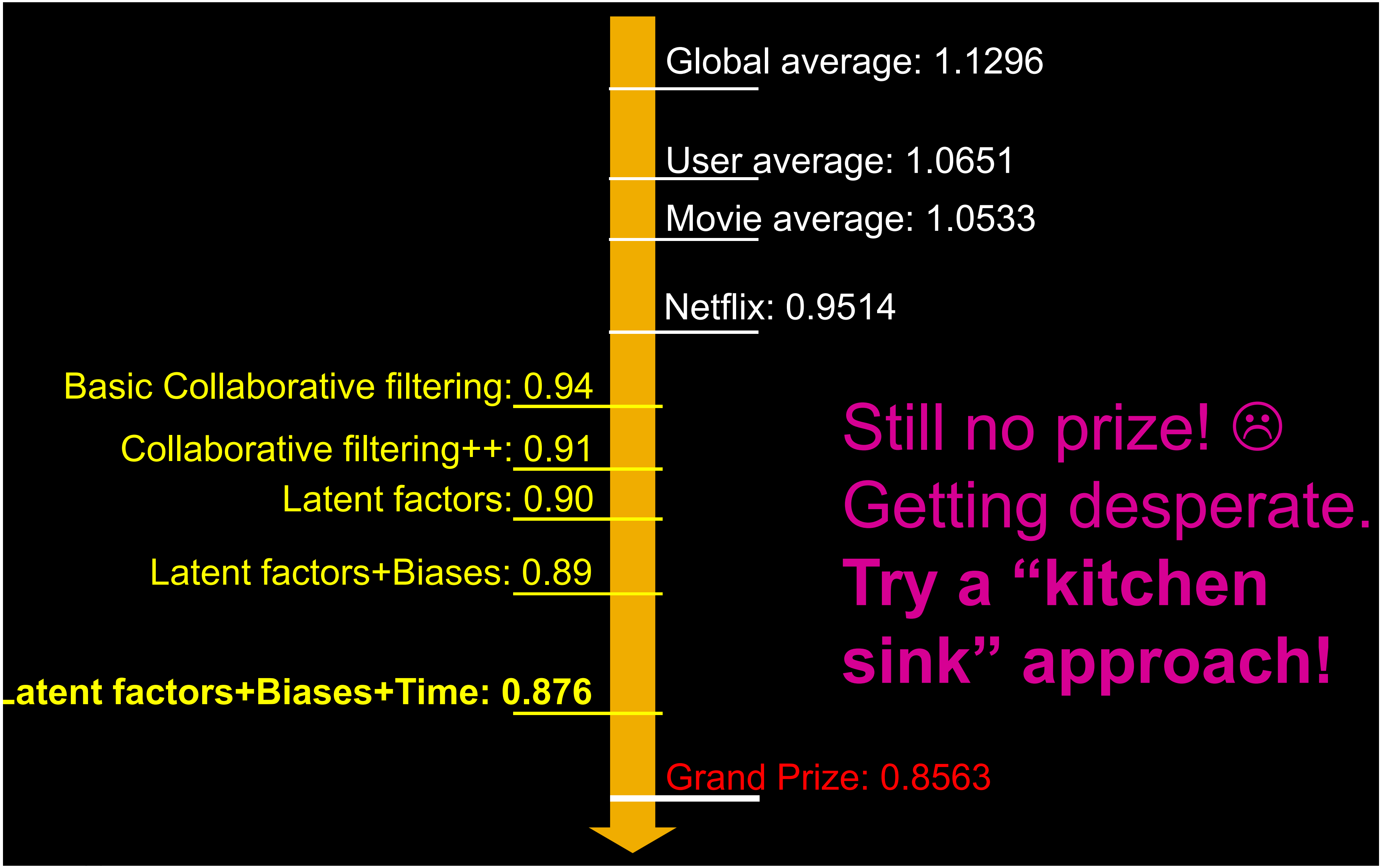
COMP4434 Big Data Analytics

Lecture 10 Collaborative Filtering & Dimensionality Reduction

HUANG Xiao

xiaohuang@comp.polyu.edu.hk



Global average: 1.1296

User average: 1.0651

Movie average: 1.0533

Netflix: 0.9514

Basic Collaborative filtering: 0.94

Collaborative filtering++: 0.91

Latent factors: 0.90

Latent factors+Biases: 0.89

Latent factors+Biases+Time: 0.876

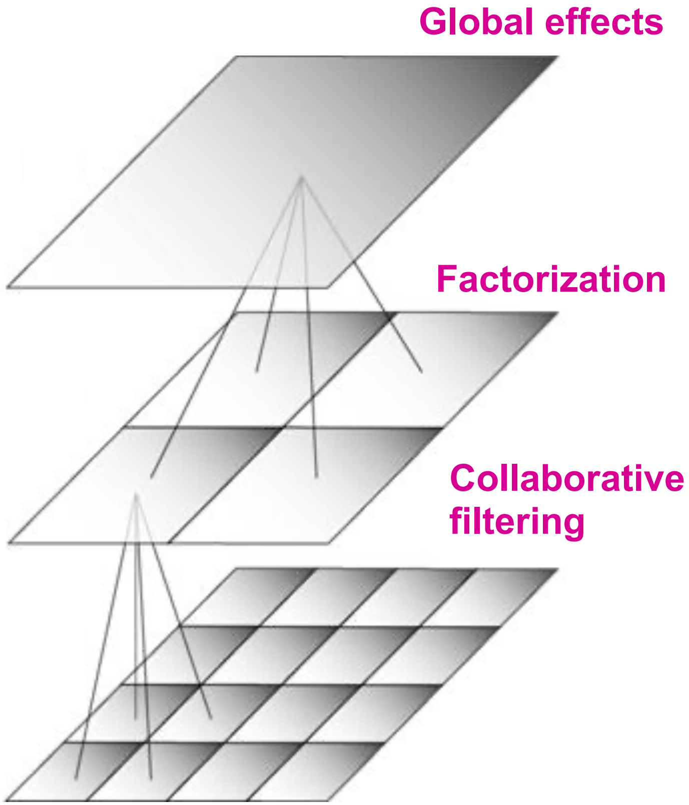
Grand Prize: 0.8563

Still no prize! ☹
Getting desperate.
Try a "kitchen sink" approach!

