 21 & 15 & 6 & 3 \\ 28 & 20 & 8 & 4 \end{array}}
\end{array}
$$

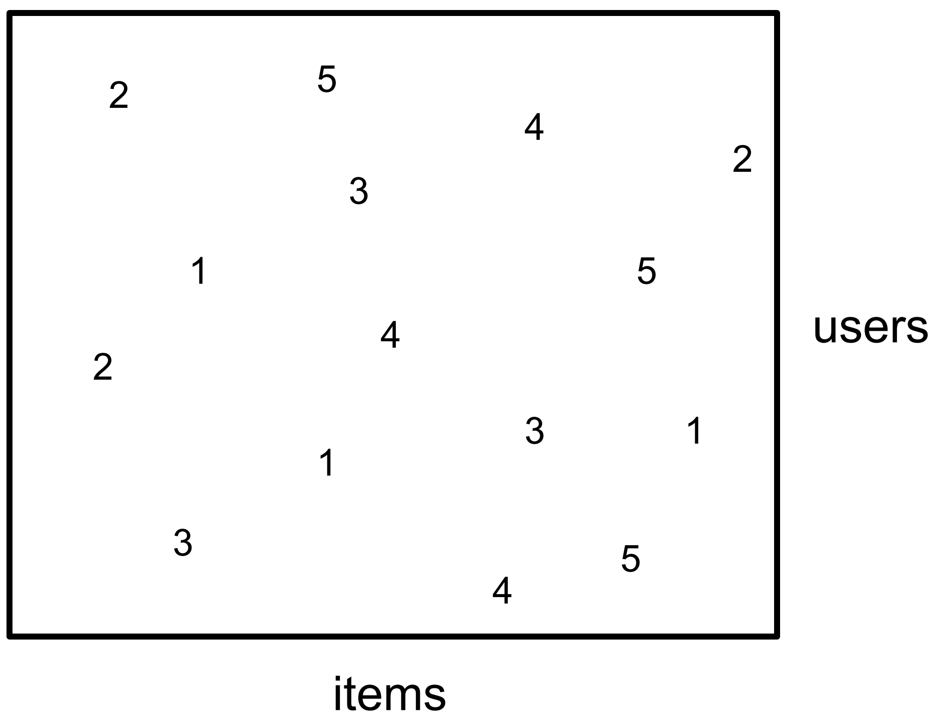
Matrix Completion

Rank-one example:

$$
\begin{bmatrix}
7 & \blacksquare & 2 & \blacksquare \\
\blacksquare & 50 & \blacksquare & 10 \\
21 & \blacksquare & \blacksquare & 3 \\
\blacksquare & 20 & 8 & \blacksquare
\end{bmatrix}
$$

