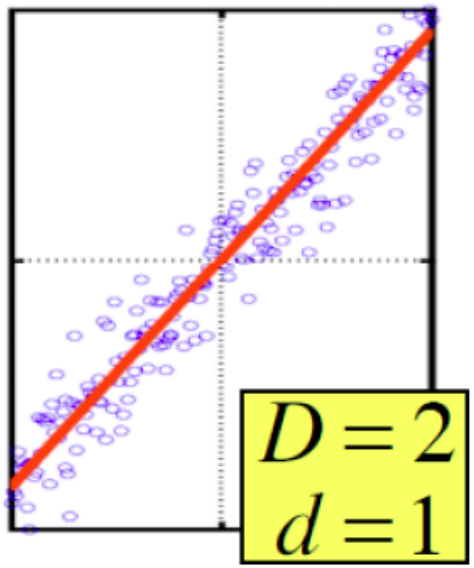
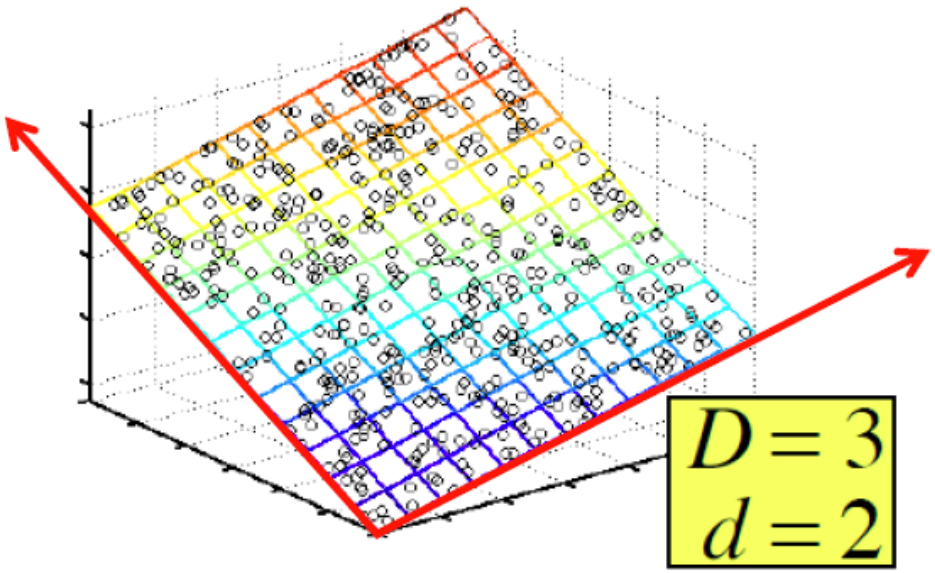
# CS60021: Scalable Data Mining

# Dimensionality Reduction

Sourangshu Bhattacharya

# Dimensionality Reduction




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

# Dimensionality Reduction

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

| 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 |

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]

# Rank of a Matrix

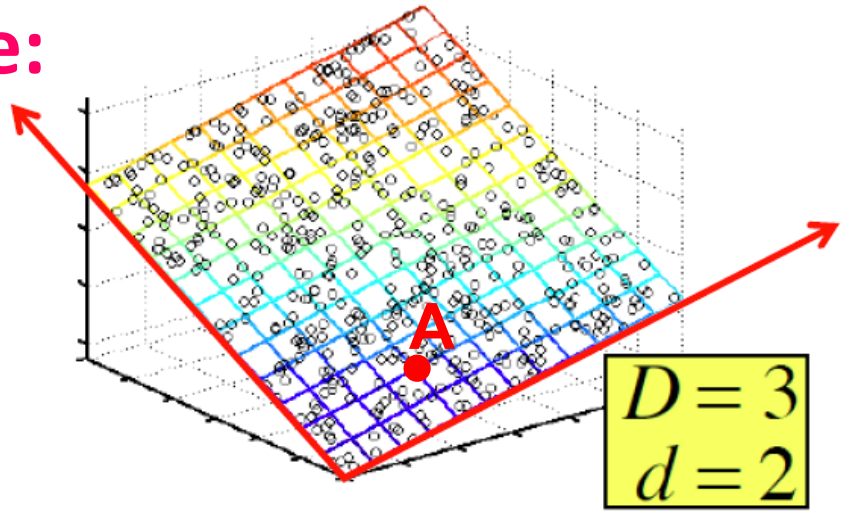
- **Q:** What is **rank** of a matrix **A**?
- **A:** Number of **linearly independent** columns of **A**
- **For example:**
  - Matrix **A** = $\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$ has rank **r=2**
    - **Why?** The first two rows are linearly independent, so the rank 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.
- **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:

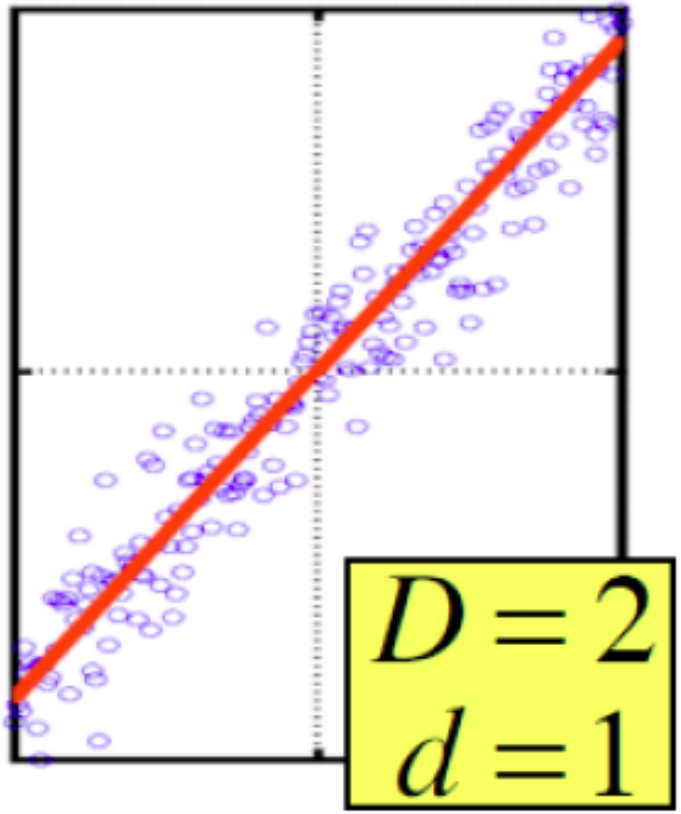
$$\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix} \begin{matrix} \text{A} \\ \text{B} \\ \text{C} \end{matrix}$$




- **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

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



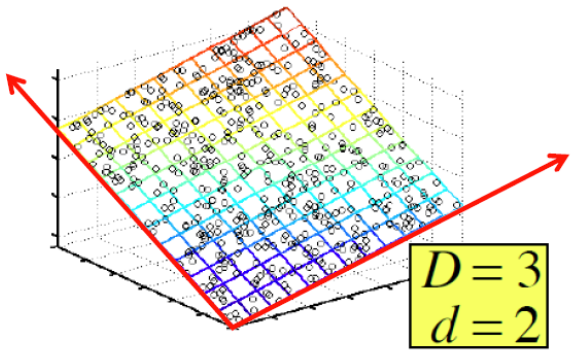

Rather than representing every point with 2 coordinates we represent each point with 1 coordinate (corresponding to the position of the point on the red line).