https://www.kaggle.com/netflix-inc/netflix-prize-data

BellKor Recommender System

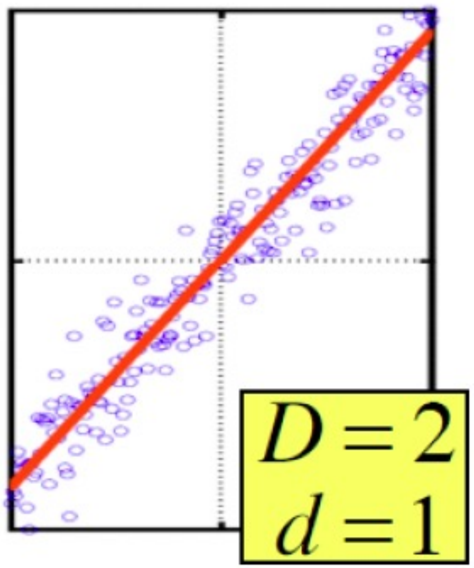
- **The winner of the Netflix Challenge!**
- **Multi-scale modeling of the data:** Combine top level, “regional” modeling of the data, with a refined, local view:
  - **Global:**
    - Overall deviations of users/movies
  - **Factorization:**
    - Addressing “regional” effects
  - **Collaborative filtering:**
    - Extract local patterns

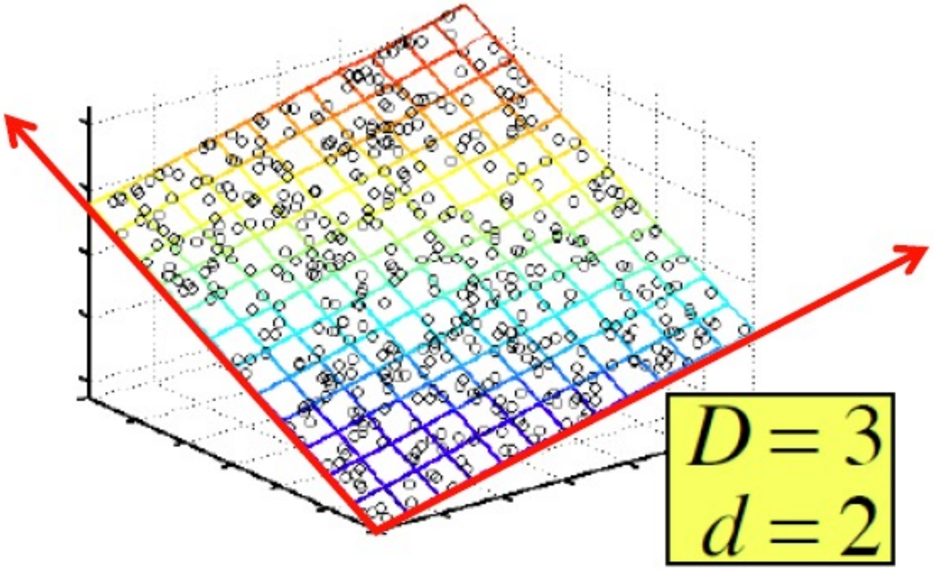


Problems with Error Measures

- **Narrow focus on accuracy sometimes misses the point**
  - Prediction Diversity
  - Prediction Context
  - Order of predictions
- **In practice, we care only to predict high ratings:**
  - RMSE might penalize a method that does well for high ratings and badly for others

Dimensionality Reduction

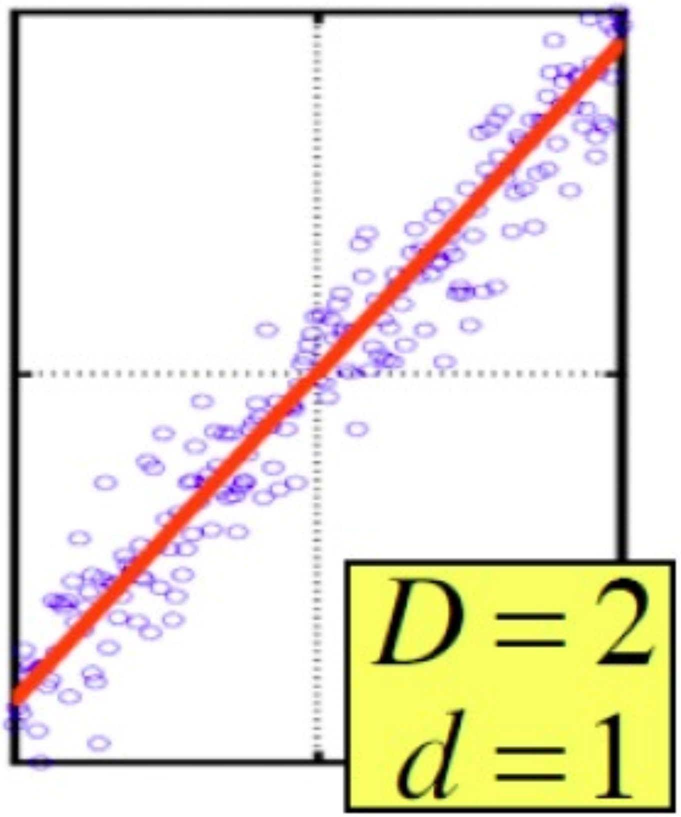




- **Assumption:** Data lies on or near a low *d*-dimensional subspace
- **Red axes of this subspace are effective representation of the data**