Application: Collaborative Filtering

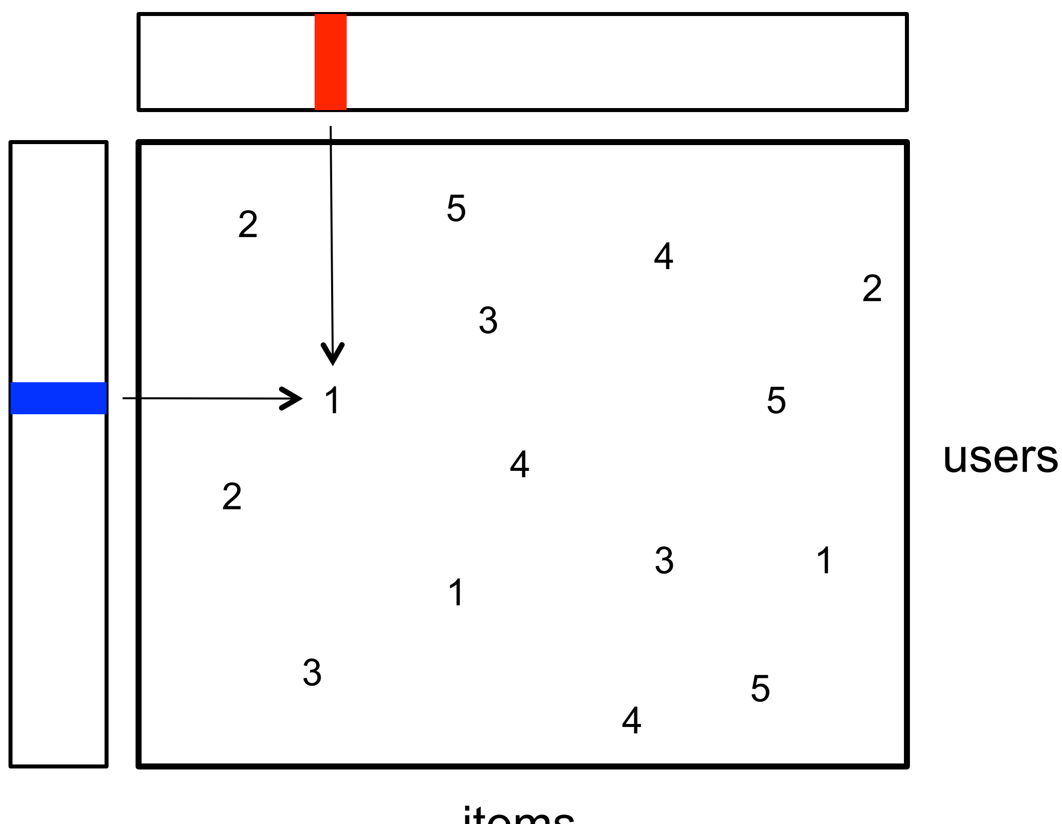
The “netflix problem”: predict user preferences for items
(using data from other users’ preferences)



Application: Collaborative Filtering

Low rank == a “small number of hidden factors” govern our likes and dislikes

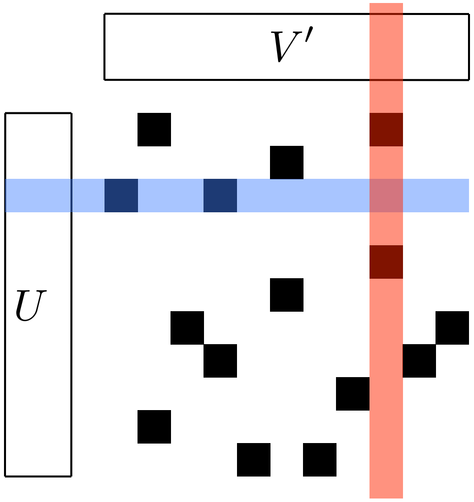
$$m_{ij} \approx f(\langle u_i, v_j \rangle)$$



Matrix Completion

Most popular method: **Alternating Least Squares:**

$$\min_{U,V} \sum_{(i,j)\in\Omega} (m_{ij} - \langle u_i, v_j \rangle)^2$$



Iteratively and alternately:

hold one of U (or V) fixed, solve least-squares for the other

Fast, parellel / distributed etc.

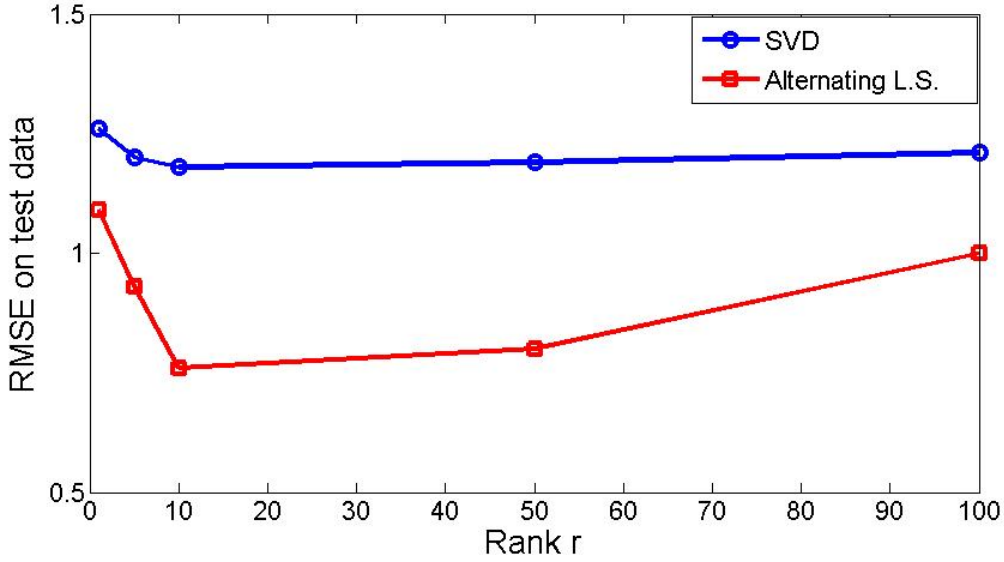
If link function f non-linear, do Alternating Minimization.

