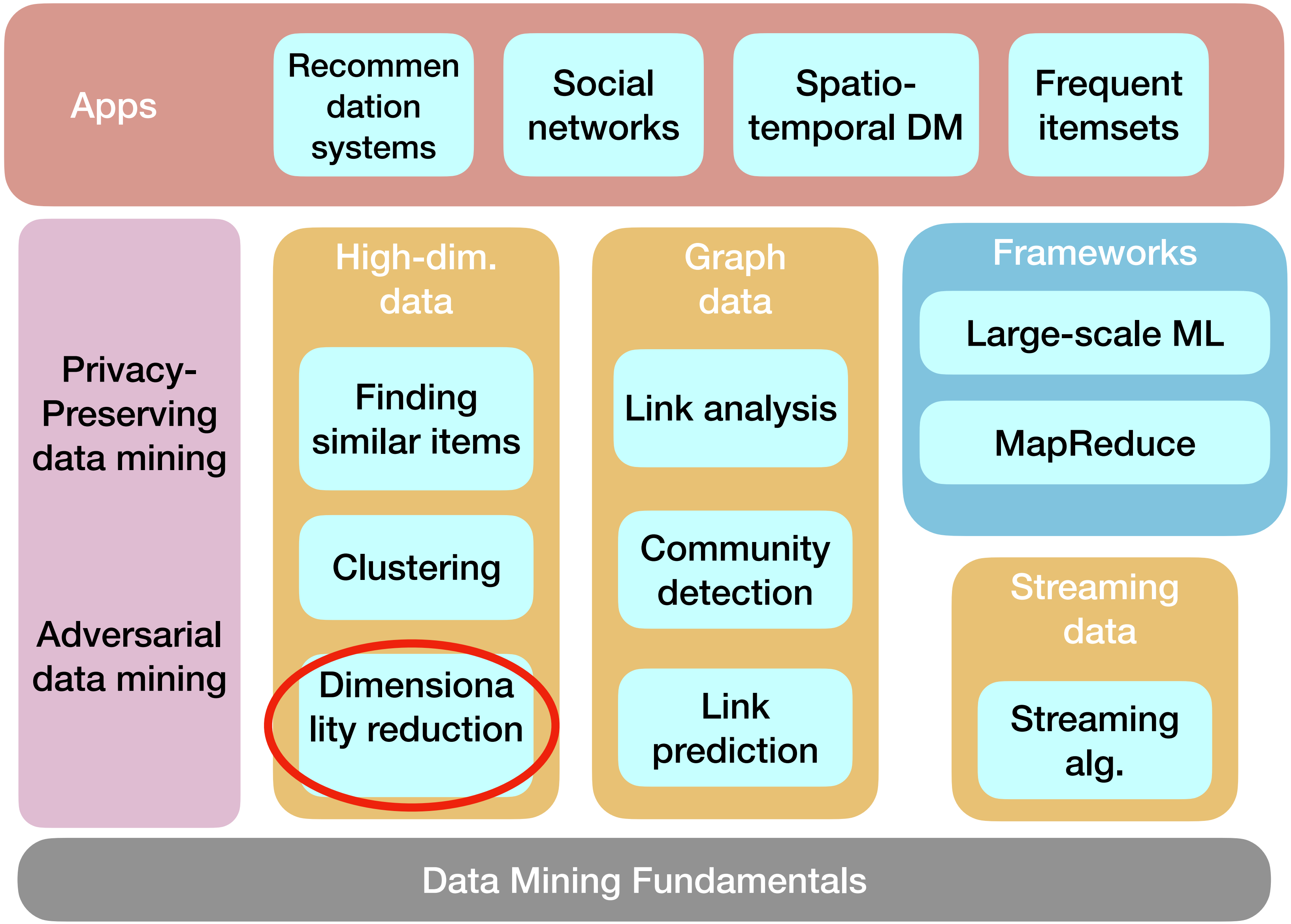
EE359 Data Mining Lecture 5

Dimensionality Reduction

Liyao Xiang
http://xiangliyao.cn/
Shanghai Jiao Tong University

Course Landscape

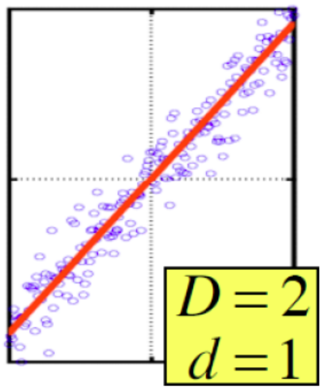


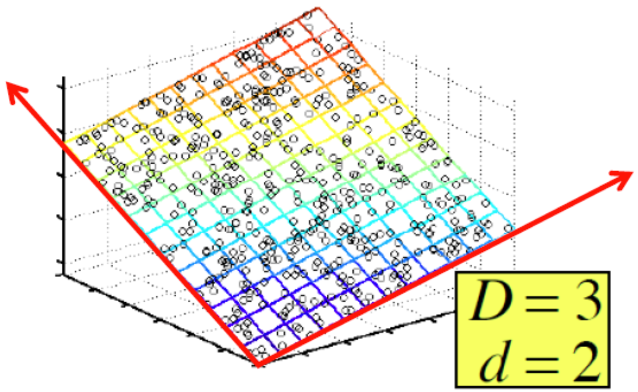
Dimensionality Reduction

- Data lies on or near a low d-dimensional subspace
- Axes of the subspace are effective representation of the data
- **Goal: discover the axis of data!**

The matrix is really "2-dimensional." All rows can be reconstructed by scaling [1 1 1 0 0] or [0 0 0 1 1]

day / customer	We 7/10/96	Th 7/11/96	Fr 7/12/96	Sa 7/13/96	Su 7/14/96
ABC Inc.	1	1	1	0	0
DEF Ltd.	2	2	2	0	0
GHI Inc.	1	1	1	0	0
KLM Co.	5	5	5	0	0
Smith	0	0	0	2	2
Johnson	0	0	0	3	3
Thompson	0	0	0	1	1





Rather than representing every point with 2 coordinates we represent each point with 1 coordinate (corresponding to the position of the point on the red line).

Why Reduce Dimensions?

- Discover hidden correlations/topics
  - Words that occur commonly together
- Remove redundant and noisy features
  - Not all words are useful
- Interpretation and visualization
- Easier storage and processing of the data

Outline

- Singular Value Decomposition
- CUR Decomposition
- Principal Components Analysis
- Factor Analysis

Rank is Dimensionality

- Cloud of points 3D space:
  - Think of point positions as a matrix: $\begin{array}{cc} \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix} & \begin{array}{c} \textbf{A} \\ \textbf{B} \\ \textbf{C} \end{array} \end{array}$

    1 row per point

- We can rewrite coordinates more efficiently!
  - Old basis vectors: [1 0 0] [0 1 0] [0 0 1]
  - New basis vectors: [1 2 1] [-2 -3 1]
  - Then A has new coordinates: [1 0]. B: [0 1], C: [1 1]
    - Notice: We reduced the number of coordinates!

Rank is Dimensionality

- What is rank of a matrix A?
- Number of linearly independent columns of A
- For example:
- Matrix A = $\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$ has rank r=2

  - Why? The first two rows are linearly independent, so the rank is at least 2, but all three rows are linearly dependent (the first is equal to the sum of the second and third) so the rank must be less than 3.

- We can write A as two “basis” vectors: [1 2 1] [-2 -3 1]
- And new coordinates of : [1 0] [0 1] [1 -1]

SVD

$$A_{[m \times n]} = U_{[m \times r]} \Sigma_{[r \times r]} (V_{[n \times r]})^T$$

- A: Input data matrix
  - m x n matrix (e.g., m documents, n terms)
- U: Left singular vectors
  - m x r matrix (m documents, r concepts)
- Σ: Singular values
  - r x r diagonal matrix (strength of each ‘concept,’ r : rank of the matrix A)
- V: Right singular vectors
  - n x r matrix (n terms, r concepts)