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

# Why Reduce Dimensions?

## 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**



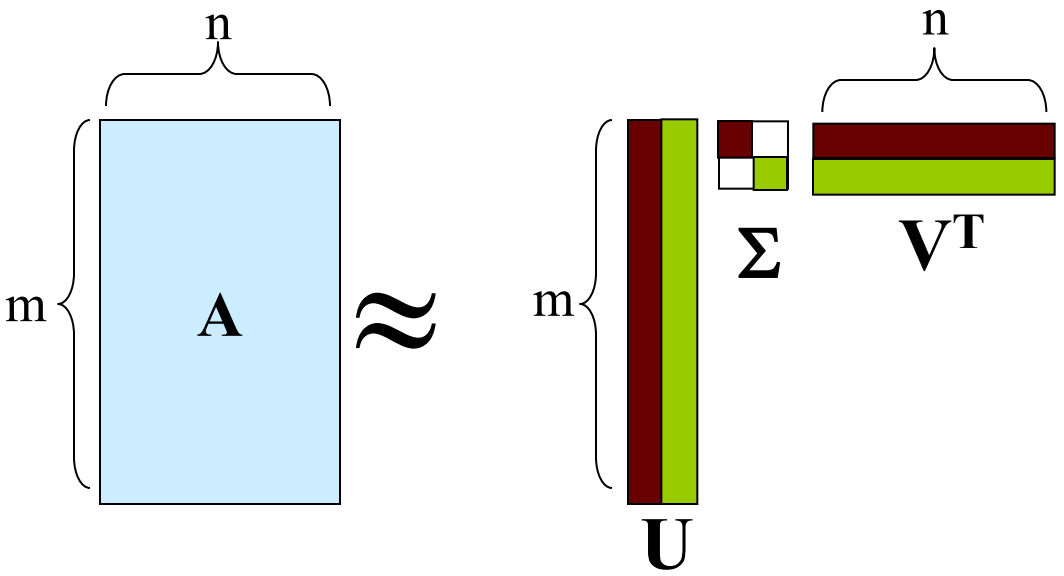
# SVD - Definition

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

- **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)