Dimensionality Reduction

- **Goal of dimensionality reduction is to discover the red axis of data!**

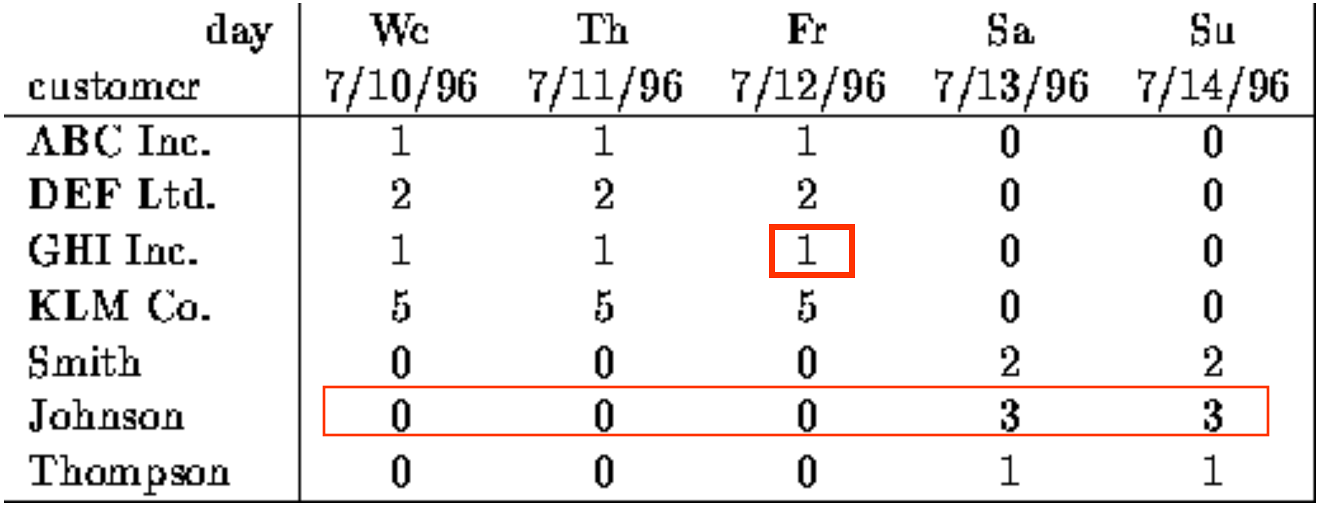


Rather than representing every point with 2 coordinates we represent each point with 1 coordinate (corresponding to the position of the point on the red line).

By doing this we incur a bit of **error** as the points do not exactly lie on the line

Dimensionality Reduction

- **Compress / reduce dimensionality:**
  - E.g.,



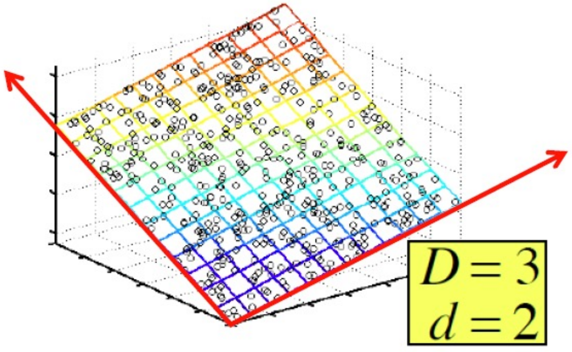
day	We	Th	Fr	Sa	Su
customer	7/10/96	7/11/96	7/12/96	7/13/96	7/14/96
ABC Inc.	1	1	1	0	0
DEF Ltd.	2	2	2	0	0
GHI Inc.	1	1	1	0	0
KLM Co.	5	5	5	0	0
Smith	0	0	0	2	2
Johnson	0	0	0	3	3
Thompson	0	0	0	1	1

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

  - 10^6 rows; 10^3 columns; no updates
  - Random access to any cell(s); **small error: OK**

Why Reduce Dimensions?

- **Data preprocessing is an important part for effective machine learning and data mining**
  - ML and DM techniques may not be effective for high-dimensional data
- **Dimensionality reduction is an effective approach to downsizing data**
  - The intrinsic dimension may be small
  - Discover hidden correlations/topics
    E.g., words that occur commonly together
  - Remove redundant and noisy features
    E.g., not all words are useful
- **Interpretation and visualization**
- **Easier storage and processing of the data**



Rank of a Matrix

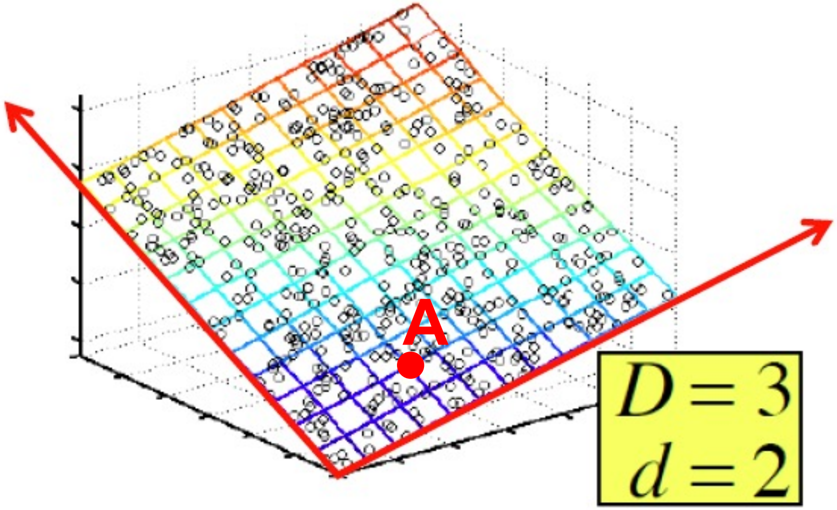
- What is **rank** of a matrix **A**?
  - Number of **linearly independent** columns of **A**
  - E.g., Matrix **A** = $\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$ has rank **r=2**
    - **Why?** The first two rows are linearly independent, so the rank is at least 2, but all 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.
- **Why do we care about low rank?**
  - We can write **A** as two "basis" vectors: [1 2 1] [-2 -3 1]
  - And new coordinates of : [1 0] [0 1] [1 -1]

Rank is “Dimensionality”

- **Cloud of points 3D space:**
  - Think of point positions as a matrix:

1 row per point:

$$\left[\begin{array}{rrr} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{array}\right] \begin{array}{l} A \\ B \\ C \end{array}$$