SVD

$U^T U = I$; $V^T V = I$
(I: identity matrix)
Column orthonormal

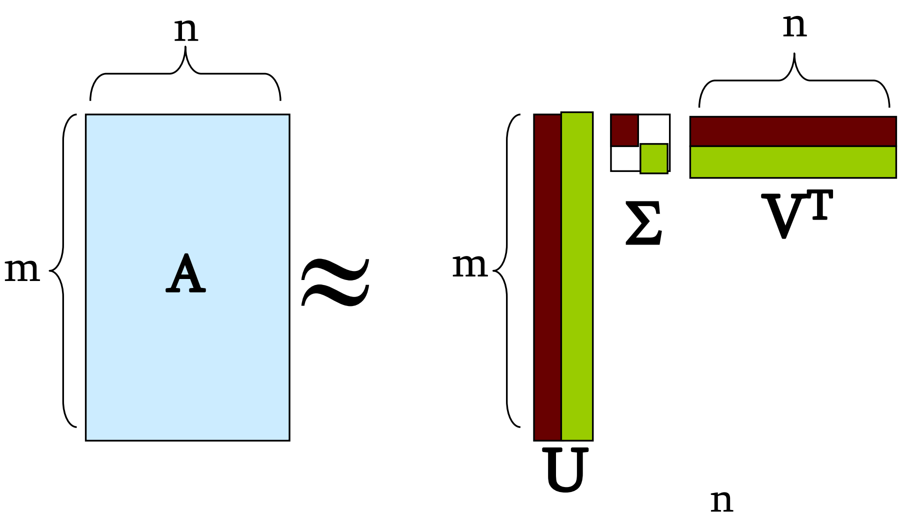
scalar

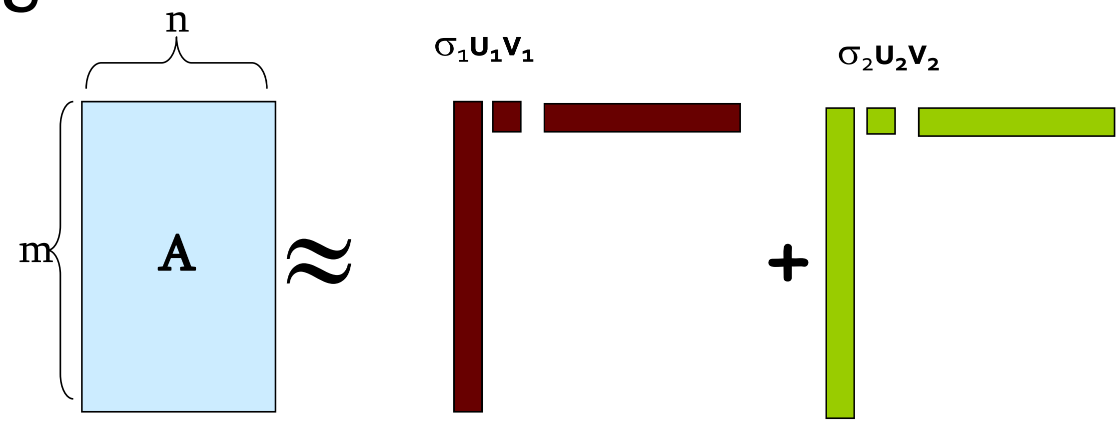
vector

$$\mathbf{A} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i^T$$

vector

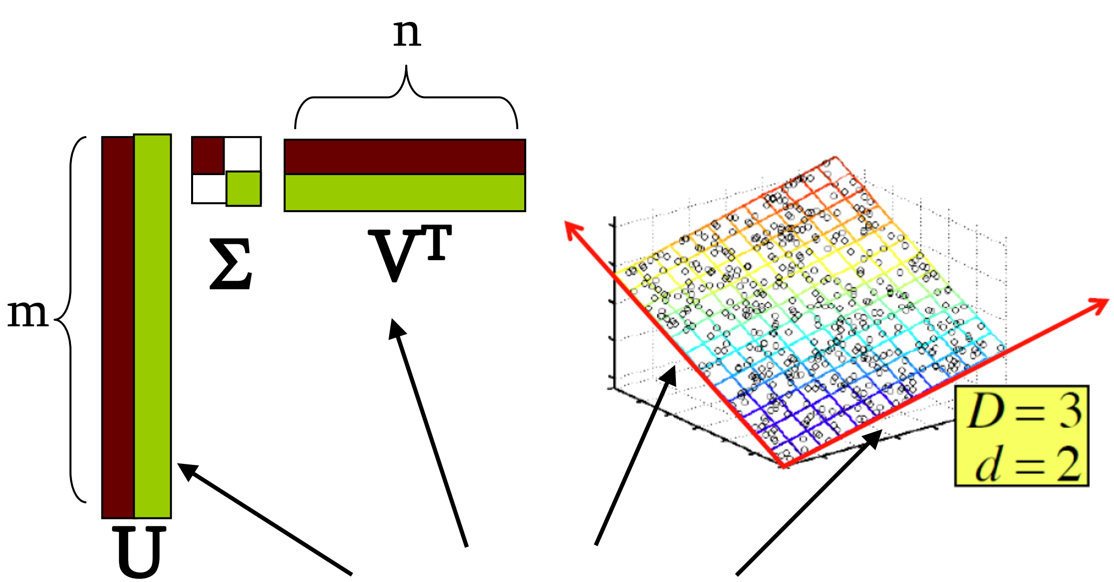
Σ: Diagonal, positive, in decreasing order





SVD Example: Users-to-Movies

$$
\begin{array}{c|ccccc|}
 & \text{Movie 1} & \cdots & & & \text{Movie 5} \\
\hline
\text{User 1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\cdots & \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
 & \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
 & \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
 & 0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
 & 0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
\text{User 7} & 0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2} \\
\hline
\end{array}
=
$$



"Concepts"/latent dimensions/latent factors

SVD Example: Users-to-Movies

Sci-Fi concept

Comedy concept

Rest

U: user-to-concept similarity matrix

“Strength” of the Sci-Fi concept

$$
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
=
\begin{bmatrix}
\mathbf{0.13} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.68} & 0.11 & -0.05 \\
0.15 & \mathbf{-0.59} & \mathbf{0.65} \\
0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{-0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

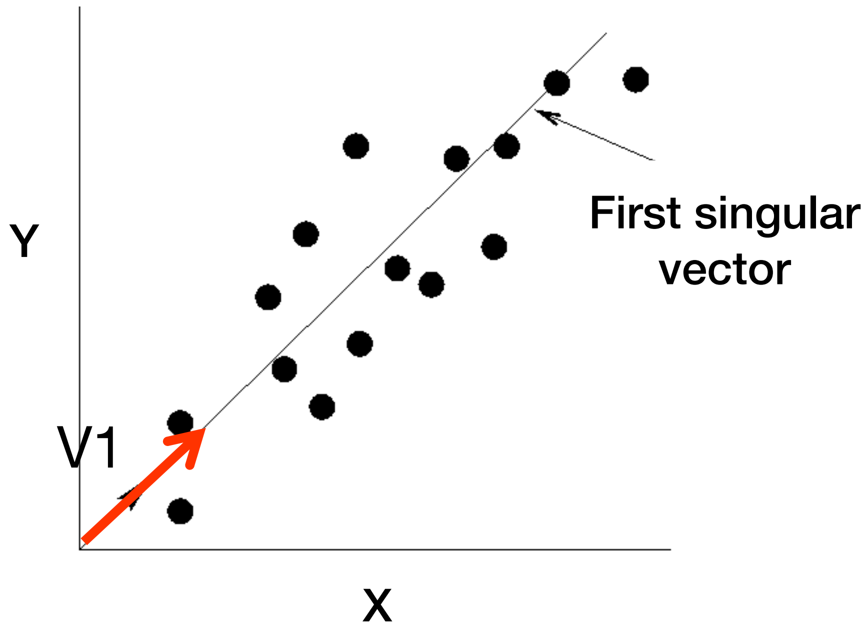
Sci-Fi concept

Comedy concept

Rest

V^T: concept-to-movie similarity matrix

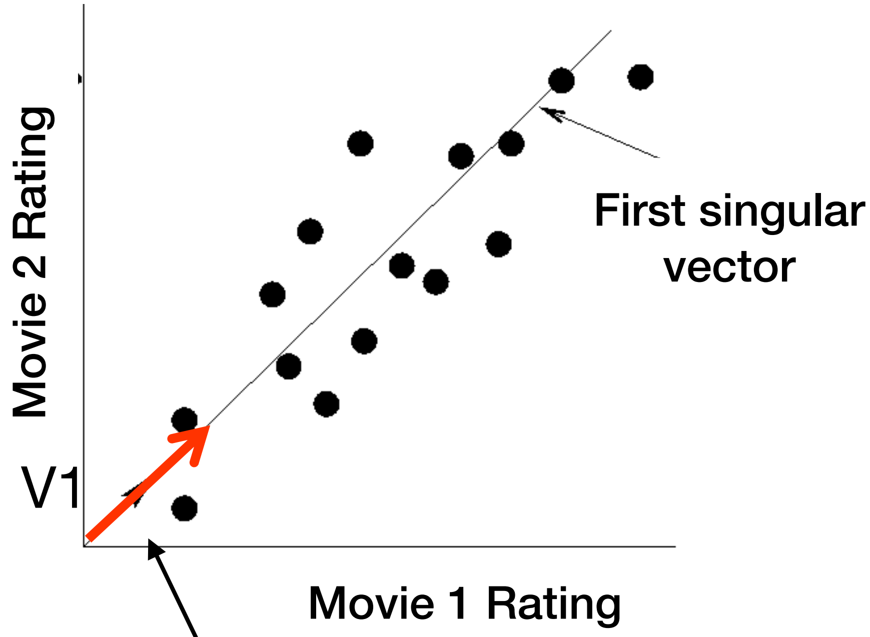
SVD-Another View



- Instead of using two coordinates (x, y) to describe point locations, let’s use only one coordinate (z)
- Point’s position is its location along vector V1