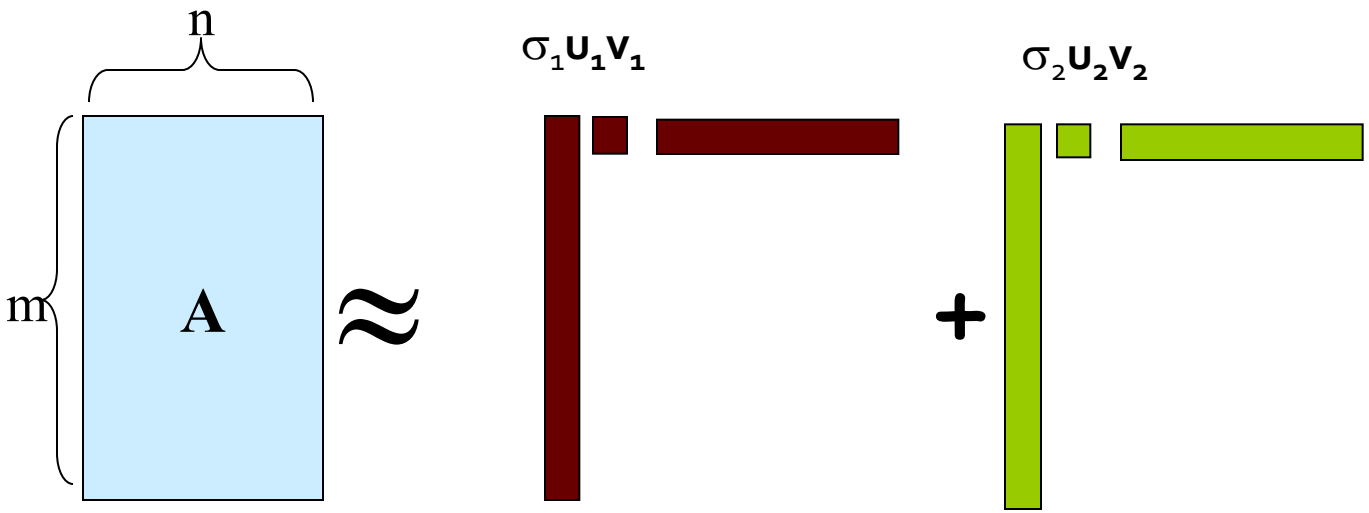
# SVD

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



# SVD

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




$\sigma_i$ … scalar
$\mathbf{u}_i$ … vector
$\mathbf{v}_i$ … vector

# SVD - Properties

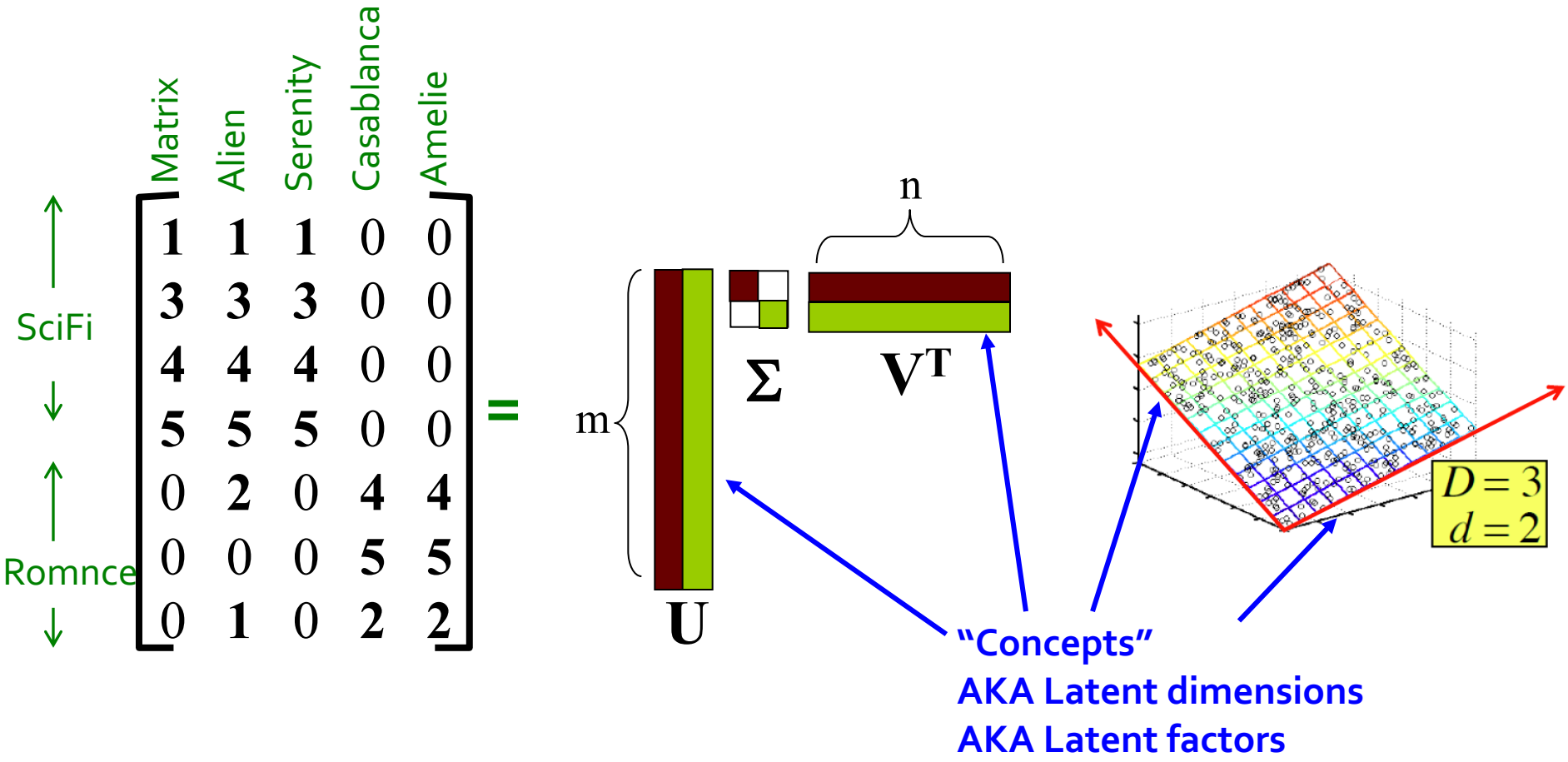
It is **always** possible to decompose a real matrix $A$ into $A = U \Sigma V^T$ , where

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

Nice proof of uniqueness: http://www.mpi-inf.mpg.de/~bast/ir-seminar-ws04/lecture2.pdf

# SVD – Example: Users-to-Movies

- **A = U Σ $V^T$ - example: Users to Movies**



# SVD – Example: Users-to-Movies

- **$A = U \Sigma V^T$ - example: Users to Movies**

| | Matrix | Alien | Serenity | Casablanca | Amelie |
|---|---|---|---|---|---|
| SciFi | 1 | 1 | 1 | 0 | 0 |
| | 3 | 3 | 3 | 0 | 0 |
| | 4 | 4 | 4 | 0 | 0 |
| | 5 | 5 | 5 | 0 | 0 |
| Romnce | 0 | 2 | 0 | 4 | 4 |
| | 0 | 0 | 0 | 5 | 5 |
| | 0 | 1 | 0 | 2 | 2 |

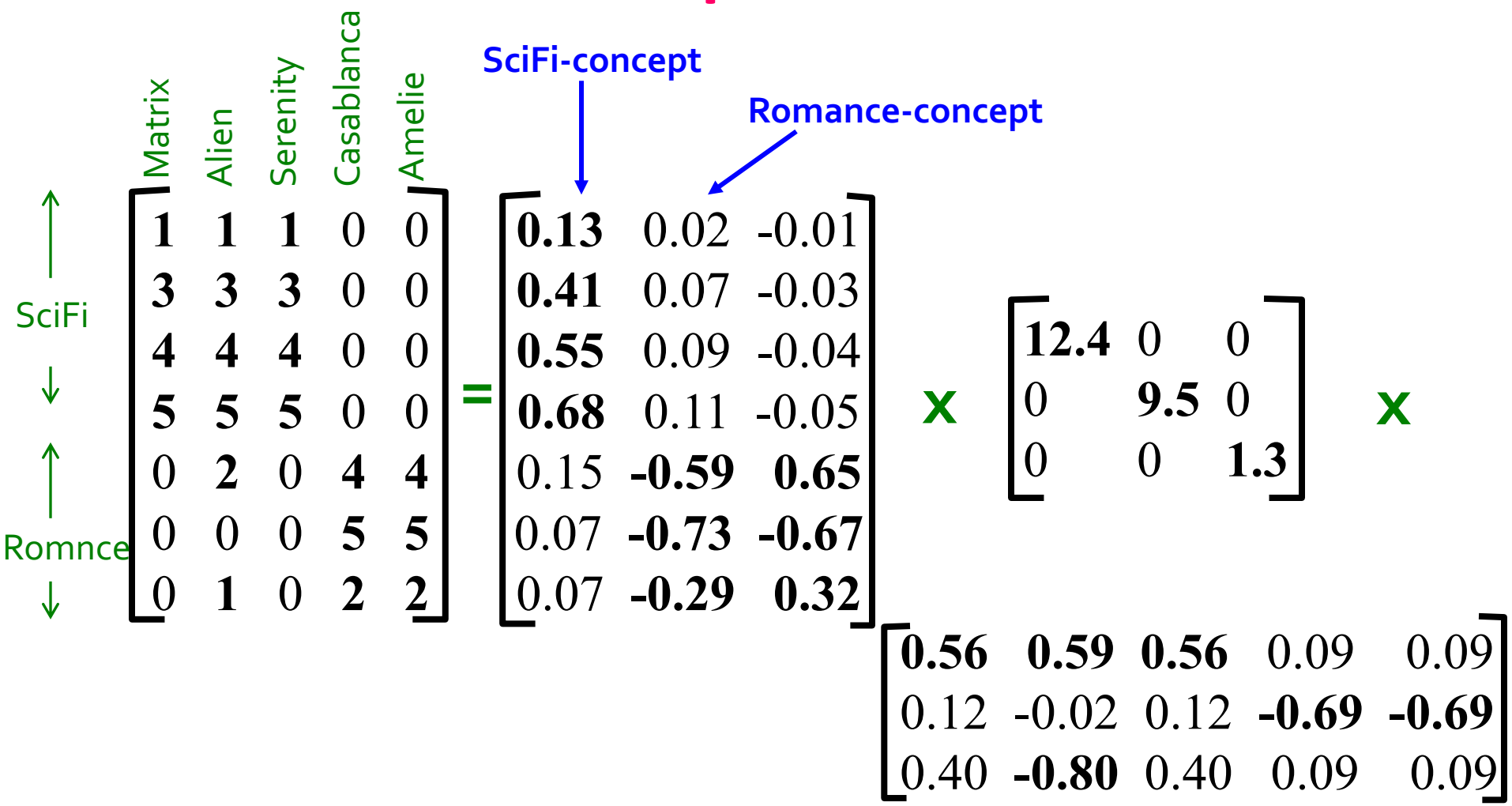
$=$

$$\begin{bmatrix} \mathbf{0.13} & 0.02 & -0.01 \\ \mathbf{0.41} & 0.07 & -0.03 \\ \mathbf{0.55} & 0.09 & -0.04 \\ \mathbf{0.68} & 0.11 & -0.05 \\ 0.15 & \mathbf{-0.59} & \mathbf{0.65} \\ 0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\ 0.07 & \mathbf{-0.29} & \mathbf{0.32} \end{bmatrix} \times \begin{bmatrix} \mathbf{12.4} & 0 & 0 \\ 0 & \mathbf{9.5} & 0 \\ 0 & 0 & \mathbf{1.3} \end{bmatrix} \times$$

$$\begin{bmatrix} \mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\ 0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