Very recently: theoretical guarantees on when this works.

Matrix Completion

Example: Movielens 1M data

6000 users, 4000 movies, 1M ratings



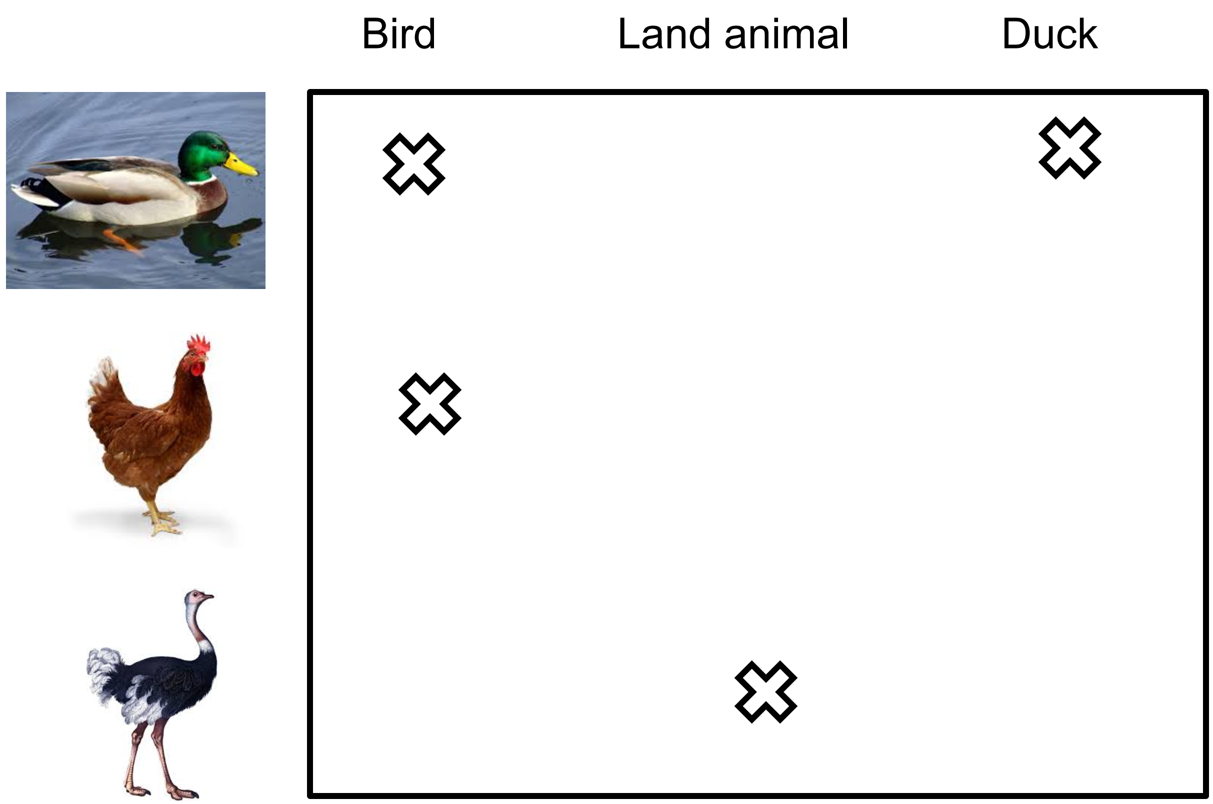
Matrix completion is better than “0-filled SVD”

(i.e. treat unseen ratings as 0 and do rank-r approx.)