SVD-Another View

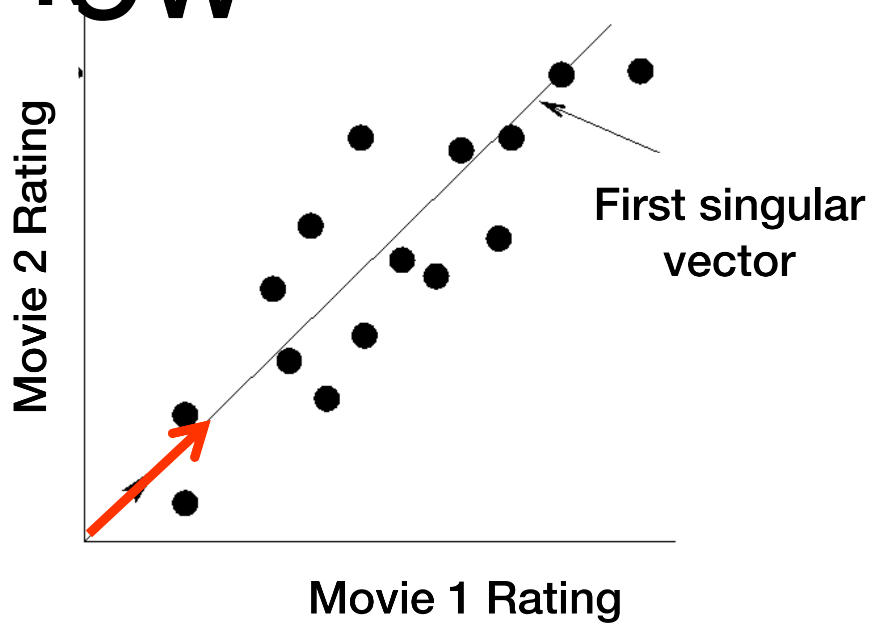
Linear combinations of movies' ratings form a concept!



Variance ('spread') on the V1 axis

$$\begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\ \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\ \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\ 0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\ 0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\ 0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2} \end{bmatrix} = \begin{bmatrix} \mathbf{0.13} & 0.02 & -0.01 \\ \mathbf{0.41} & 0.07 & -0.03 \\ \mathbf{0.55} & 0.09 & -0.04 \\ \mathbf{0.68} & 0.11 & -0.05 \\ 0.15 & \mathbf{-0.59} & \mathbf{0.65} \\ 0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\ 0.07 & \mathbf{-0.29} & \mathbf{0.32} \end{bmatrix} \times \begin{bmatrix} \mathbf{12.4} & 0 & 0 \\ 0 & \mathbf{9.5} & 0 \\ 0 & 0 & \mathbf{1.3} \end{bmatrix} \times \begin{bmatrix} \mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\ 0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

SVD-Another View



UΣ: gives the coordinates of the points in the projection axis

Projection of users on the "Sci-Fi" axis

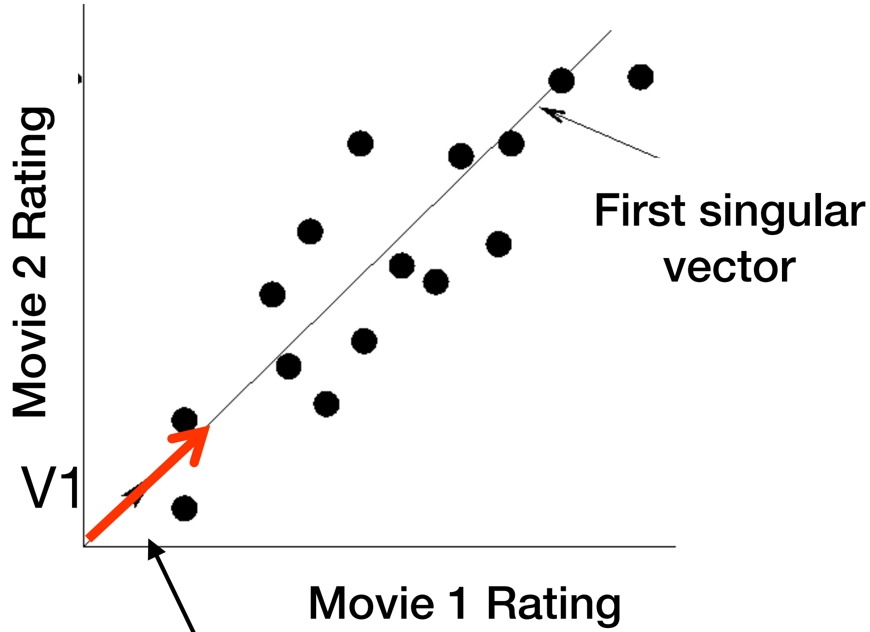
$$
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
=
\begin{bmatrix}
1.61 & 0.19 & -0.01 \\
5.08 & 0.66 & -0.03 \\
6.82 & 0.85 & -0.05 \\
8.43 & 1.04 & -0.06 \\
1.86 & -5.60 & 0.84 \\
0.86 & -6.93 & -0.87 \\
0.86 & -2.75 & 0.41
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

SVD-Another View

How to choose V1? Minimize
the sum of reconstruction errors

$$\sum_{i=1}^{N}\sum_{j=1}^{D}\left\|x_{ij} - z_{ij}\right\|^2$$

New coordinates



SVD gives `best' axis to project on

$$\begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\ \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\ \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\ 0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\ 0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\ 0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2} \end{bmatrix} = \begin{bmatrix} \mathbf{0.13} & 0.02 & -0.01 \\ \mathbf{0.41} & 0.07 & -0.03 \\ \mathbf{0.55} & 0.09 & -0.04 \\ \mathbf{0.68} & 0.11 & -0.05 \\ 0.15 & \mathbf{-0.59} & \mathbf{0.65} \\ 0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\ 0.07 & \mathbf{-0.29} & \mathbf{0.32} \end{bmatrix} \times \begin{bmatrix} \mathbf{12.4} & 0 & 0 \\ 0 & \mathbf{9.5} & 0 \\ 0 & 0 & \mathbf{1.3} \end{bmatrix} \times \begin{bmatrix} \mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\ 0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

SVD-Dimensionality Reduction

- How is dimensionality reduction done?

  Set the smallest singular values to zero!

$$
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\approx
\begin{bmatrix}
\mathbf{0.13} & 0.02 & \cancel{-0.01} \\
\mathbf{0.41} & 0.07 & \cancel{-0.03} \\
\mathbf{0.55} & 0.09 & \cancel{-0.04} \\
\mathbf{0.68} & 0.11 & \cancel{-0.05} \\
0.15 & \mathbf{-0.59} & \cancel{\mathbf{0.65}} \\
0.07 & \mathbf{-0.73} & \cancel{\mathbf{-0.67}} \\
0.07 & \mathbf{-0.29} & \cancel{\mathbf{0.32}}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \cancel{\mathbf{1.3}}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\
\cancel{0.40} & \cancel{\mathbf{-0.80}} & \cancel{0.40} & \cancel{0.09} & \cancel{0.09}
\end{bmatrix}
$$

SVD-Dimensionality Reduction

- How is dimensionality reduction done?

Set the smallest singular values to zero!

$$\begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\ \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\ \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\ 0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\ 0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\ 0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2} \end{bmatrix} \approx \begin{bmatrix} \mathbf{0.92} & \mathbf{0.95} & \mathbf{0.92} & 0.01 & 0.01 \\ \mathbf{2.91} & \mathbf{3.01} & \mathbf{2.91} & -0.01 & -0.01 \\ \mathbf{3.90} & \mathbf{4.04} & \mathbf{3.90} & 0.01 & 0.01 \\ \mathbf{4.82} & \mathbf{5.00} & \mathbf{4.82} & 0.03 & 0.03 \\ 0.70 & \mathbf{0.53} & 0.70 & \mathbf{4.11} & \mathbf{4.11} \\ -0.69 & 1.34 & -0.69 & \mathbf{4.78} & \mathbf{4.78} \\ 0.32 & \mathbf{0.23} & 0.32 & \mathbf{2.01} & \mathbf{2.01} \end{bmatrix}$$