# SVD – Example: Users-to-Movies

- **A = U Σ $V^T$ - example: Users to Movies**



# SVD – Example: Users-to-Movies

- **A = U Σ $V^T$ - example:**

*U* is "user-to-concept" similarity matrix

| | Matrix | Alien | Serenity | Casablanca | Amelie |
|---|---|---|---|---|---|
| SciFi | 1 | 1 | 1 | 0 | 0 |
| | 3 | 3 | 3 | 0 | 0 |
| | 4 | 4 | 4 | 0 | 0 |
| | 5 | 5 | 5 | 0 | 0 |
| Romnce | 0 | 2 | 0 | 4 | 4 |
| | 0 | 0 | 0 | 5 | 5 |
| | 0 | 1 | 0 | 2 | 2 |

=

| SciFi-concept | Romance-concept | |
|---|---|---|
| **0.13** | 0.02 | -0.01 |
| **0.41** | 0.07 | -0.03 |
| **0.55** | 0.09 | -0.04 |
| **0.68** | 0.11 | -0.05 |
| 0.15 | **-0.59** | **0.65** |
| 0.07 | **-0.73** | **-0.67** |
| 0.07 | **-0.29** | **0.32** |

x

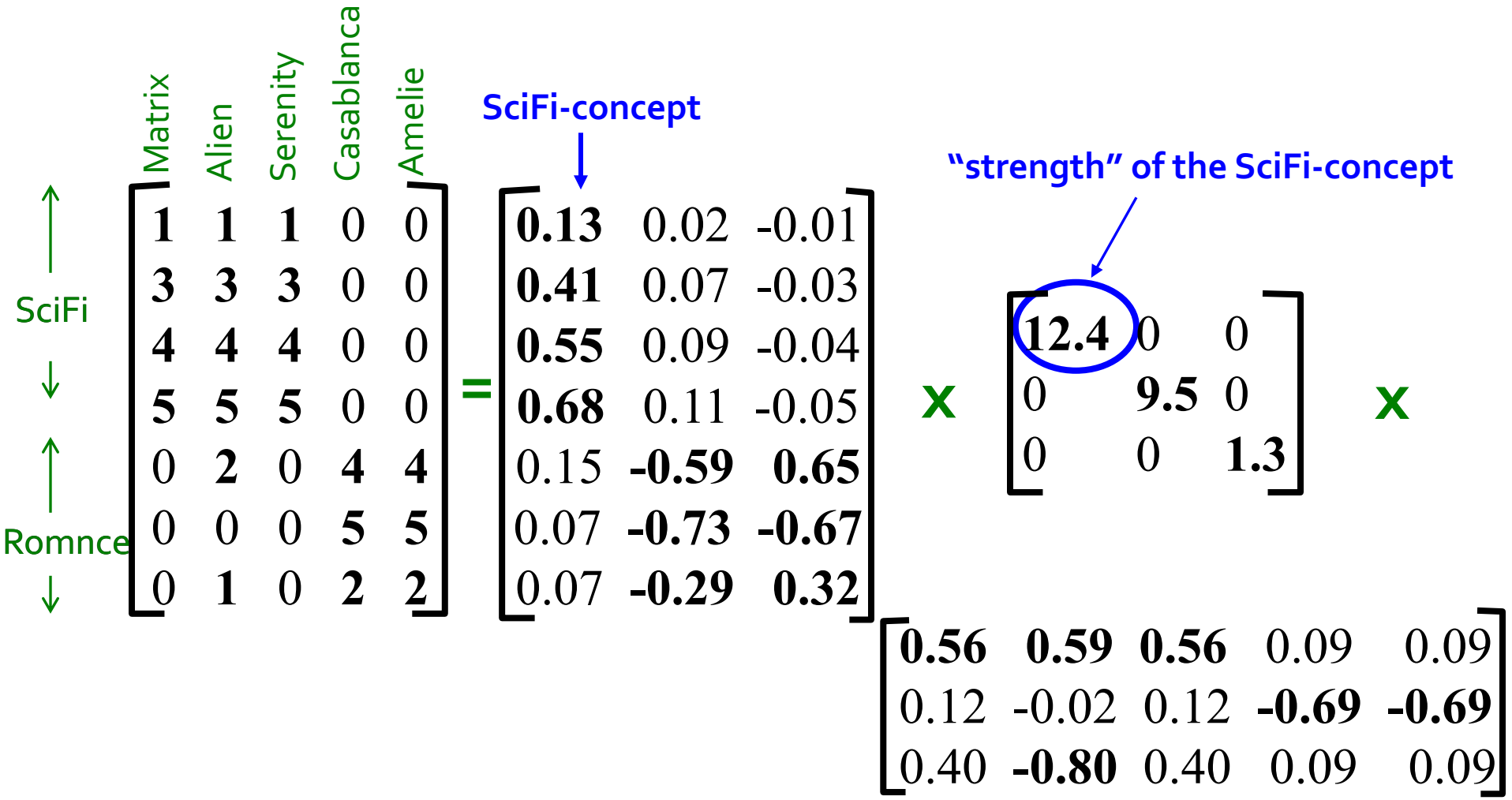
| | | |
|---|---|---|
| **12.4** | 0 | 0 |
| 0 | **9.5** | 0 |
| 0 | 0 | **1.3** |

x

| | | | | |
|---|---|---|---|---|
| **0.56** | **0.59** | **0.56** | 0.09 | 0.09 |
| 0.12 | -0.02 | 0.12 | **-0.69** | **-0.69** |
| 0.40 | **-0.80** | 0.40 | 0.09 | 0.09 |