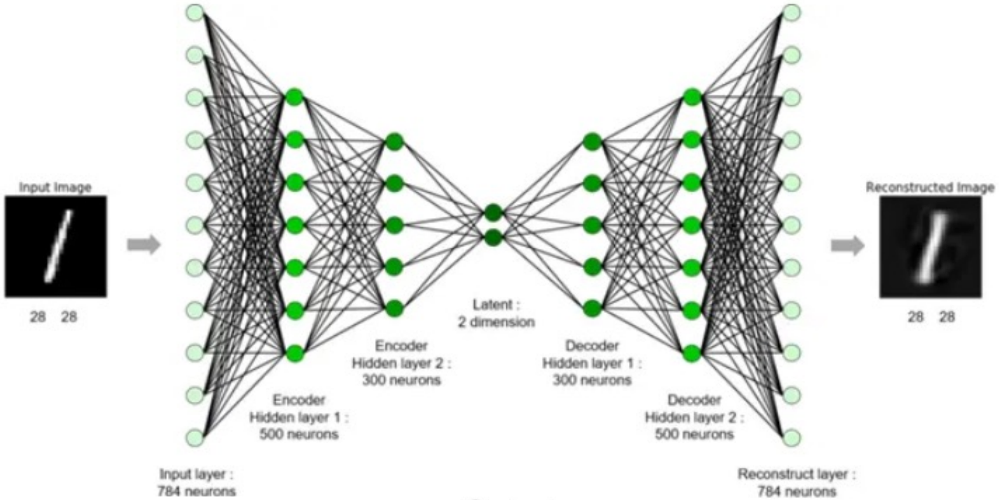


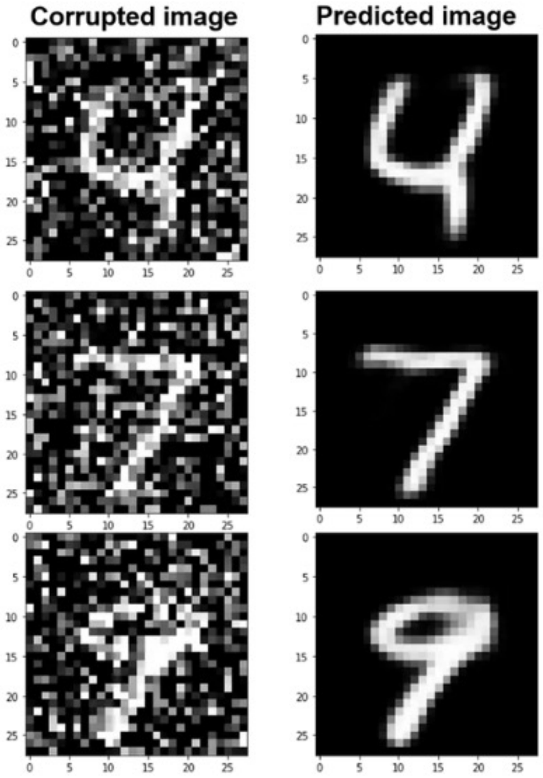
- **We can rewrite coordinates more efficiently!**
  - Old basis vectors: [1 0 0] [0 1 0] [0 0 1]
  - **New basis vectors: [1 2 1] [-2 -3 1]**
  - Then **A** has new coordinates: [1 0]. **B**: [0 1], **C**: [1 -1]
    - **Notice: We reduced the number of coordinates!**

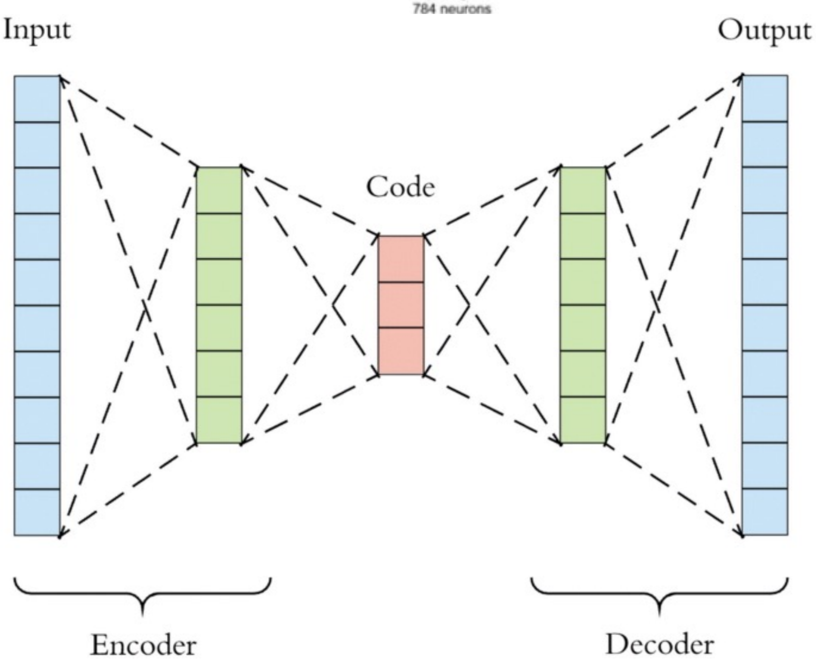
Dimensionality Reduction Techniques

- Singular value decomposition (SVD)
- Principal component analysis (PCA)
- Non-negative matrix factorization (NMF)
- Linear discriminant analysis (LDA)
- Autoencoders
- Feature selection

Dimensionality Reduction by using Hidden Layers


Input Image
28 28
Input layer : 784 neurons
Encoder Hidden layer 1 : 500 neurons
Encoder Hidden layer 2 : 300 neurons
Latent : 2 dimension
Decoder Hidden layer 1 : 300 neurons
Decoder Hidden layer 2 : 500 neurons
Reconstruct layer : 784 neurons
Reconstructed Image
28 28





In CF, feature learning is Dimensionality Reduction

Movie	Alice $\theta^{(1)}$	Bob $\theta^{(2)}$	Carol $\theta^{(3)}$	Dave $\theta^{(4)}$	X1	X2
Love letter $x^{(1)}$	5	5	0	0	?	?
Romancer $x^{(2)}$	5	?	?	0	?	?
Stay with me $x^{(3)}$	?	4	0	?	?	?
KungFu Panda $x^{(4)}$	0	0	5	4	?	?
FightFightFight $x^{(5)}$	0	0	5	?	?	?

- Hypothesis function

$$h_{i,j}(x,\theta) = \left(\theta^{(j)}\right)^T x^{(i)} = \theta_0^{(j)} x_0^{(i)} + \theta_1^{(j)} x_1^{(i)} + \theta_2^{(j)} x_2^{(i)} + \theta_3^{(j)} x_3^{(i)}$$

- In CF, the model learns feature X1 & X2.
- Represent each movie as its ratings & reduce the dimension from 4 (# users) to 2 (X1 & X2)

Can we learn latent factors directly?