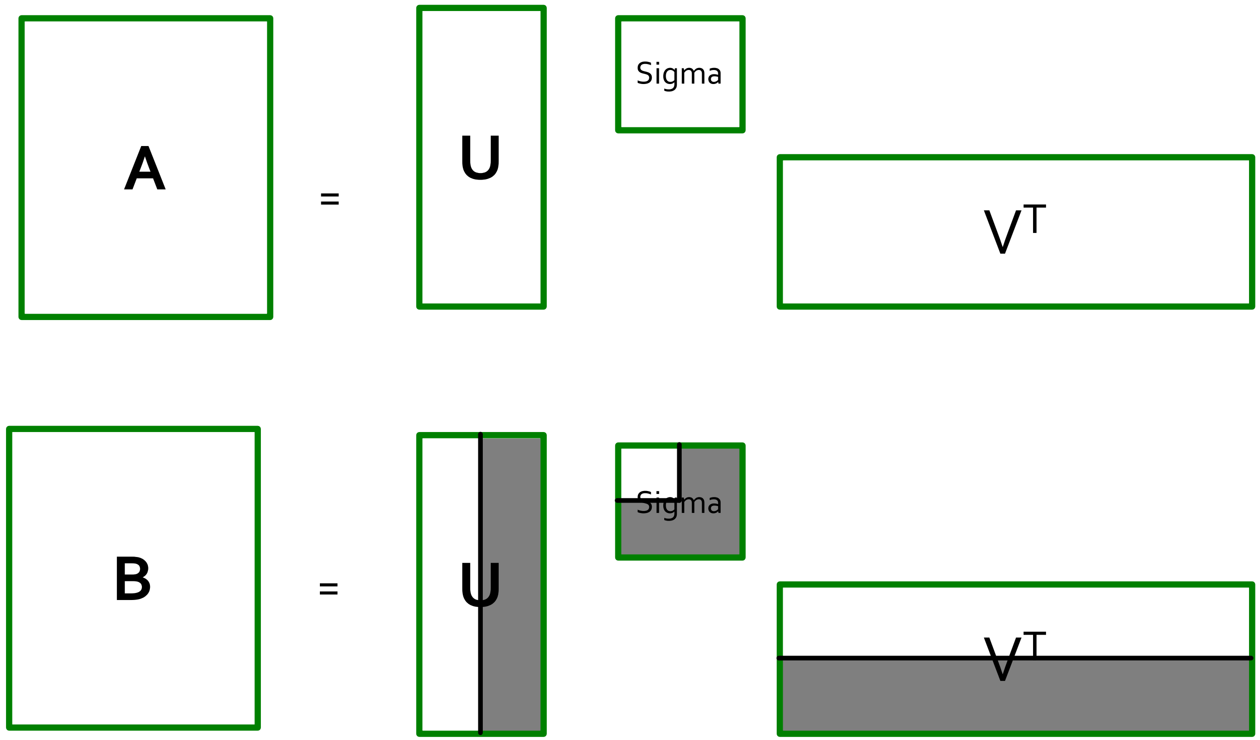
Frobenius norm:

$\|M\|_F = \sqrt{\Sigma_{ij} M_{ij}^2}$

$\|A-B\|_F = \sqrt{\Sigma_{ij} (A_{ij}-B_{ij})^2}$

is "small"

SVD—Best Low Rank Approx.



B approximates A!

- What do we mean by "best?"
  - B is a solution to $\min_B \| A - B \|_F$ where rank(B) = k

SVD—Best Low Rank Approx.

- **Theorem**: Let A = U Σ V^T and B = U S V^T where S = diagonal r x r matrix with $s_i=\sigma_i$ (i=1…k) else $s_i=0$, then B is a best rank(B)=k approx. to A
- How many **σs** to keep?
  - Rule-of-a thumb: keep 80-90% of ‘energy’ $= \sum_i \sigma_i^2$

$$n\left\{\overbrace{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix}}^{m}\right. = \sigma_1 \quad u_1 \quad v^T_1 \quad + \quad \sigma_2 \quad u_2 \quad v^T_2 \quad + \ldots$$

Assume: $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots$

How to Compute SVD?

- SVD gives us: $A = U \Sigma V^T$
- **Eigen**-decomposition: $A = X \Lambda X^T$
  - A is symmetric
  - U, V, X are orthonormal ($U^TU=I$)
  - Σ, Λ are diagonal
- Now let's calculate:
  - $\mathbf{AA^T} = U \Sigma V^T(U \Sigma V^T)^T = U \Sigma V^T(V \Sigma^T U^T) = \mathbf{U \Sigma \Sigma^T U^T}$
  - $\mathbf{A^TA} = V \Sigma^T U^T (U \Sigma V^T) = \mathbf{V \Sigma \Sigma^T V^T}$

How to Compute SVD?

- We need a method for finding the **principal eigenvalue** (the largest one) and the corresponding **eigenvector** of a symmetric matrix M

- Method:
  - Start with any "guess eigenvector" x_0
  - Construct $x_{k+1} = \dfrac{Mx_k}{||Mx_k||}$ for k = 0, 1, …
    - || … || denotes the Frobenius norm
  - Stop when consecutive x_k changes little

How to Compute SVD?

- Once you have the principal eigenvector x, you find its eigenvalue λ by $\lambda = x^T M x$
  - Proof: By the definition of eigenvalue, we have $x\lambda = Mx$
- Example: if we take $x^T = [0.53, 0.85]$, then

$$\lambda = [0.53 0.85] \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 0.53 \\ 0.85 \end{bmatrix} = 4.25$$

- Eliminate the portion of the matrix M that can be generated by the first eigenpair λ and M: $M^* = M - \lambda x x^T$
- Recursively find the principal eigenpair for M^*, eliminate the effect of that pair, and so on

SVD — Complexity

- To compute SVD:
  - $O(nm^2)$ or $O(n^2m)$ (whichever is less)
- But:
  - Less work, if we just want singular values
  - or if we want first k singular vectors
  - or if the matrix is sparse
- Implemented in linear algebra packages like LINPACK, Matlab, SPlus, Mathematica …

Case Study

- How to find users that like Movie 1?
- Map query into a `concept space' — how?

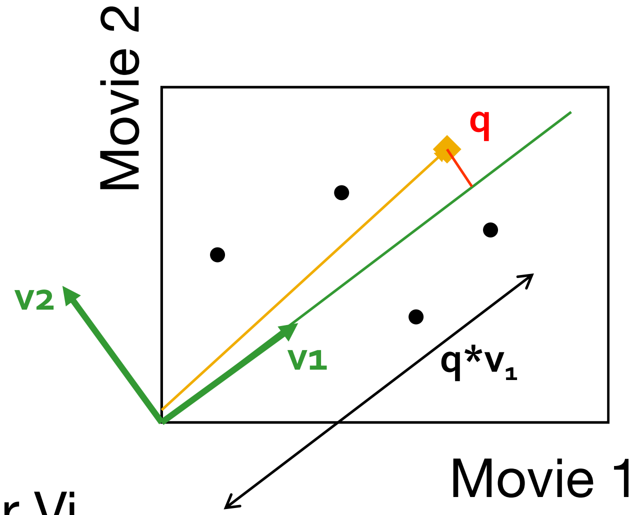
$$
\begin{array}{c}
\begin{array}{ccccc} \text{Movie 1} & & & & \end{array} \\
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
\mathbf{0.13} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.68} & 0.11 & -0.05 \\
0.15 & \mathbf{-0.59} & \mathbf{0.65} \\
0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{-0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

Case Study

- How to find users that like Movie 1?
- Map query into a `concept space'

$$\mathbf{q} = \begin{array}{c} \text{Movie 1} \\ \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Project into concept space:
Inner product with each `concept' vector Vi



$$\mathbf{q} = \begin{array}{c} \text{Movie 1} \\ \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \times \begin{bmatrix} 0.56 & 0.12 \\ 0.59 & -0.02 \\ 0.56 & 0.12 \\ 0.09 & -0.69 \\ 0.09 & -0.69 \end{bmatrix} = \begin{array}{cc} \text{Sci-Fi concept} & \\ \begin{bmatrix} 2.8 & 0.6 \end{bmatrix} \end{array}$$

Movie-to-concept

Case Study

- How to find users that like Movie 1?
- Map query into a `concept space'

$$\mathbf{d} = \begin{array}{c} \begin{array}{ccccc} & \text{Movie 2} & \text{Movie 3} & & \end{array} \\ \begin{bmatrix} 0 & 4 & 5 & 0 & 0 \end{bmatrix} \end{array} \times \begin{bmatrix} 0.56 & 0.12 \\ 0.59 & -0.02 \\ 0.56 & 0.12 \\ 0.09 & -0.69 \\ 0.09 & -0.69 \end{bmatrix} = \begin{bmatrix} 5.2 & 0.4 \end{bmatrix}$$

$$\mathbf{d} = \begin{bmatrix} 0 & 4 & 5 & 0 & 0 \end{bmatrix} \dashrightarrow \begin{bmatrix} 5.2 & 0.4 \end{bmatrix}$$

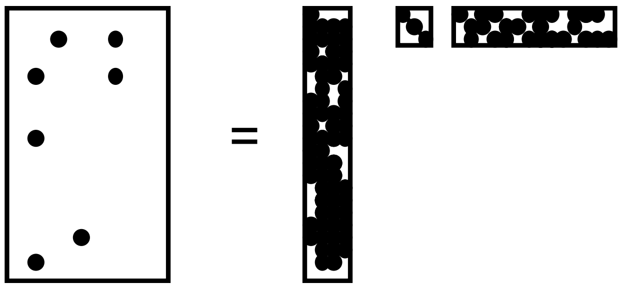
$$\mathbf{q} = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \end{bmatrix} \dashrightarrow \begin{bmatrix} 2.8 & 0.6 \end{bmatrix}$$

User **d** that rated Movie 2 & 3 will be similar to user **q** that rated Movie 1, although they have **zero ratings in common**!