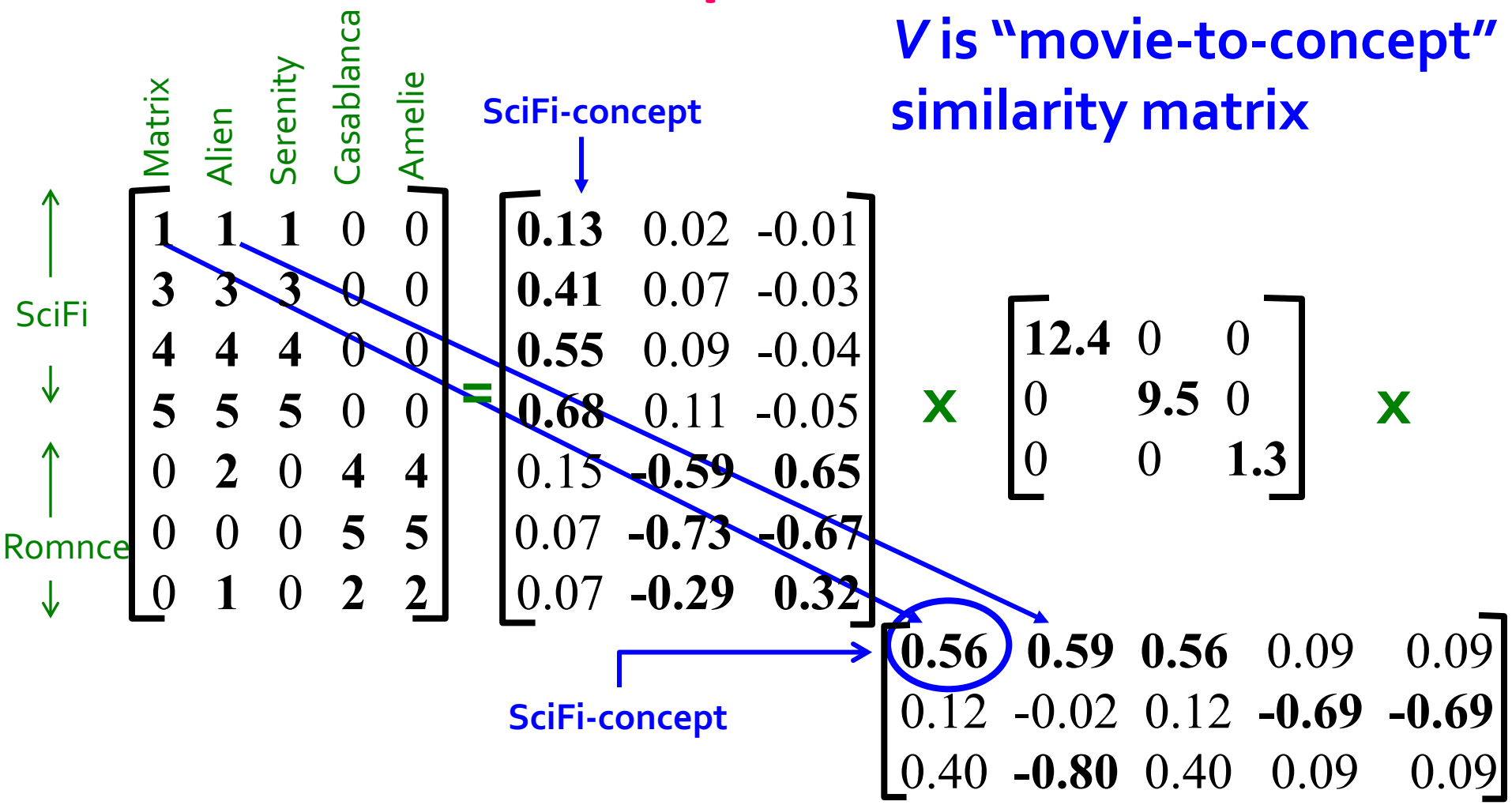
# SVD – Example: Users-to-Movies

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



# SVD – Example: Users-to-Movies

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



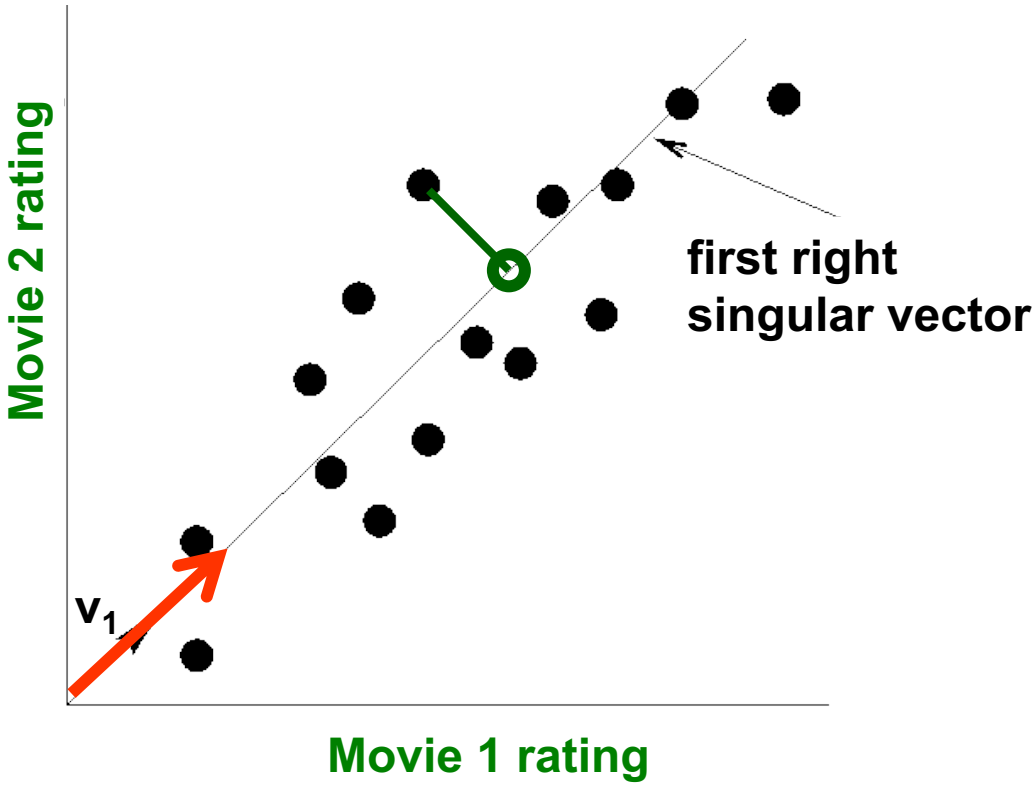
# SVD - Interpretation #1

'**movies**', '**users**' and '**concepts**':

- ***U***: user-to-concept similarity matrix
- ***V***: movie-to-concept similarity matrix
- $\Sigma$: its diagonal elements:
  'strength' of each concept

# Dimensionality Reduction with SVD

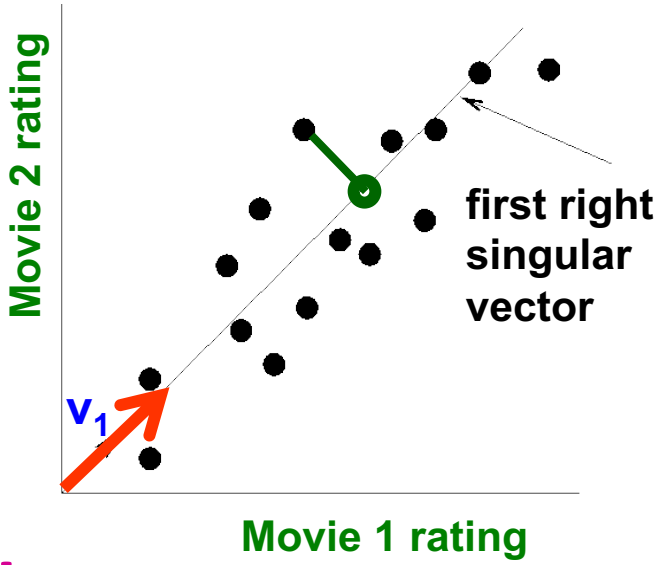
# SVD – Dimensionality Reduction




- 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 $v_1$
- **How to choose $v_1$? Minimize reconstruction error**