Embeddings

Task: given “samples” and “labels”, each a vector of features and a list of some labels for some samples, find labels for remaining samples.

E.g.: image labeling



Embeddings

Idea: map (i.e. “embed”) features to low-dimensional spaces such that inner products of embeddings model affinity

1 if sample i
has label j
0 else

Sample
features

Label
Features

$$m_{ij} \approx f(\, \langle \, Ax_i \, , \, By_j \, \rangle \,)$$

“link
function”

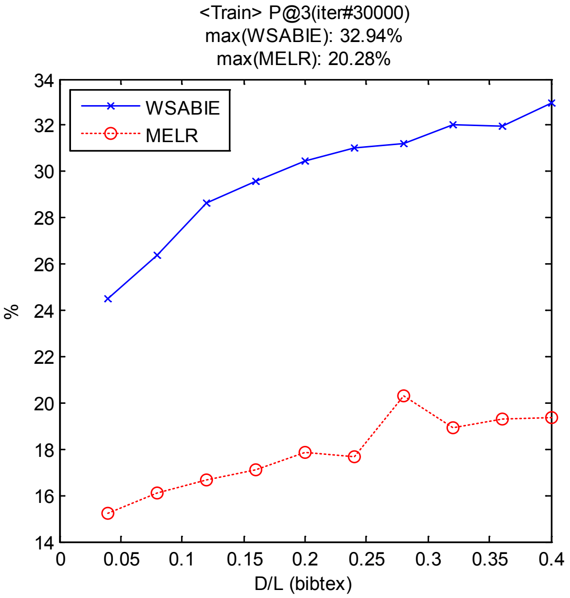
Embedding matrices to be learnt
from data

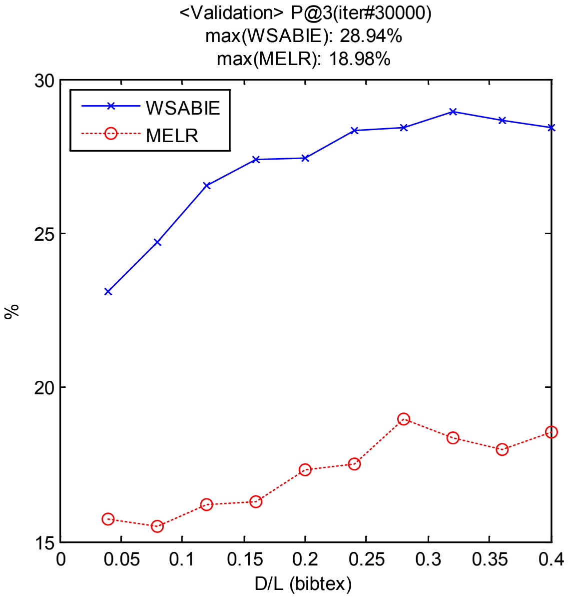
Embeddings

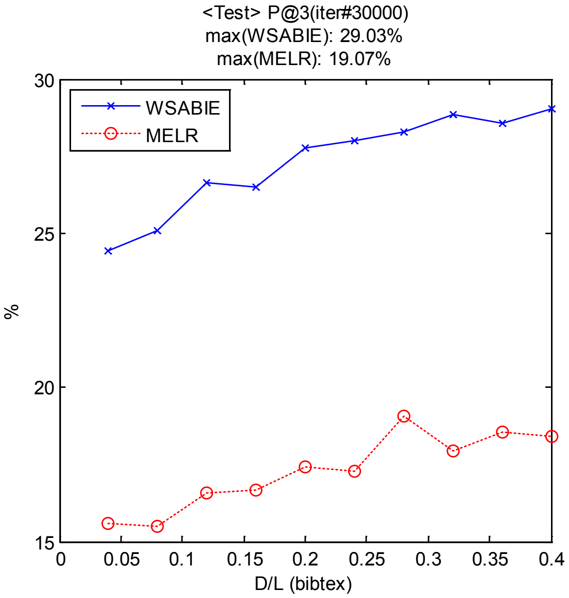
Example: predicting missing authors of documents

“sample”: a paper. Features: words in the title, abstract.