SVD Drawbacks

✓ Optimal low-rank approximation in terms of Frobenius norm

- Interpretability problem:
    - A singular vector specifies a linear combination of all input columns or rows
- Lack of sparsity:
    - Singular vectors are dense!

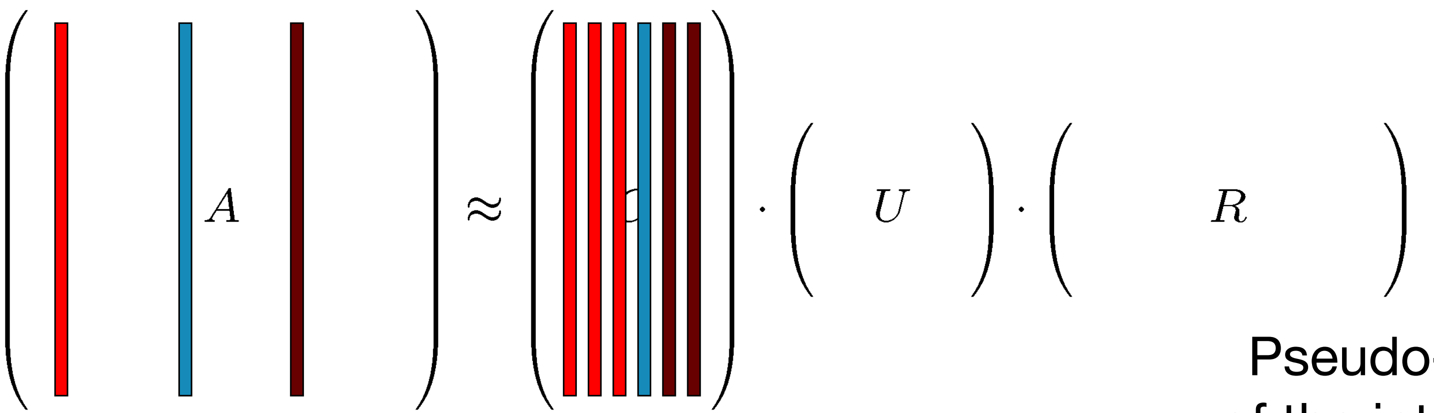


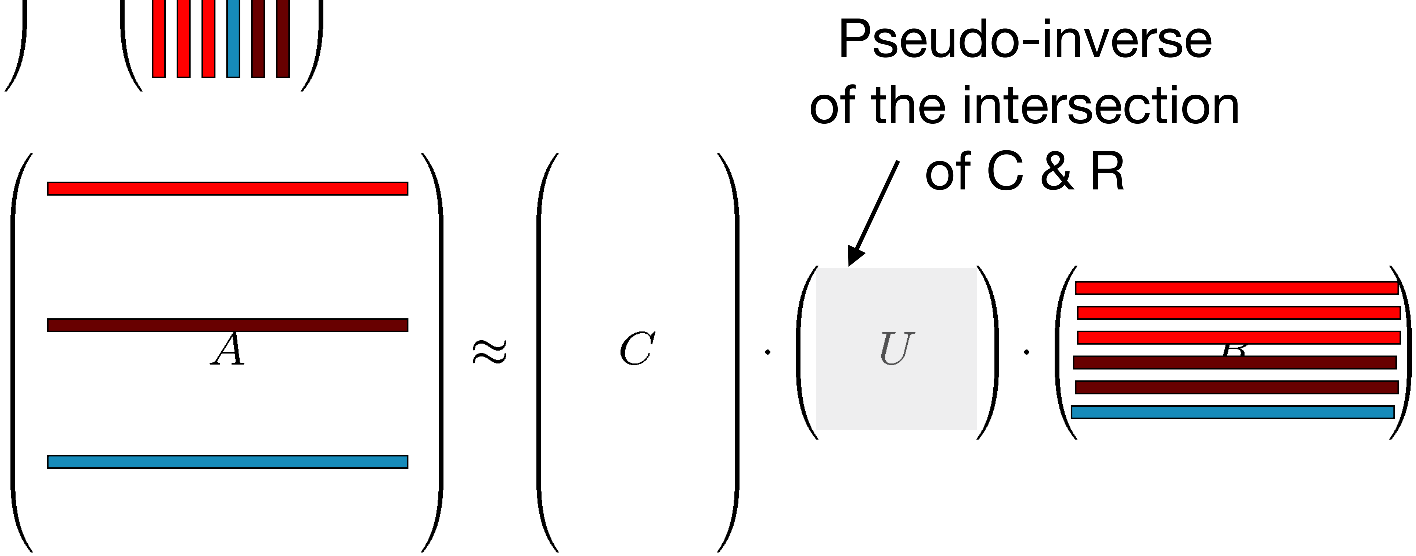
Outline

- Singular Value Decomposition
- CUR Decomposition
- Principal Components Analysis
- Factor Analysis

CUR Decomposition

- Goal: Express A as a product of matrices C,U,R and make $\|A-C\cdot U\cdot R\|_F$ small
- "Constraints" on C and R:





How CUR Works?

- To decrease the expected error between A and its decomposition, we must pick rows and columns in a nonuniform manner
- Sampling columns (similarly for rows):

Input: matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, sample size c
Output: $\mathbf{C}_d \in \mathbb{R}^{m \times c}$

1. for $x = 1 : n$ [column distribution]
2. $P(x) = \sum_i \mathbf{A}(i,x)^2 / \sum_{i,j} \mathbf{A}(i,j)^2$ ⟵ Prob. proportional to importance
3. for $i = 1 : c$ [sample columns]
4. Pick $j \in 1 : n$ based on distribution $P(x)$
5. Compute $\mathbf{C}_d(:,i) = \mathbf{A}(:,j)/\sqrt{cP(j)}$

Note this is a randomized algorithm, same column can be sampled more than once

Computing U

- Let W be the "intersection" of sampled columns C and rows R
    - Let SVD of $W = XZY^T$
- Then $U = W^+ = YZ^+X^T$ (Z^+: reciprocals of non-zero singular values $Z^+_{ii} = 1/ Z_{ii}$)
- Why pseudo inverse works?
    - $W = XZY^T$ then $W^{-1} = (Y^T)^{-1} Z^{-1} X^{-1}$
    - Due to orthonomality: $X^{-1}=X^T$ and $Y^{-1}=Y^T$
    - Since Z is diagonal $Z^{-1} = 1/Z_{ii}$
    - **Thus**, if **W** is nonsingular, pseudoinverse is the true inverse

CUR Pros & Cons

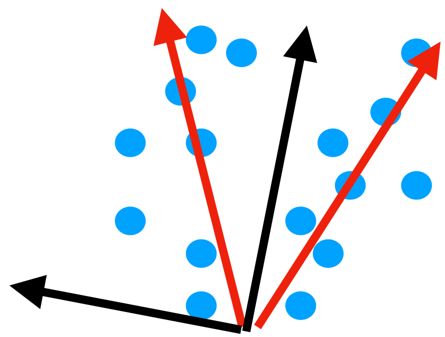
✓ Easy interpretation

✓ Sparse basis

- Basis vectors are actual columns and rows

– Duplicate columns and rows

- Columns of large norms will be sampled many times



SVD dimensions are orthogonal
CUR finds two clouds!