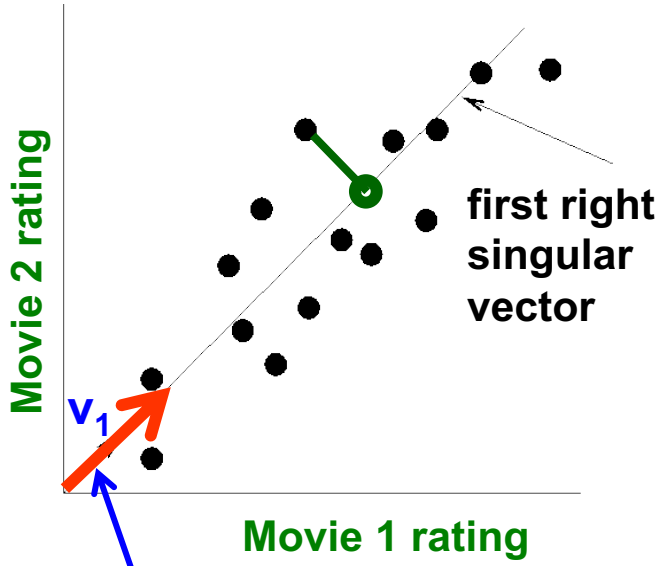
# SVD – Dimensionality Reduction

- **Goal: Minimize the sum of reconstruction errors:**
  - where  are the "old" and  are the "new" coordinates
- **SVD gives 'best' axis to project on:**
  - '**best**' = minimizing the reconstruction errors
- **In other words, minimum reconstruction error**



# SVD - Interpretation #2

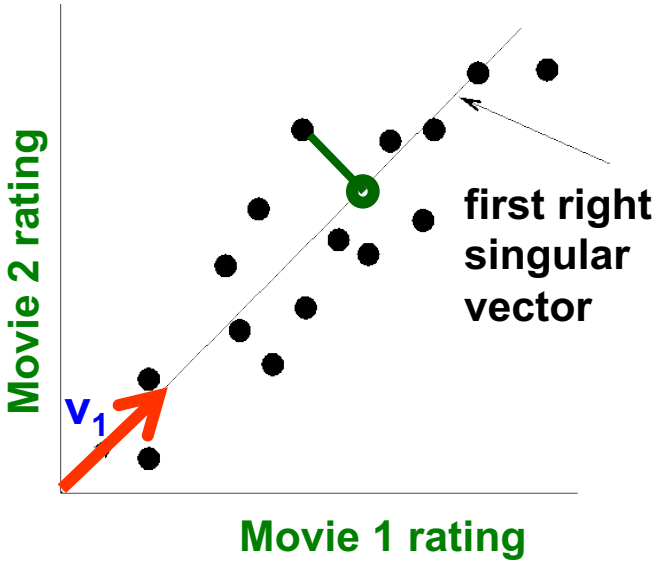
- **$A = U \Sigma V^T$ - example:**
  - **V**: "movie-to-concept" matrix
  - **U**: "user-to-concept" matrix




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

# SVD - Interpretation #2

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



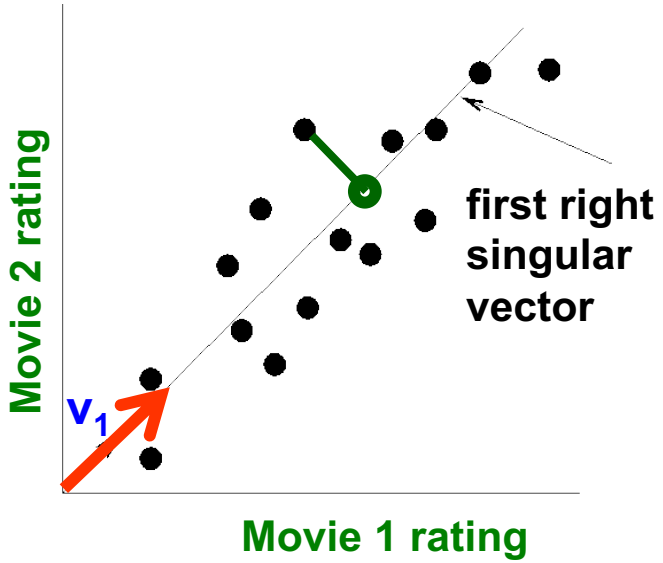

variance ('spread') on the $v_1$ axis

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

# SVD - Interpretation #2

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

- $U \Sigma$: Gives the coordinates of the points in the projection axis




$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix}$$

**Projection of users on the "Sci-Fi" axis ($U \Sigma)^T$:**

$$\begin{bmatrix} 1.61 & 0.19 & -0.01 \\ 5.08 & 0.66 & -0.03 \\ 6.82 & 0.85 & -0.05 \\ 8.43 & 1.04 & -0.06 \\ 1.86 & -5.60 & 0.84 \\ 0.86 & -6.93 & -0.87 \\ 0.86 & -2.75 & 0.41 \end{bmatrix}$$

# SVD - Interpretation #2

## More details

- Q: How exactly is dim. reduction done?