“label”: author name. Features: university, department, etc.

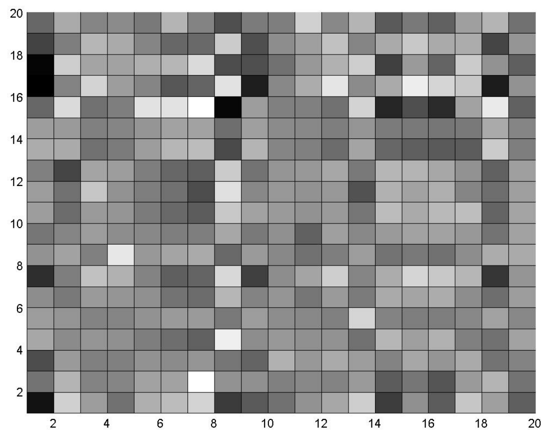


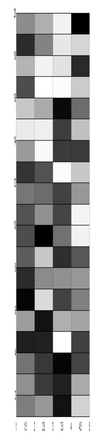


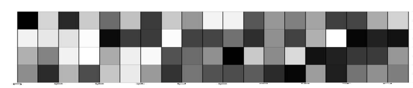


Singular vectors with Structure

Often we would like to impose additional structure on the matrix factors







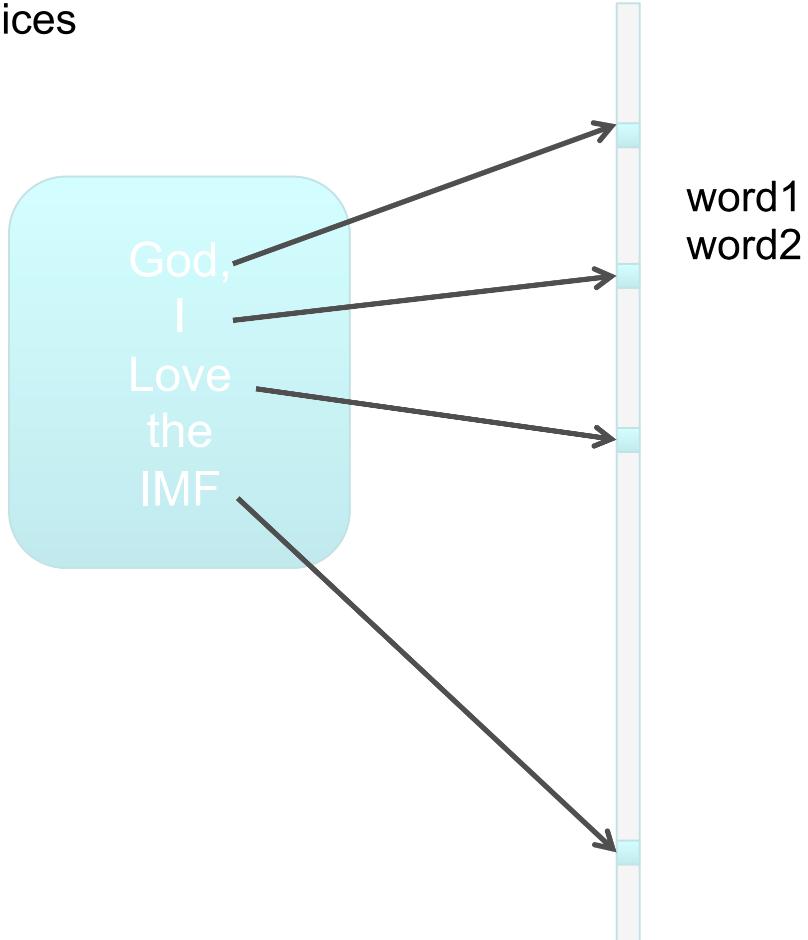
Non-negative Matrix Factorization:
Factors should be positive