SVD vs. CUR

sparse and small

SVD: $A = U \Sigma V^T$

Huge but sparse   Big and dense

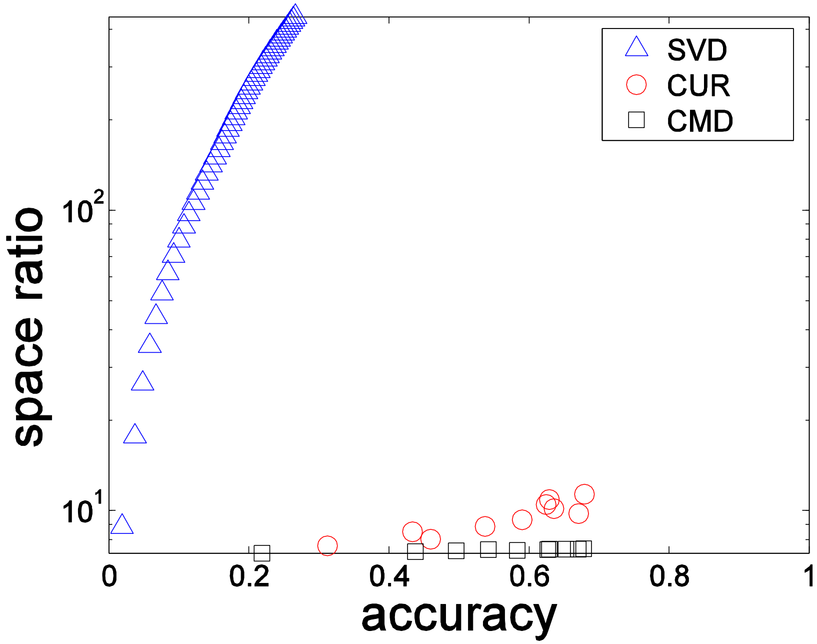
dense but small

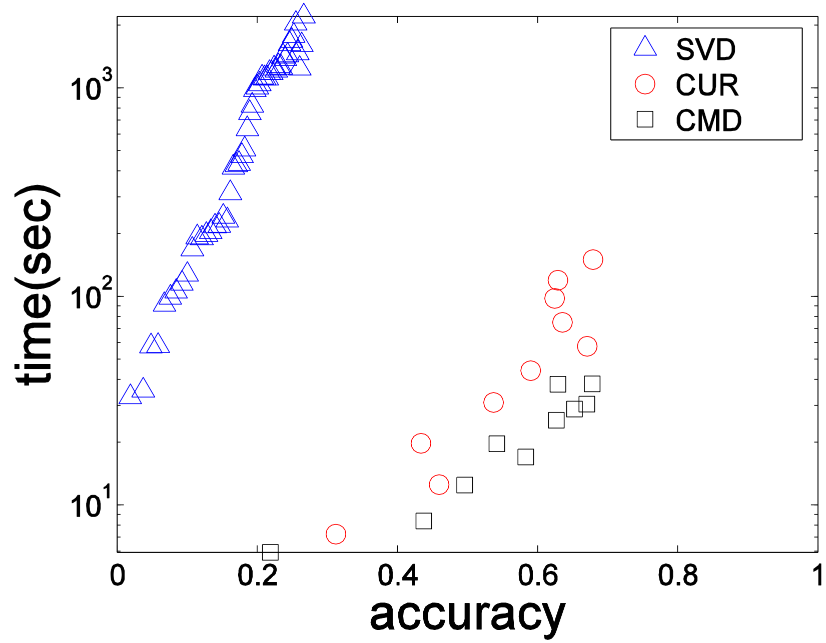
CUR: $A = C U R$

Huge but sparse   Big but sparse

SVD vs. CUR

- Reduce dimensionality for a sparse author-conference matrix: 428K authors (rows), 3659 conferences (columns)





- Accuracy = 1 – relative sum squared errors
- Space ratio = #output matrix entries / #input matrix entries
- CPU time

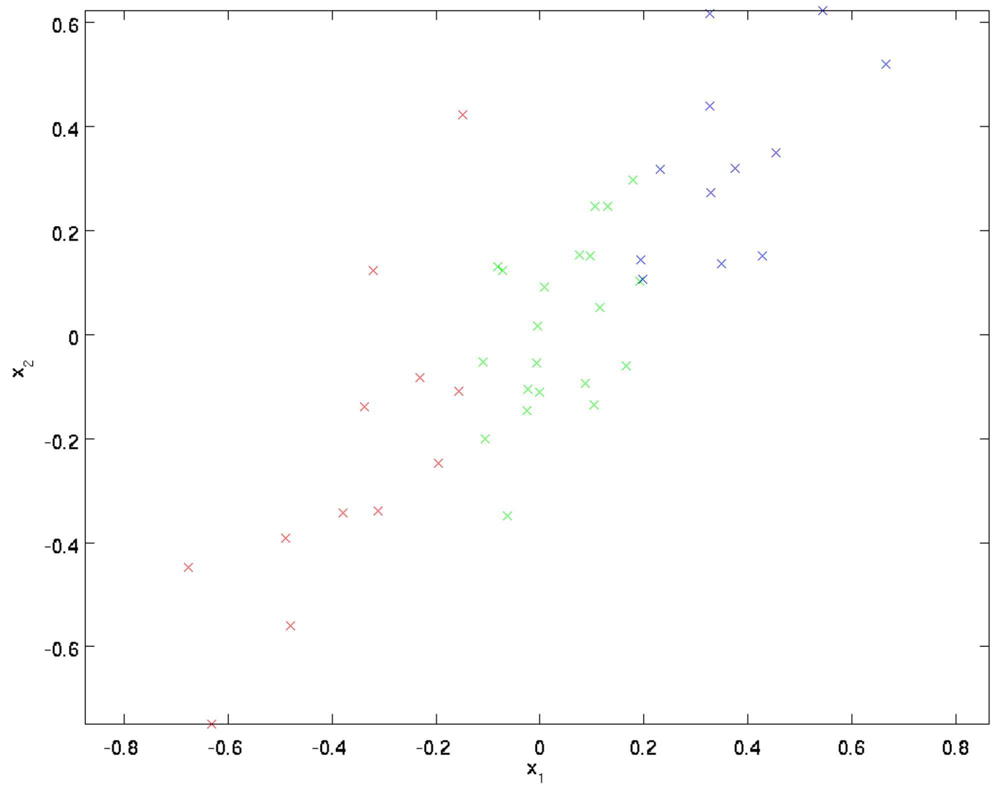
Sun, Faloutsos: Less is More: Compact Matrix Decomposition for Large Sparse Graphs, SDM '07.

Outline

- Singular Value Decomposition
- CUR Decomposition
- Principal Components Analysis
- Factor Analysis

SVD Application — PCA

- PCA only requires an eigenvector calculation
- Given a dataset of m points with n=2 dimensional inputs. Suppose we want to reduce the data dimension to n=1

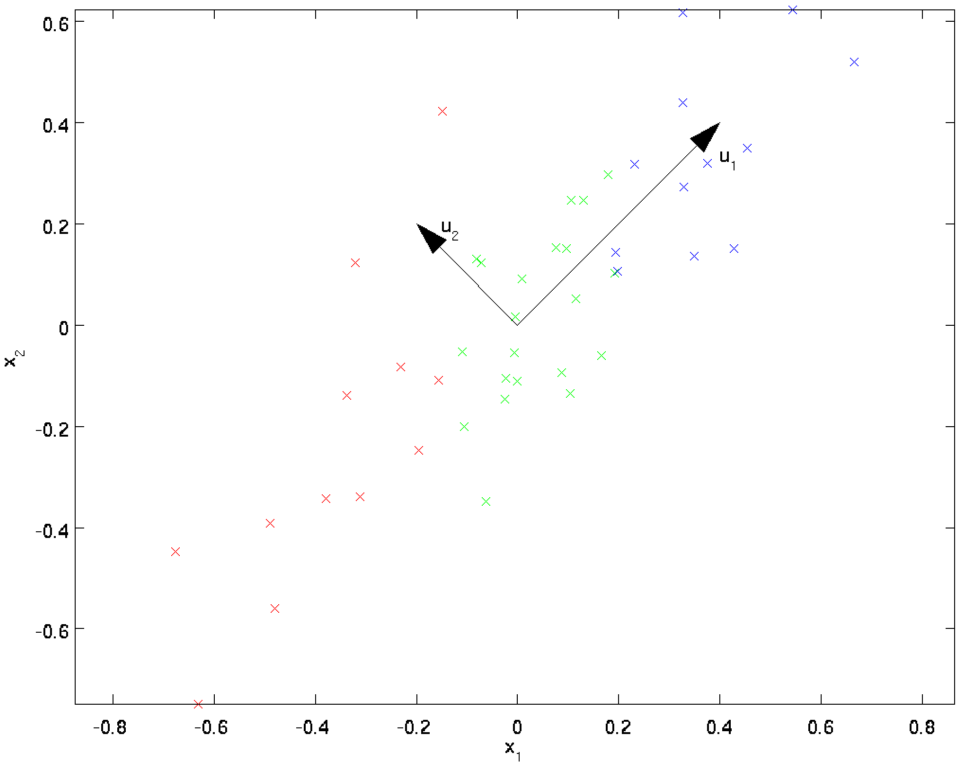


Principal Components Analysis

- Pre-process the data so that the features have the same mean (zero) and variance

1. Let $\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$.
2. Replace each $x^{(i)}$ with $x^{(i)} - \mu$.
3. Let $\sigma_j^2 = \frac{1}{m}\sum_i (x_j^{(i)})^2$
4. Replace each $x_j^{(i)}$ with $x_j^{(i)}/\sigma_j$.