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

Netflix data: $\mathbf{R} \approx Q \cdot P^T$

users

items

1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

R

≈

factors

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

Q

users

factors

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6

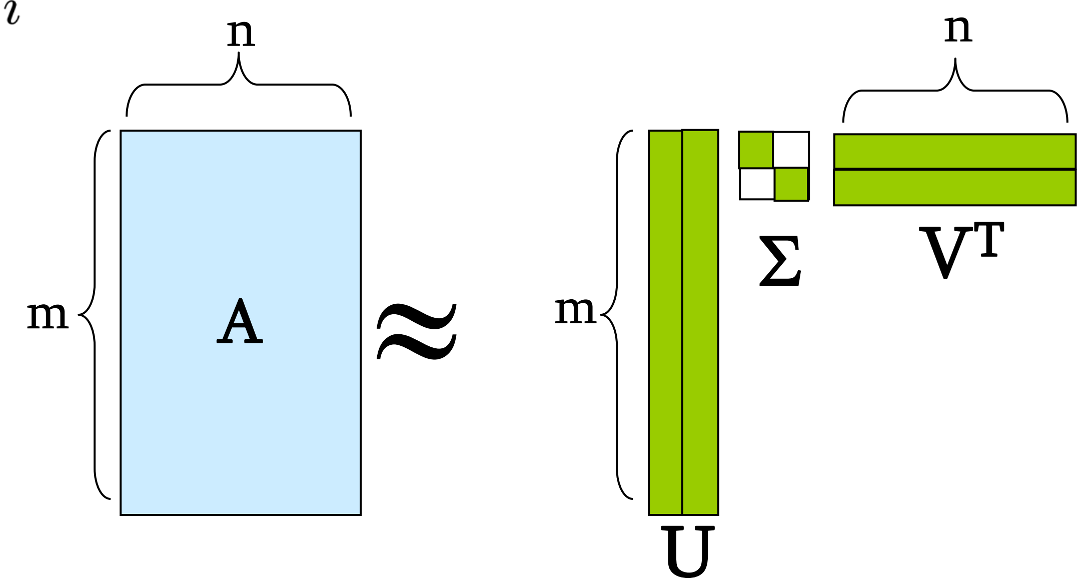
***P*ᵀ**

- Ratings can be recovered by latent factors (low-dimensional features)

$$h_{i,j}(x,\theta) = \left(\theta^{(j)}\right)^T x^{(i)} = \theta_0^{(j)} x_0^{(i)} + \theta_1^{(j)} x_1^{(i)} + \theta_2^{(j)} x_2^{(i)} + \theta_3^{(j)} x_3^{(i)}$$

Singular value decomposition (SVD) - Properties

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



It is **always** possible to decompose a real matrix ***A*** into ***A* = *U* Σ *V*^*T*** , where

- ***U*, Σ, *V***: unique
- ***U*, *V***: column orthonormal
  - ***U^T U* = *I*; *V^T V* = *I*** (***I***: identity matrix)
  - (Columns are orthogonal unit vectors)
- Σ: diagonal
  - Entries (**singular values**) are positive,
    and sorted in decreasing order (**$\sigma_1 \geq \sigma_2 \geq ... \geq 0$**)

Singular value decomposition (SVD) - Definition

$$\mathbf{A}_{[m \times n]} = \mathbf{U}_{[m \times r]} \Sigma_{[r \times r]} (\mathbf{V}_{[n \times r]})^T$$

- **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 – Example: Users-to-Movies

- $A = U\,\Sigma\,V^T$

- **U**: latent representation of movies
- **V**: latent representation of users

$$
\begin{array}{cccccc}
 & \text{Alice} & \text{Bob} & \text{Carol} & \text{Dave} & \vdots \\
\uparrow & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\text{SciFi} & \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
 & \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\downarrow & \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
\uparrow & 0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
\text{Romance} & 0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
\downarrow & 0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{array}
=
\begin{array}{ccc}
\text{SciFi-concept} & \text{Romance-concept} & \\
\downarrow & \swarrow & \\
\mathbf{0.14} & 0.02 & 0.01 \\
\mathbf{0.41} & 0.07 & 0.03 \\
\mathbf{0.55} & 0.09 & 0.04 \\
\mathbf{0.69} & 0.11 & 0.05 \\
0.15 & \mathbf{-0.59} & \mathbf{-0.65} \\
0.07 & \mathbf{-0.73} & \mathbf{0.68} \\
0.07 & \mathbf{-0.30} & \mathbf{-0.33}
\end{array}
\times
\begin{bmatrix}
\mathbf{12.5} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.13 & -0.03 & 0.13 & \mathbf{-0.70} & \mathbf{-0.70} \\
0.41 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

Columns are orthogonal unit vectors:
0.14^2 + 0.41^2 + 0.55^2 + 0.69^2 + 0.15^2 + 0.07^2 + 0.07^2 ≈ 1

Only Keep Major Factors

Set smallest singular values to zero

$$
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\approx
\begin{bmatrix}
\mathbf{0.14} & 0.02 & \cancel{-0.01} \\
\mathbf{0.41} & 0.07 & \cancel{-0.03} \\
\mathbf{0.55} & 0.09 & \cancel{-0.04} \\
\mathbf{0.69} & 0.11 & \cancel{-0.05} \\
0.15 & \mathbf{-0.59} & \cancel{\mathbf{0.65}} \\
0.07 & \mathbf{-0.73} & \cancel{\mathbf{-0.68}} \\
0.07 & \mathbf{-0.29} & \cancel{\mathbf{0.32}}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \cancel{\mathbf{1.3}}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.13 & -0.02 & 0.12 & \mathbf{-0.70} & \mathbf{-0.70} \\
\cancel{0.41} & \cancel{\mathbf{-0.80}} & \cancel{0.40} & \cancel{-0.09} & \cancel{-0.09}
\end{bmatrix}
$$

Dimensionality Reduction can reduce noise

$$\begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\ \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\ \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\ 0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\ 0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\ 0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2} \end{bmatrix} \approx \begin{bmatrix} \mathbf{0.14} & 0.02 \\ \mathbf{0.41} & 0.07 \\ \mathbf{0.55} & 0.09 \\ \mathbf{0.69} & 0.11 \\ 0.15 & \mathbf{-0.59} \\ 0.07 & \mathbf{-0.73} \\ 0.07 & \mathbf{-0.29} \end{bmatrix} \times \begin{bmatrix} \mathbf{12.4} & 0 \\ 0 & \mathbf{9.5} \end{bmatrix} \times \begin{bmatrix} \mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\ 0.13 & -0.02 & 0.12 & \mathbf{-0.70} & \mathbf{-0.70} \end{bmatrix}$$

$$\approx \begin{bmatrix} \mathbf{0.94} & \mathbf{1.01} & \mathbf{0.94} & -0.00 & -0.00 \\ \mathbf{2.98} & \mathbf{3.04} & \mathbf{2.98} & -0.00 & -0.00 \\ \mathbf{3.98} & \mathbf{4.05} & \mathbf{3.98} & -0.00 & -0.00 \\ \mathbf{4.97} & \mathbf{5.06} & \mathbf{4.97} & -0.01 & -0.01 \\ 0.36 & \mathbf{1.29} & 0.36 & \mathbf{4.08} & \mathbf{4.08} \\ -0.37 & 0.73 & -0.37 & \mathbf{4.92} & \mathbf{4.92} \\ 0.18 & \mathbf{0.65} & 0.18 & \mathbf{2.04} & \mathbf{2.04} \end{bmatrix}$$