Sparse PCA:
Factors should be sparse

…

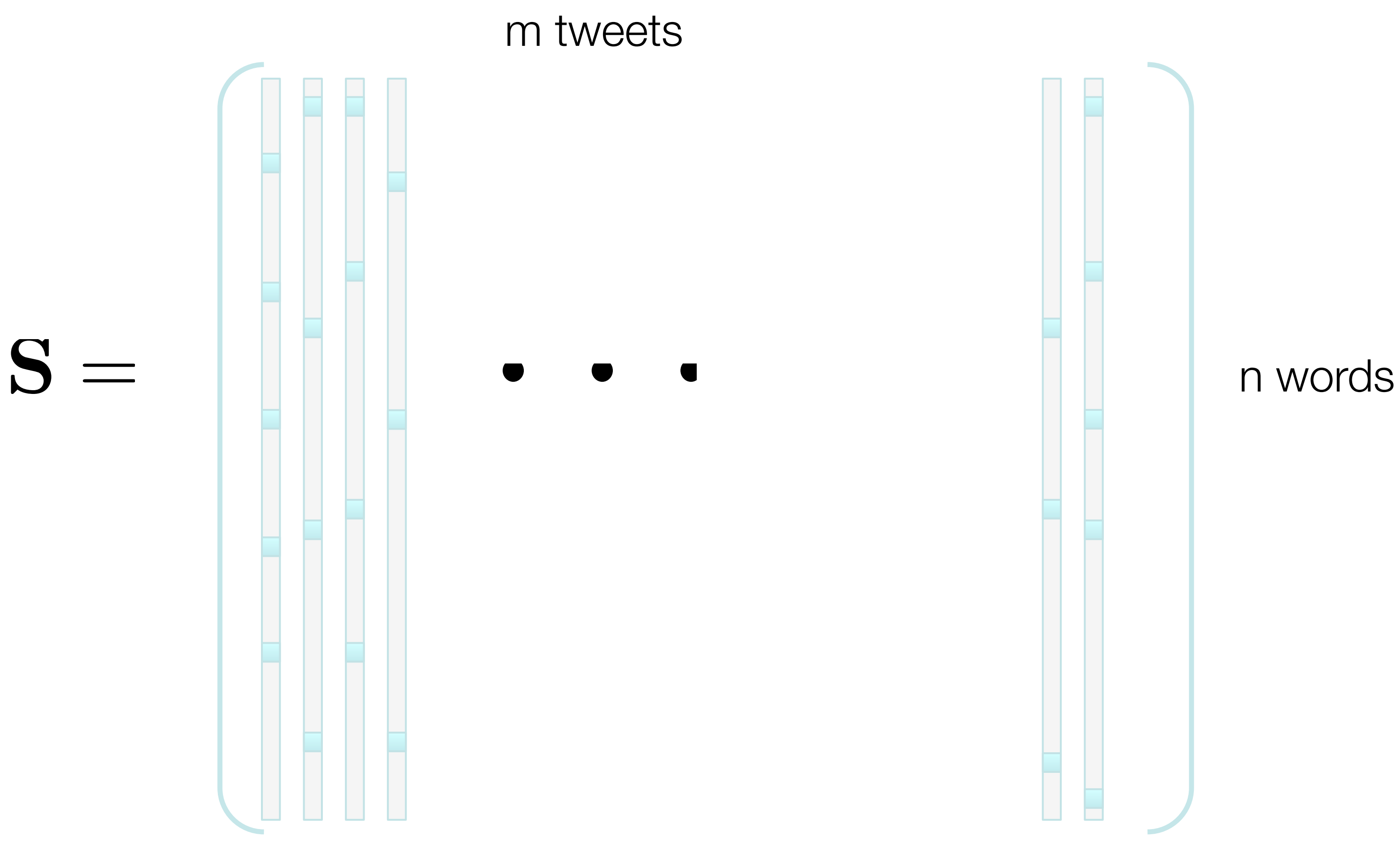
Sparse PCA Example: Greek Twitter Analysis
(Thanks to: Alex Dimakis)