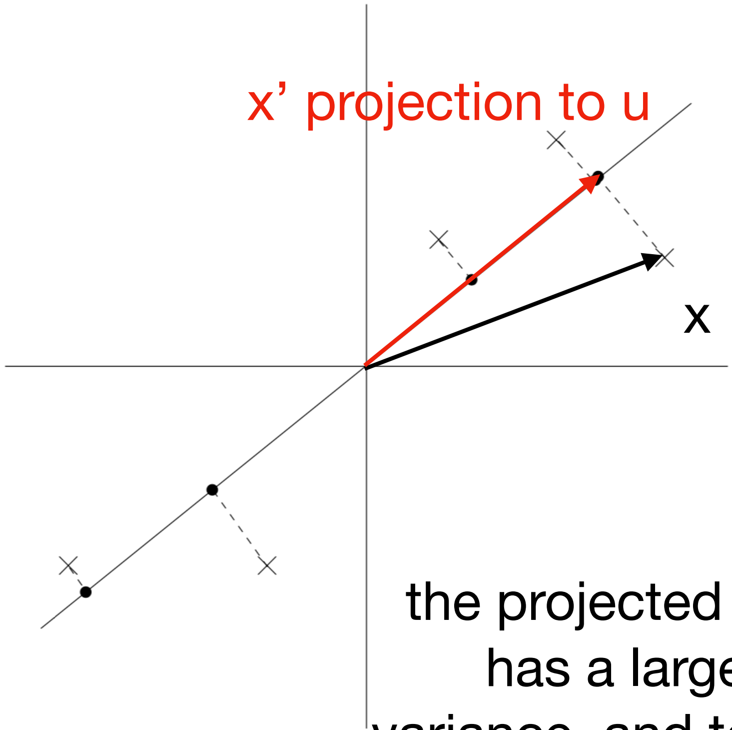
data vary more in the direction of u1 than u2



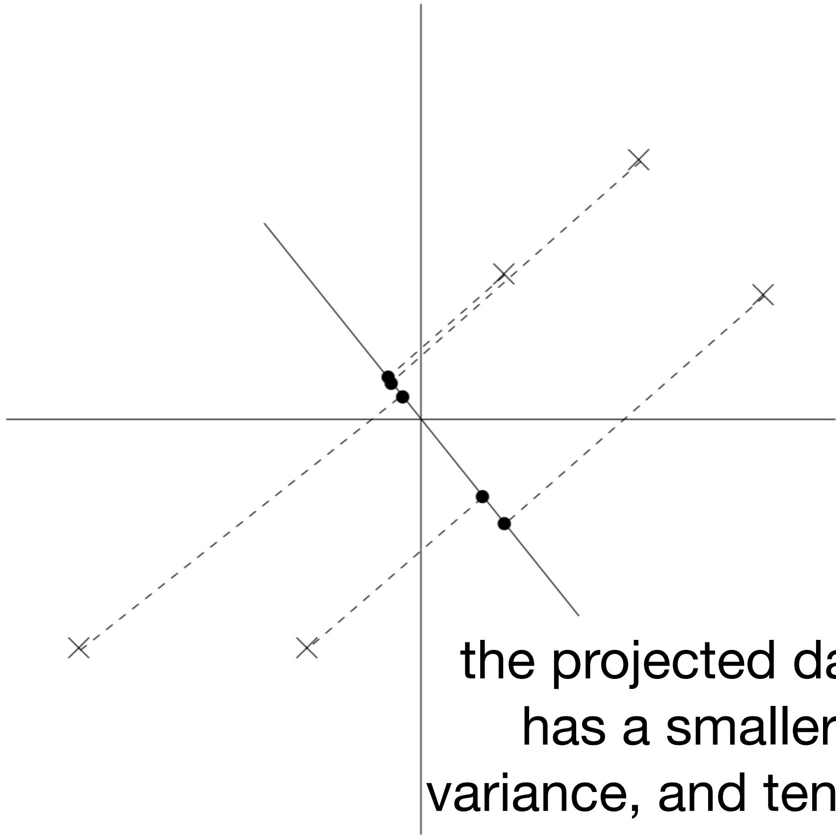
Renormalization rescales the different attributes to make them more comparable

Principal Components Analysis

- How to find the principal direction of variation of the data?
  - One way is to find the unit vector u such that the variance of the projected data to u is maximized



the projected data has a larger variance, and tend to be far from 0



the projected data has a smaller variance, and tend to be close to 0

Principal Components Analysis

- The length of the projection is given by $x^T u$. We would like to choose a unit-length u so as to maximize

$$\begin{aligned}\frac{1}{m}\sum_{i=1}^{m}({x^{(i)}}^T u)^2 &= \frac{1}{m}\sum_{i=1}^{m} u^T x^{(i)} {x^{(i)}}^T u \\ &= u^T\left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)} {x^{(i)}}^T\right) u\end{aligned}$$

- Maximizing the above subject to $||u||_2=1$ gives the principal eigenvector of $\Sigma = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} {x^{(i)}}^T$ $\quad \Sigma u = \lambda u \Leftrightarrow u^T\Sigma u = \lambda\, u^T u = \lambda$
- We have found a 1-dimensional subspace to approximate the data!

SVD & PCA

- SVD: picking basis that minimizes the approximation error arising from projecting the data onto the k-dimensional subspace spanned by them

- PCA: seeking the "major axis of variation" (the direction on which the data approximately lies)

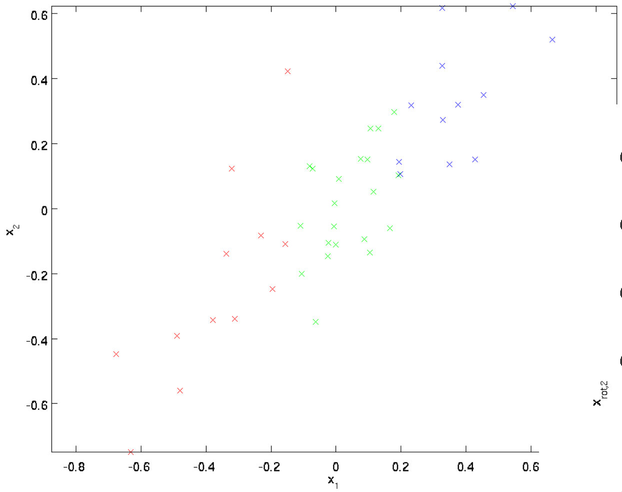
the same if the matrix is **zero-centered**

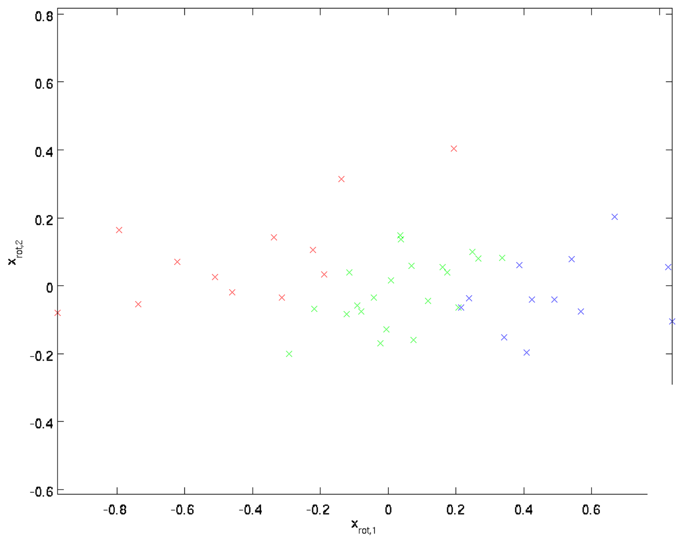
Find the best low rank approximation

Find how data varies in relationship to its mean

PCA — Another View

- Rotate the data



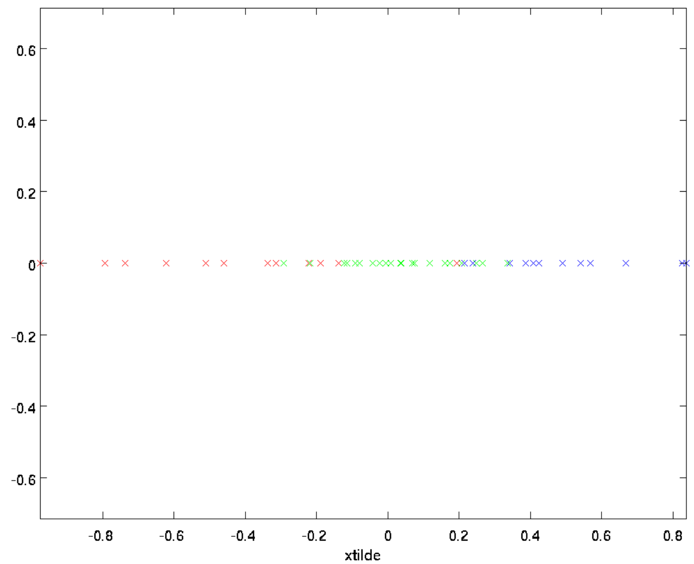


1-dimensional approximation

$$\tilde{x}^{(i)} = x_{\mathrm{rot},1}^{(i)} = u_1^T x^{(i)} \in \Re.$$

$$x_{\mathrm{rot}} = U^T x = \begin{bmatrix} u_1^T x \\ u_2^T x \end{bmatrix}$$

Training set rotated into the basis $u_1, u_2, \ldots, u_n$



PCA — Another View

- The principal direction of variation of the data is the first dimension $x_{rot,1}$ of this rotated data

$$\tilde{x}^{(i)} = x^{(i)}_{\mathrm{rot},1} = u_1^T x^{(i)} \in \Re.$$

- x_{rot} is an n dimensional vector, where the first few components are likely to be large, and the later components are likely to be small

- PCA drops the later components of x_{rot} and just approximates them with 0's

$$\tilde{x} = \begin{bmatrix} x_{\mathrm{rot},1} \\ \vdots \\ x_{\mathrm{rot},k} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \approx \begin{bmatrix} x_{\mathrm{rot},1} \\ \vdots \\ x_{\mathrm{rot},k} \\ x_{\mathrm{rot},k+1} \\ \vdots \\ x_{\mathrm{rot},n} \end{bmatrix} = x_{\mathrm{rot}}$$