Estimate unknown ratings as inner-products of factors

? = 2.4

users

items

1		3			5			5		4	
		5	4	?		4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

~

~

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

●

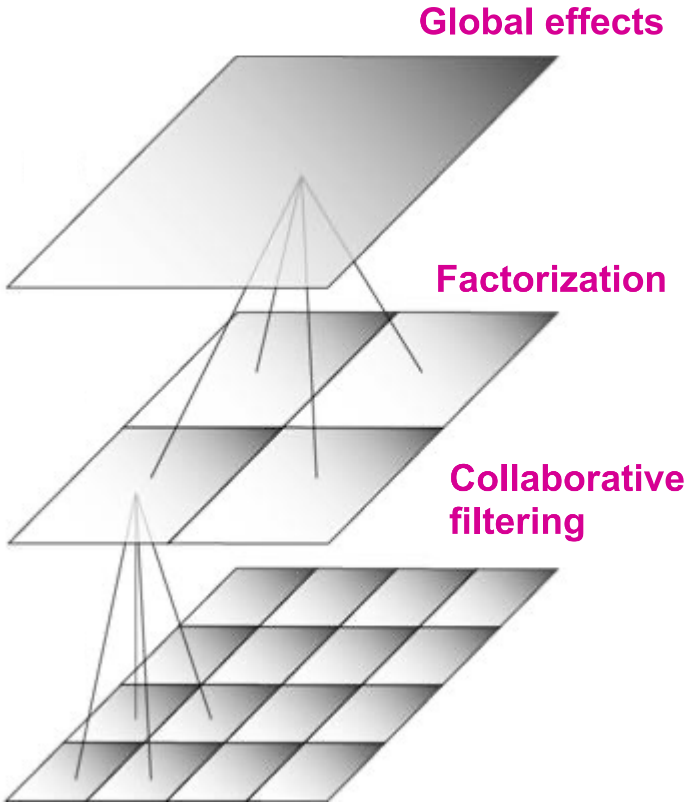
users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

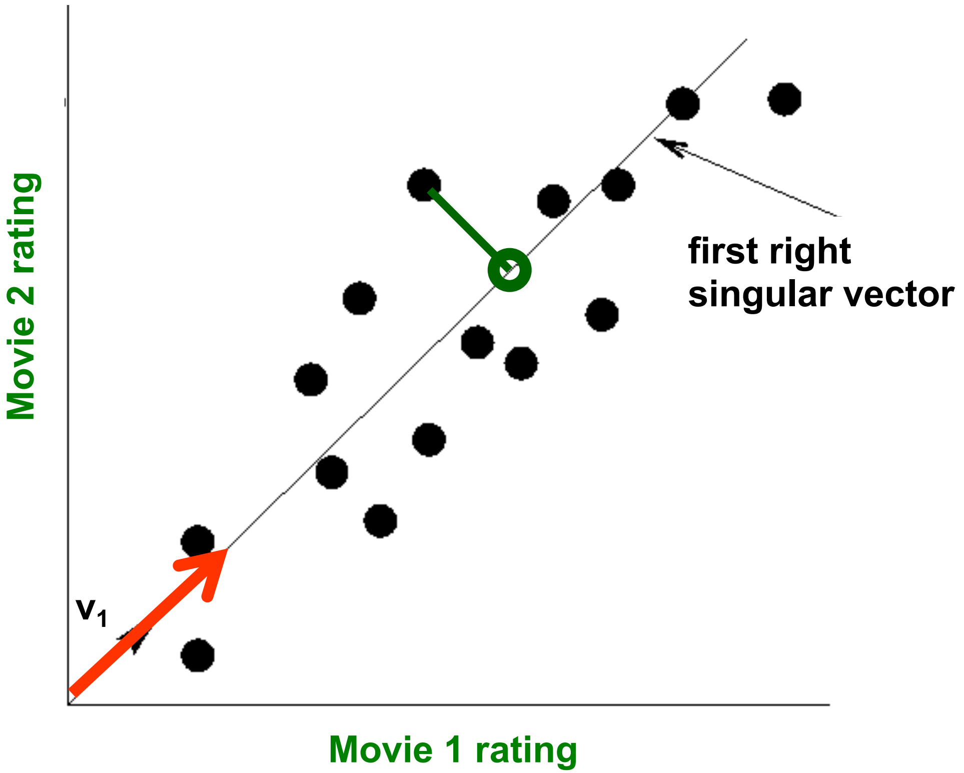
A rank-3 SVD approximation

Recap: BellKor Recommender System

- **The winner of the Netflix Challenge!**
- **Multi-scale modeling of the data:** Combine top level, "regional" modeling of the data, with a refined, local view:
  - **Global:**
    - Overall deviations of users/movies
  - **Factorization:**
    - Addressing "regional" effects
  - **Collaborative filtering:**
    - Extract local patterns



From 2D to 1D

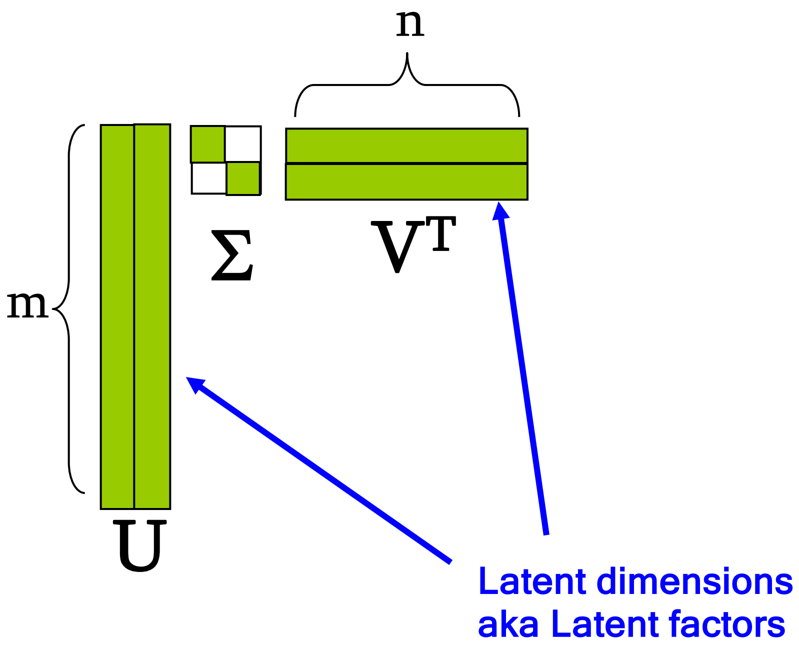


- Instead of using two coordinates (x, y) to describe point locations, let's use only one coordinate (z)
- Point's position is its location along vector v_1