Each tweet as a long (50K), super-sparse vector (5-10 non-zeros) with 1s in word indices



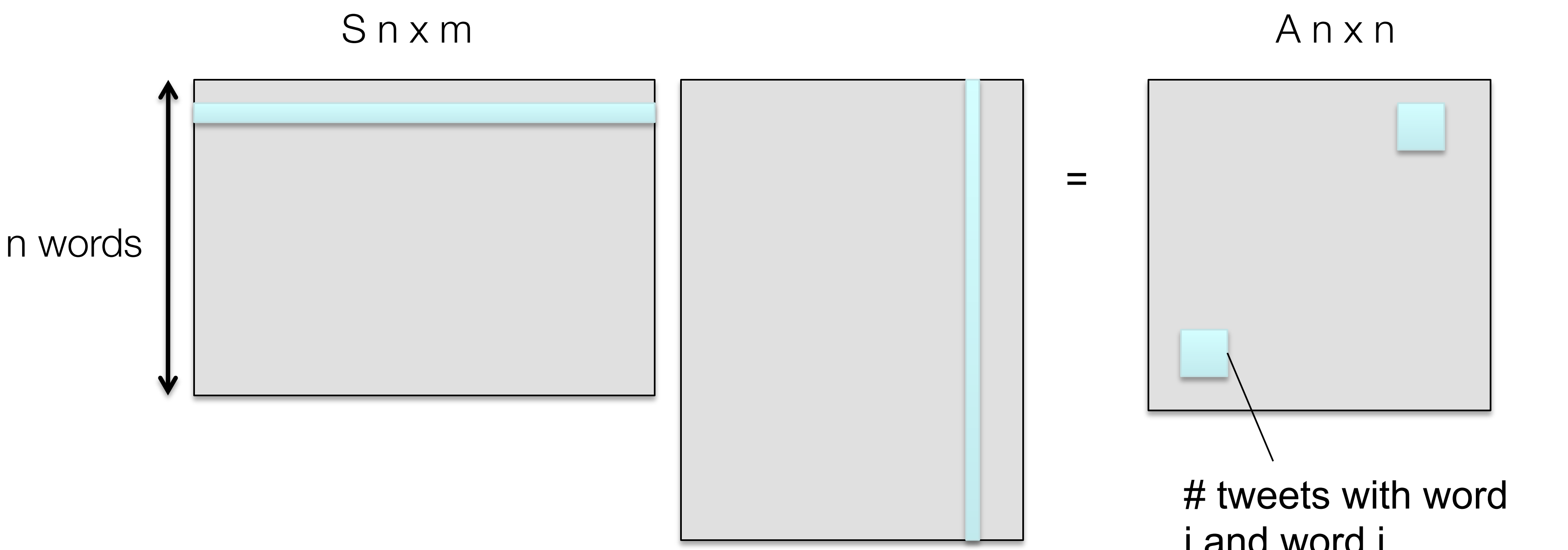
Data Sample Matrix

We collect all tweet vectors in a sample matrix of size $n \times m$



Correlation matrix

$$A = S\,S^T$$



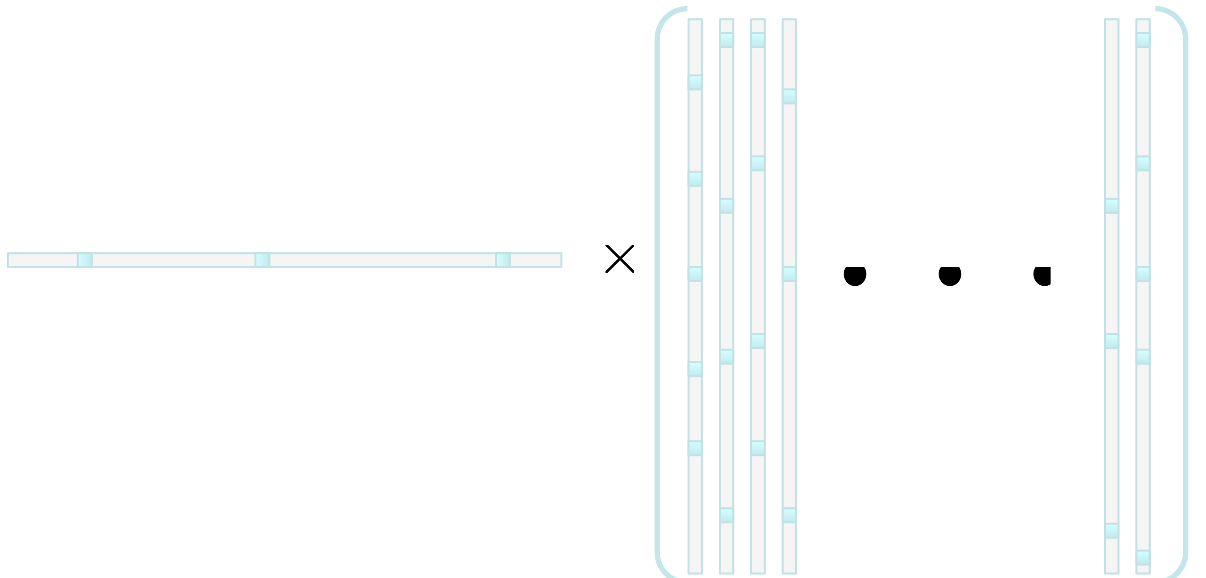
vanilla PCA

$$\underset{\|x\|_2=1}{\arg\max}\; x^T A x$$

Largest Eigenvector.
Maximizes `explained variance' of the data set
Very useful for dimensionality reduction
Easy to compute

PCA finds An `EigenTweet'

Finds a vector that **closely matches** most tweets



i.e, a vector that **maximizes the sum of projections** with each tweet

$$\max \left\|\mathbf{x}^T \mathbf{S}\right\|^2$$

The problem with PCA

- **Top Eigenvector will be dense!**

Dense =
A tweet with thousands of words
(makes no sense)

Eurovision	0.1
Protests	0.02
Greece	.
Morning	.
Deals	.
Engage	
Offers	
Uprising	
Protest	
Elections	
teachers	
Summer	
support	
Schools	
.	
.	
.	
Crisis	
Earthquake	
IMF	0.001

The problem with PCA

- **Top Eigenvector will be dense!**

Dense =
A tweet with thousands of words
(makes no sense)

Eurovision	0.1
Protests	0.02
Greece	.
Morning	.
Deals	.
Engage	
Offers	
Uprising	
Protest	
Elections	
teachers	
Summer	
support	
Schools	
.	
.	
.	
Crisis	
Earthquake	
IMF	0.001

- **We want super sparse**

Sparse = Interpretable

Strong	0.75
Earthquake	0.49
Greece	0.23
Morning	0.31

Experiments (5 days in May 2011)

k=10, top 4 sparse PCs for the data set (65,000 tweets)

`skype, microsoft, acquisition, billion, acquired, acquires, buy, dollars, acquire, google`

`eurovision greece lucas finals final stereo semifinal contest greek watching`

`love received greek know damon amazing hate twitter great sweet`

`downtown athens murder years brutal stabbed incident  camera year crime`

Summary

Low-rank matrices have a lot of algebraic structure

The are widely used in data analysis and machine learning
- visualization
- preprocessing data and dimensionality reduction
- to prevent over-fitting in prediction
- to reveal insights from data

Research Directions:
- algorithm design for matrix factorizations w/ extra structure
- theoretical analyses (esp. statistical guarantees)
- big-data settings (e.g. one-pass or two-pass algorithms)
- applications

Thanks !