Outline

- Singular Value Decomposition
- CUR Decomposition
- Principal Components Analysis
- Factor Analysis

Factor Analysis

- Consider a setting that the data dimension n >> number of training examples m
- It might be difficult to model the data as mixture of Gaussian. Why?
- If we model the data as Gaussian, and estimate the mean and covariance by

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

Σ is singular if m << n, cannot compute Σ^{-1}

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)(x^{(i)}-\mu)^T$$

- We may place some restrictions on Σ to obtain non-singular Σ
  - Fit a diagonal matrix Σ: $\Sigma_{jj} = \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)}-\mu_j)^2$

Factor Analysis Model

- Joint distribution on (x, z):

$$\begin{aligned} \underset{R^k}{z} &\sim \mathcal{N}(0, I) \\ x|z &\sim \mathcal{N}(\underset{R^n}{\mu} + \underset{R^{n\times k}}{\Lambda} z, \underset{R^{n\times n}}{\Psi}) \end{aligned}$$

- Each data point x is generated by sampling a k-dimensional multivariate Gaussian z, and map it to n-dimensional space (k < n)

- Factor analysis model:

$$\begin{aligned} z &\sim \mathcal{N}(0, I) \\ \epsilon &\sim \mathcal{N}(0, \Psi) \\ x &= \mu + \Lambda z + \epsilon \end{aligned}$$

z and ε are independent

- Consider joint Gaussian distribution $\begin{bmatrix} z \\ x \end{bmatrix} \sim \mathcal{N}(\mu_{zx}, \Sigma)$

Marginals and Conditionals of Gaussians

- x_1 and x_2 are jointly multivariate Gaussian
- $x = (x_1, x_2)^T$, suppose $x \sim \mathcal{N}(\mu, \Sigma)$ where $\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$
- What is the marginal distribution of x_1?

$$\mathrm{E}[x_1] = \mu_1$$

$$\mathrm{Cov}(x_1) = \mathrm{E}[(x_1 - \mu_1)(x_1 - \mu_1)] = \Sigma_{11}$$

$\Rightarrow$ $x_1 \sim \mathcal{N}(\mu_1, \Sigma_{11})$

- What is the conditional distribution of x_1 given x_2?

$$\begin{aligned} \mu_{1|2} &= \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2), \\ \Sigma_{1|2} &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}. \end{aligned}$$

$\Rightarrow$ $x_1|x_2 \sim \mathcal{N}(\mu_{1|2}, \Sigma_{1|2})$

Factor Analysis Model

- Consider joint Gaussian distribution $\begin{bmatrix} z \\ x \end{bmatrix} \sim \mathcal{N}(\mu_{zx}, \Sigma)$
- $\mathrm{E}[z] = 0$

$$\begin{aligned} \mathrm{E}[x] &= \mathrm{E}[\mu + \Lambda z + \epsilon] \\ &= \mu + \Lambda \mathrm{E}[z] + \mathrm{E}[\epsilon] \\ &= \mu. \end{aligned}$$

$$\mu_{zx} = \begin{bmatrix} \vec{0} \\ \mu \end{bmatrix}$$

- Joint Gaussian distribution $\begin{bmatrix} z \\ x \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \vec{0} \\ \mu \end{bmatrix}, \begin{bmatrix} I & \Lambda^T \\ \Lambda & \Lambda\Lambda^T + \Psi \end{bmatrix} \right)$
- Marginal distribution of x ~ $\mathcal{N}(\mu, \Lambda\Lambda^T + \Psi)$
  - log likelihood:

$$\ell(\mu, \Lambda, \Psi) = \log \prod_{i=1}^{m} \frac{1}{(2\pi)^{n/2} |\Lambda\Lambda^T + \Psi|^{1/2}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu)^T (\Lambda\Lambda^T + \Psi)^{-1} (x^{(i)} - \mu) \right)$$

EM for Factor Analysis

- Conditional distribution of a Gaussian

$$
\begin{aligned}
\mu_{z^{(i)}|x^{(i)}} &= \Lambda^T(\Lambda\Lambda^T+\Psi)^{-1}(x^{(i)}-\mu), \\
\Sigma_{z^{(i)}|x^{(i)}} &= I-\Lambda^T(\Lambda\Lambda^T+\Psi)^{-1}\Lambda.
\end{aligned}
$$

$$
Q_i(z^{(i)}) = \frac{1}{(2\pi)^{k/2}|\Sigma_{z^{(i)}|x^{(i)}}|^{1/2}} \exp\left(-\frac{1}{2}(z^{(i)}-\mu_{z^{(i)}|x^{(i)}})^T\Sigma^{-1}_{z^{(i)}|x^{(i)}}(z^{(i)}-\mu_{z^{(i)}|x^{(i)}})\right)
$$

- M-step: need to maximize w.r.t. μ, Λ, $\boldsymbol{\Psi}$: $\displaystyle\sum_{i=1}^m \int_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \mu, \Lambda, \Psi)}{Q_i(z^{(i)})} dz^{(i)}$

$$
\begin{aligned}
&\sum_{i=1}^m \int_{z^{(i)}} Q_i(z^{(i)}) \left[\log p(x^{(i)}|z^{(i)};\mu,\Lambda,\Psi) + \log p(z^{(i)}) - \log Q_i(z^{(i)})\right] dz^{(i)} \\
&= \sum_{i=1}^m \mathrm{E}_{z^{(i)}\sim Q_i}\left[\log p(x^{(i)}|z^{(i)};\mu,\Lambda,\Psi) + \log p(z^{(i)}) - \log Q_i(z^{(i)})\right]
\end{aligned}
$$

EM for Factor Analysis

- M-step (fitting Λ): need to maximize

$$\sum_{i=1}^{m} \mathrm{E}\left[\log p(x^{(i)}|z^{(i)};\mu,\Lambda,\Psi)\right]$$

$$= \sum_{i=1}^{m} \mathrm{E}\left[\log \frac{1}{(2\pi)^{n/2}|\Psi|^{1/2}} \exp\left(-\frac{1}{2}(x^{(i)}-\mu-\Lambda z^{(i)})^T \Psi^{-1}(x^{(i)}-\mu-\Lambda z^{(i)})\right)\right]$$

$$= \sum_{i=1}^{m} \mathrm{E}\left[-\frac{1}{2}\log|\Psi| - \frac{n}{2}\log(2\pi) - \frac{1}{2}(x^{(i)}-\mu-\Lambda z^{(i)})^T \Psi^{-1}(x^{(i)}-\mu-\Lambda z^{(i)})\right]$$

- Setting the derivative to 0 and we get:

$$\Lambda = \left(\sum_{i=1}^{m}(x^{(i)}-\mu)\mathrm{E}_{z^{(i)}\sim Q_i}\left[{z^{(i)}}^T\right]\right)\left(\sum_{i=1}^{m}\mathrm{E}_{z^{(i)}\sim Q_i}\left[z^{(i)}{z^{(i)}}^T\right]\right)^{-1}$$

Similar to normal equation for least squares regression:

$$\text{“}\theta^T = (y^T X)(X^T X)^{-1}.\text{”}$$

x’s are a linear function of z’s, we try to estimate the unknown linearity Λ relating the two

EM for Factor Analysis

- M-step (fitting Λ): work out the expectations

$$
\begin{aligned}
\mathrm{E}_{z^{(i)} \sim Q_i}\left[z^{(i)T}\right] &= \mu_{z^{(i)}|x^{(i)}}^T \\
\mathrm{E}_{z^{(i)} \sim Q_i}\left[z^{(i)} z^{(i)T}\right] &= \mu_{z^{(i)}|x^{(i)}} \mu_{z^{(i)}|x^{(i)}}^T + \Sigma_{z^{(i)}|x^{(i)}}
\end{aligned}
$$

Cov(Y) = E[YY^T] - E[Y]E[Y]^T

$$\Lambda = \left(\sum_{i=1}^m (x^{(i)} - \mu) \mu_{z^{(i)}|x^{(i)}}^T\right) \left(\sum_{i=1}^m \mu_{z^{(i)}|x^{(i)}} \mu_{z^{(i)}|x^{(i)}}^T + \Sigma_{z^{(i)}|x^{(i)}}\right)^{-1}$$

$$\Lambda = \left(\sum_{i=1}^m (x^{(i)} - \mu) \mathrm{E}_{z^{(i)} \sim Q_i}\left[z^{(i)T}\right]\right) \left(\sum_{i=1}^m \mathrm{E}_{z^{(i)} \sim Q_i}\left[z^{(i)} z^{(i)T}\right]\right)^{-1}$$

must take into account the covariance of z in p(z|x)!

- M-step (fitting μ)

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

not depend on p(z|x), only need to compute once!

Reference and Acknowledgement

- Jure Leskovec, Anand Raj, Jeff Ullman, “Mining of Massive Datasets,” Cambridge University Press, Chapter 11
- Andrew Ng’s CS229 lecture notes: http://cs229.stanford.edu/syllabus.html
- Unsupervised Feature Learning and Deep Learning: http://ufldl.stanford.edu/tutorial/