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

# SVD - Interpretation #2

## More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

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

# SVD - Interpretation #2

## More details

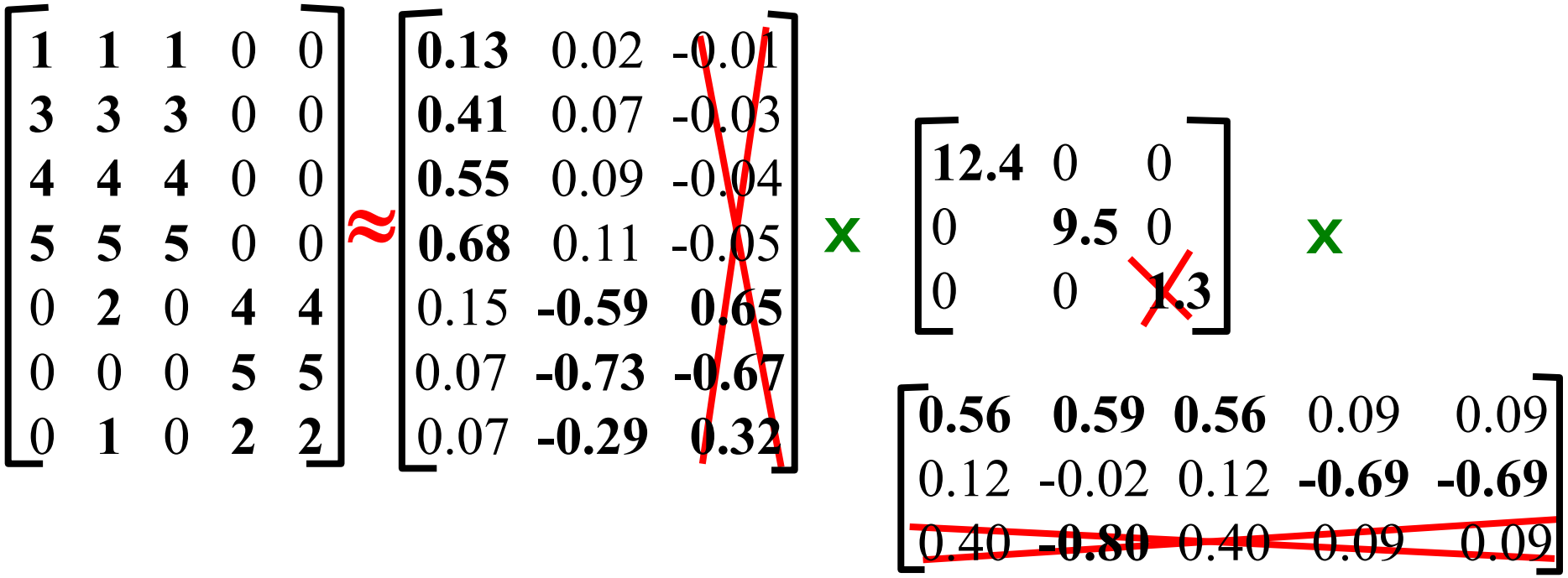
- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

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

# SVD - Interpretation #2

## More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero



# SVD - Interpretation #2

## More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.13 & 0.02 \\ 0.41 & 0.07 \\ 0.55 & 0.09 \\ 0.68 & 0.11 \\ 0.15 & -0.59 \\ 0.07 & -0.73 \\ 0.07 & -0.29 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \end{bmatrix}$$

# SVD - Interpretation #2

## More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

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

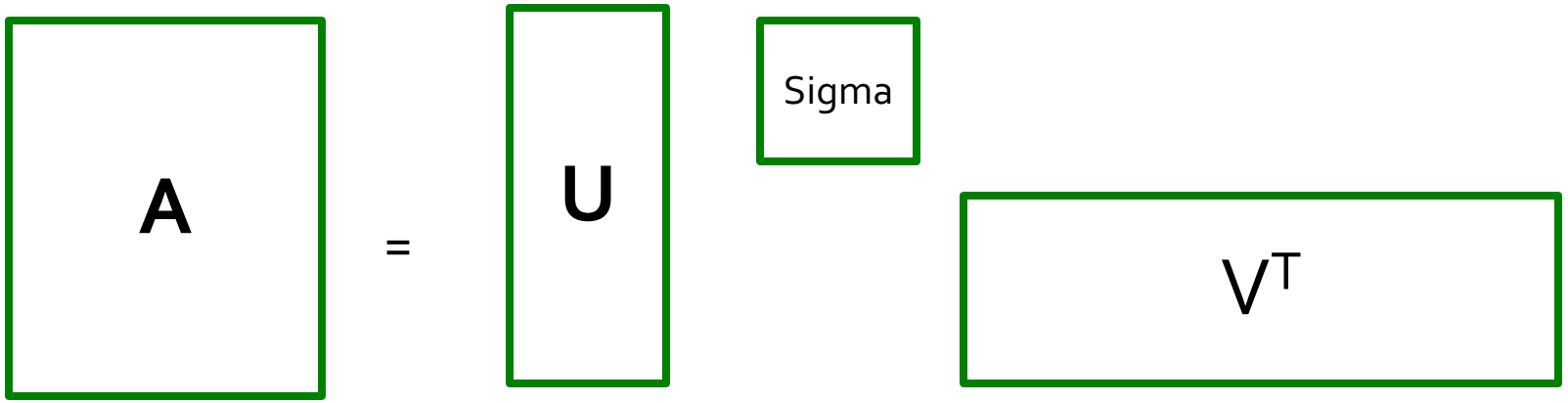
**Frobenius norm:**

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

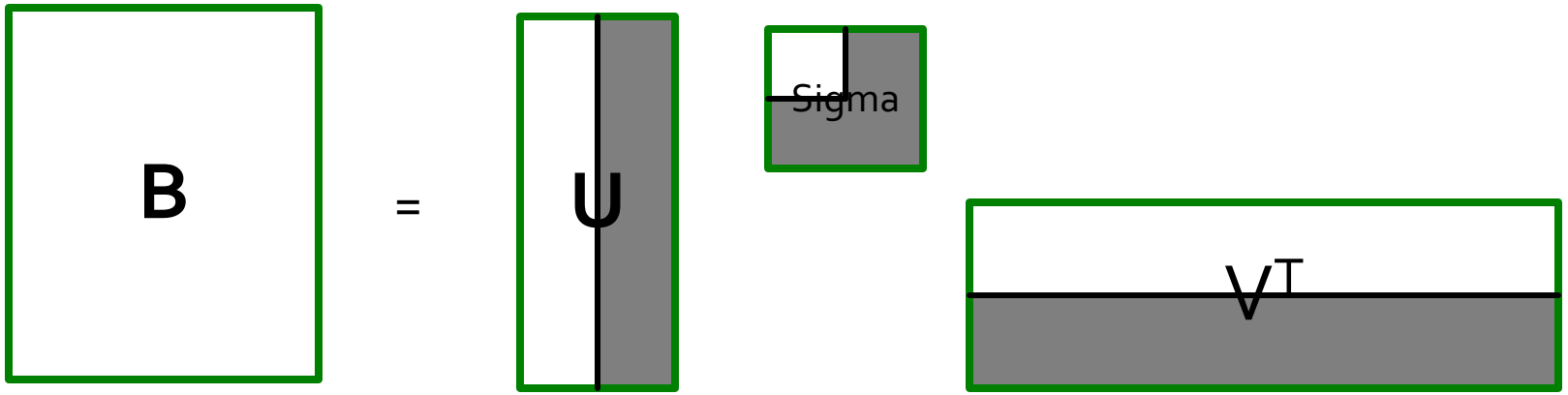
$$\|A-B\|_F = \sqrt{\Sigma_{ij} (A_{ij}-B_{ij})^2}$$

is "small"

# SVD – Best Low Rank Approx.




**B 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 *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***

What do we mean by "best":

- $B$ **is a solution to** $\min_B \|A-B\|_F$ **where** $\text{rank}(B)=k$

$$\underset{m\times n}{\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}} = \underset{m\times r}{\begin{pmatrix} & U & \\ u_{11} & \cdots & \\ \vdots & \ddots & \\ u_{m1} & & \end{pmatrix}} \underset{r\times r}{\begin{pmatrix} & \Sigma & \\ \sigma_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & \end{pmatrix}} \underset{r\times n}{\begin{pmatrix} & V^T & \\ v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ & & \end{pmatrix}}$$

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

# SVD – Best Low Rank Approx.

Details!