Movie-to-User Example: from 5D to 3D

- $A = U \Sigma V^T$

$$\begin{array}{c|ccccc|}
 & \text{Alice} & \text{Bob} & \text{Carol} & \text{Dave} & \dots \\
\hline
\text{Matrix} & 1 & 1 & 1 & 0 & 0 \\
\text{Alien} & 3 & 3 & 3 & 0 & 0 \\
\text{Serenity} & 4 & 4 & 4 & 0 & 0 \\
\text{Casablanca} & 5 & 5 & 5 & 0 & 0 \\
\text{Amelie} & 0 & 2 & 0 & 4 & 4 \\
 & 0 & 0 & 0 & 5 & 5 \\
 & 0 & 1 & 0 & 2 & 2 \\
\hline
\end{array}
=$$



Latent dimensions
aka Latent factors

Meaning of Singular values

- The first feature is the most important one
- Singular values represent the importance of features

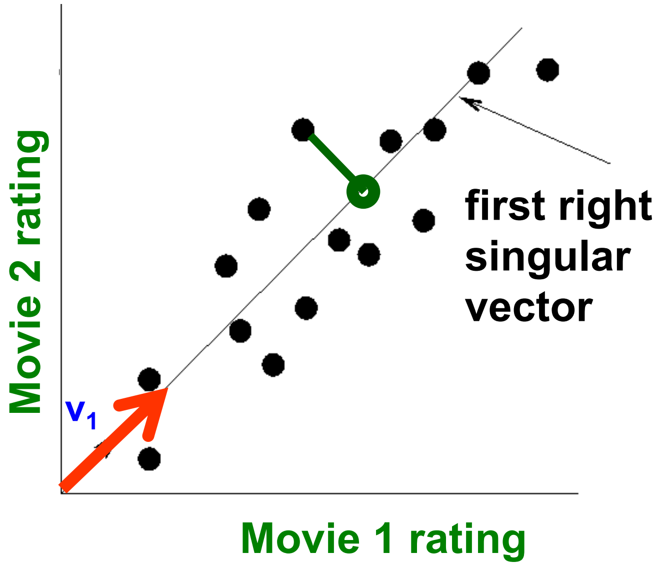
SciFi-concept

“strength” of the SciFi-concept

$$
\begin{array}{c|ccccc|}
 & \text{Alice} & \text{Bob} & \text{Carol} & \text{Dave} & \vdots \\
\hline
\uparrow & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
 & \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\text{SciFi} \downarrow & \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
 & \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
\uparrow & 0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
\text{Romnce} & 0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
\downarrow & 0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2} \\
\hline
\end{array}
=
\begin{bmatrix}
\mathbf{0.14} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.69} & 0.11 & -0.05 \\
0.15 & \mathbf{-0.59} & \mathbf{0.65} \\
0.07 & \mathbf{-0.73} & \mathbf{-0.68} \\
0.07 & \mathbf{-0.29} & \mathbf{0.33}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.13 & -0.02 & 0.13 & \mathbf{-0.70} & \mathbf{-0.70} \\
0.41 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

Singular values also represent the variance

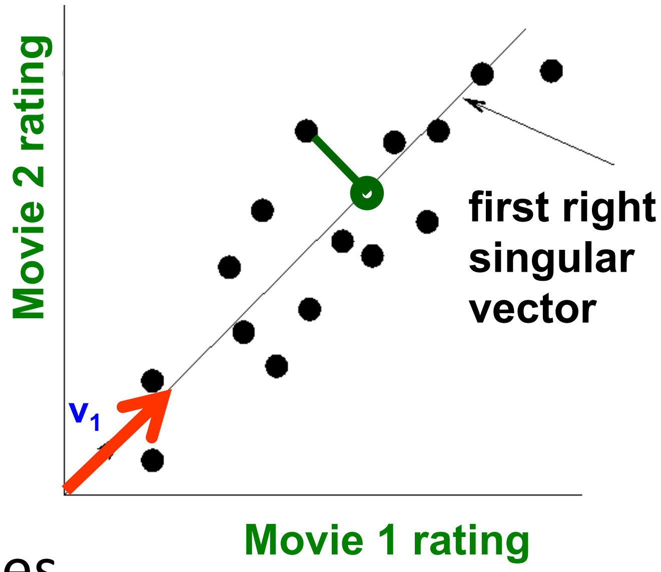
- **$A = U\,\Sigma\,V^T$ - example:**



variance ('spread') on the v_1 axis

$$
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
=
\begin{bmatrix}
\mathbf{0.14} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.69} & 0.11 & -0.05 \\
0.15 & \mathbf{-0.59} & \mathbf{0.65} \\
0.07 & \mathbf{-0.73} & \mathbf{-0.68} \\
0.07 & \mathbf{-0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.13 & -0.02 & 0.12 & \mathbf{-0.70} & \mathbf{-0.70} \\
0.41 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

SVD: minimizing reconstruction errors

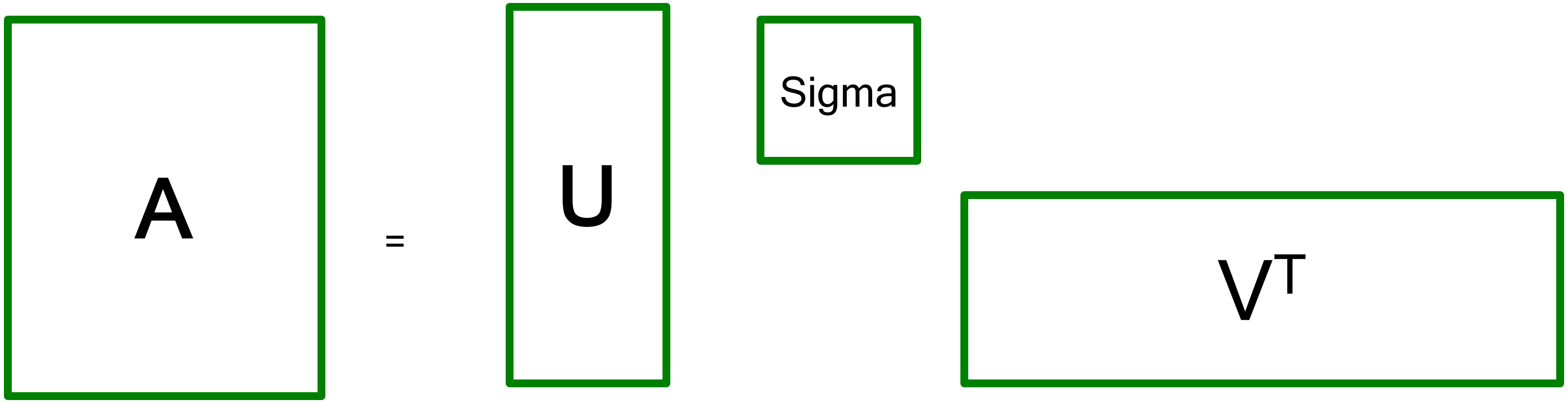


- How to choose v_1?
  Minimize reconstruction error
- Goal: Minimize the sum of reconstruction errors:

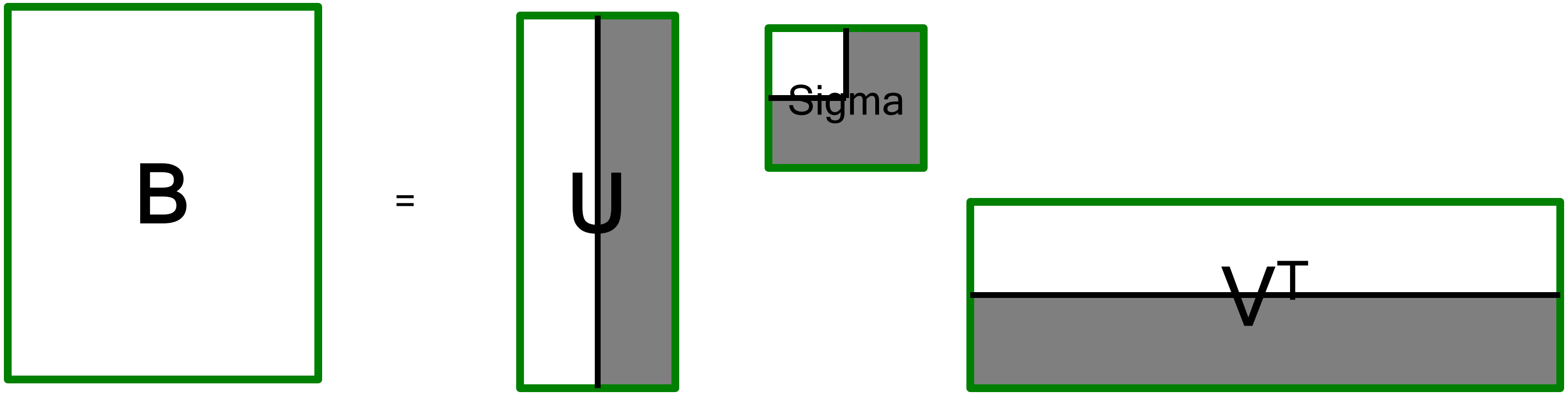
$$\sum_{i=1}^{N}\sum_{j=1}^{D}\left\|a_{ij}-b_{ij}\right\|^2$$

  - where $\boldsymbol{a}_{ij}$ are the original values in matrix **A** and $\boldsymbol{b}_{ij}$ are the reconstructed ones
    SVD gives 'best' axis to project on:
  - '**best**' = minimizing the reconstruction errors
- **In other words, minimum reconstruction error**

SVD – Best Low Rank Approx.



B is best approximation of A

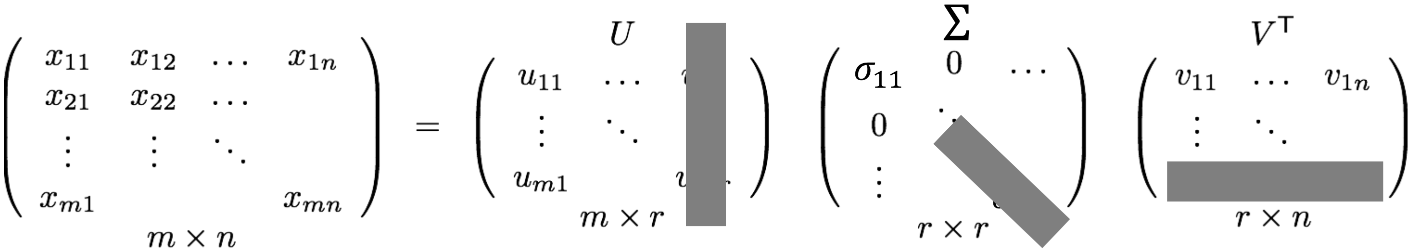


SVD – Best Low Rank Approx.

- **Theorem:**
  Let **A = U Σ V**^{T} and **B = U S V**^{T} where
  S = diagonal *rxr* matrix with ***s***$_i$**=*σ***$_i$ (*i=1…**k***) else ***s***$_i$**=0**
  then **B** is a **best** rank(**B**)=***k*** approx. to ***A***

What do we mean by "best":

- ***B*** **is a solution to min**$_B$ **‖*A*-*B*‖**$_F$ **where rank(*B*)=*k***



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

Users-to-Movies Example

- Best approximation for rank = 2
- $\|A-B\|_F = \sqrt{\Sigma_{ij} (A_{ij}-B_{ij})^2}$ is minimum

$$\begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\ \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\ \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\ 0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\ 0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\ 0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2} \end{bmatrix} \approx \begin{bmatrix} \mathbf{0.94} & \mathbf{1.01} & \mathbf{0.94} & -0.00 & -0.00 \\ \mathbf{2.98} & \mathbf{3.04} & \mathbf{2.98} & -0.00 & -0.00 \\ \mathbf{3.98} & \mathbf{4.05} & \mathbf{3.98} & -0.00 & -0.00 \\ \mathbf{4.97} & \mathbf{5.06} & \mathbf{4.97} & -0.01 & -0.01 \\ 0.36 & \mathbf{1.29} & 0.36 & \mathbf{4.08} & \mathbf{4.08} \\ -0.37 & 0.73 & -0.37 & \mathbf{4.92} & \mathbf{4.92} \\ 0.18 & \mathbf{0.65} & 0.18 & \mathbf{2.04} & \mathbf{2.04} \end{bmatrix}$$

Frobenius norm:

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

SVD - Complexity

- **A= U Σ V^T**: **unique**
- SVD: picks up linear correlations

- **To compute SVD:**
  - **O(nm^2)** or **O(n^{2}m)** (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 ...
  - numpy.linalg.svd in Python