- **Theorem:** Let $\mathbf{A} = \mathbf{U}\,\Sigma\,\mathbf{V}^T$ ($\sigma_1 \geq \sigma_2 \geq \ldots$, rank($\boldsymbol{A}$)=$\boldsymbol{r}$) **then** $\mathbf{B} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^T$
  - **S = diagonal $r \times r$ matrix** where $s_i = \sigma_i$ ($i=1\ldots k$) else $s_i = 0$

  is a best rank-$\boldsymbol{k}$ approximation to $\boldsymbol{A}$:
  - $\boldsymbol{B}$ **is a solution to** $\min_B \|A-B\|_F$ **where** $\mathrm{rank}(B)=k$

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}_{m \times n} = \underset{m \times r}{\begin{pmatrix} u_{11} & \cdots & \\ \vdots & \ddots & \\ u_{m1} & & \end{pmatrix}} \begin{pmatrix} \sigma_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & \end{pmatrix}_{r \times r} \underset{r \times n}{\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ & & \end{pmatrix}}$$

- **We will need 2 facts:**
  - where $\boldsymbol{M} = \boldsymbol{P}\,\boldsymbol{Q}\,\boldsymbol{R}$ is SVD of $\boldsymbol{M}$
  - $\mathbf{U}\,\Sigma\,\mathbf{V}^T - \mathbf{U}\,\mathbf{S}\,\mathbf{V}^T = \mathbf{U}\,(\Sigma - \mathbf{S})\,\mathbf{V}^T$

# SVD – Best Low Rank Approx.

Details!

- **We will need 2 facts:**
  - $\left|\left|M\right|\right|_F = \sum_k (q_{kk})^2$ where ***M*** = ***P Q R*** is SVD of ***M***

$$\|M\| = \sum_i \sum_j (m_{ij})^2 = \sum_i \sum_j \left( \sum_k \sum_\ell p_{ik} q_{k\ell} r_{\ell j} \right)^2$$

$$\|M\| = \sum_i \sum_j \sum_k \sum_\ell \sum_n \sum_m p_{ik} q_{k\ell} r_{\ell j} p_{in} q_{nm} r_{mj}$$

$\sum_i p_{ik} p_{in}$ is 1 if $k = n$ and 0 otherwise

**We apply:**
-- P column orthonormal
-- R row orthonormal
-- Q is diagonal

  - **U Σ V$^T$ - U S V$^T$ = U (Σ - S) V$^T$**

# SVD – Best Low Rank Approx.

Details!

- **A = U Σ V$^T$ , B = U S V$^T$** $(\sigma_1 \geq \sigma_2 \geq \ldots \geq 0, \text{rank}(A)=r)$
  - **S** = diagonal $n \times n$ matrix where $s_i=\sigma_i$ $(i=1\ldots k)$ else $s_i=0$

  **then** $B$ is solution to $\min_B \|A-B\|_F$ , $\text{rank}(B)=k$
- **Why?**

$$\min_{B,rank(B)=k} \|A-B\|_F = \min \|\Sigma - S\|_F = \min_{s_i} \sum_{i=1}^{r} (\sigma_i - s_i)^2$$

**We used: U Σ V$^T$ - U S V$^T$ = U (Σ - S) V$^T$**

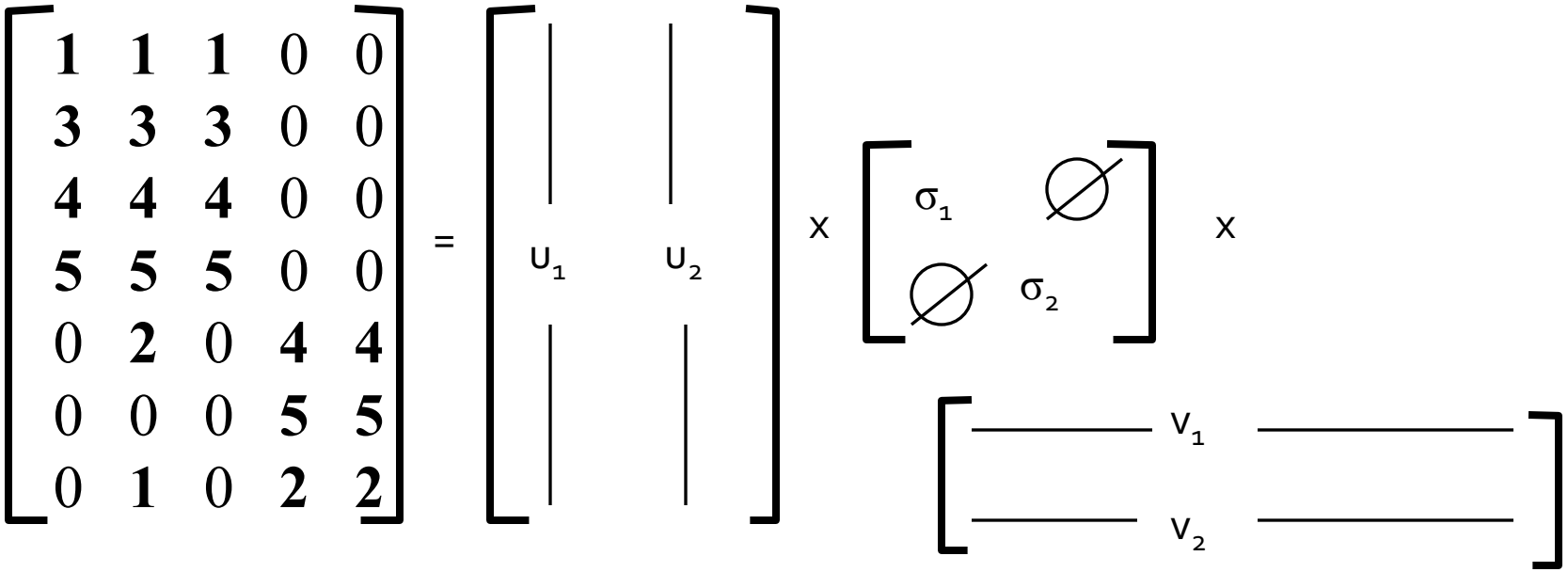
- We want to choose $s_i$ to minimize $\sum_i (\sigma_i - s_i)^2$
- Solution is to set $s_i=\sigma_i$ $(i=1\ldots k)$ and other $s_i=0$

$$= \min_{s_i} \sum_{i=1}^{k} (\sigma_i - s_i)^2 + \sum_{i=k+1}^{r} \sigma_i^2 = \sum_{i=k+1}^{r} \sigma_i^2$$

# SVD - Interpretation #2

**Equivalent:**

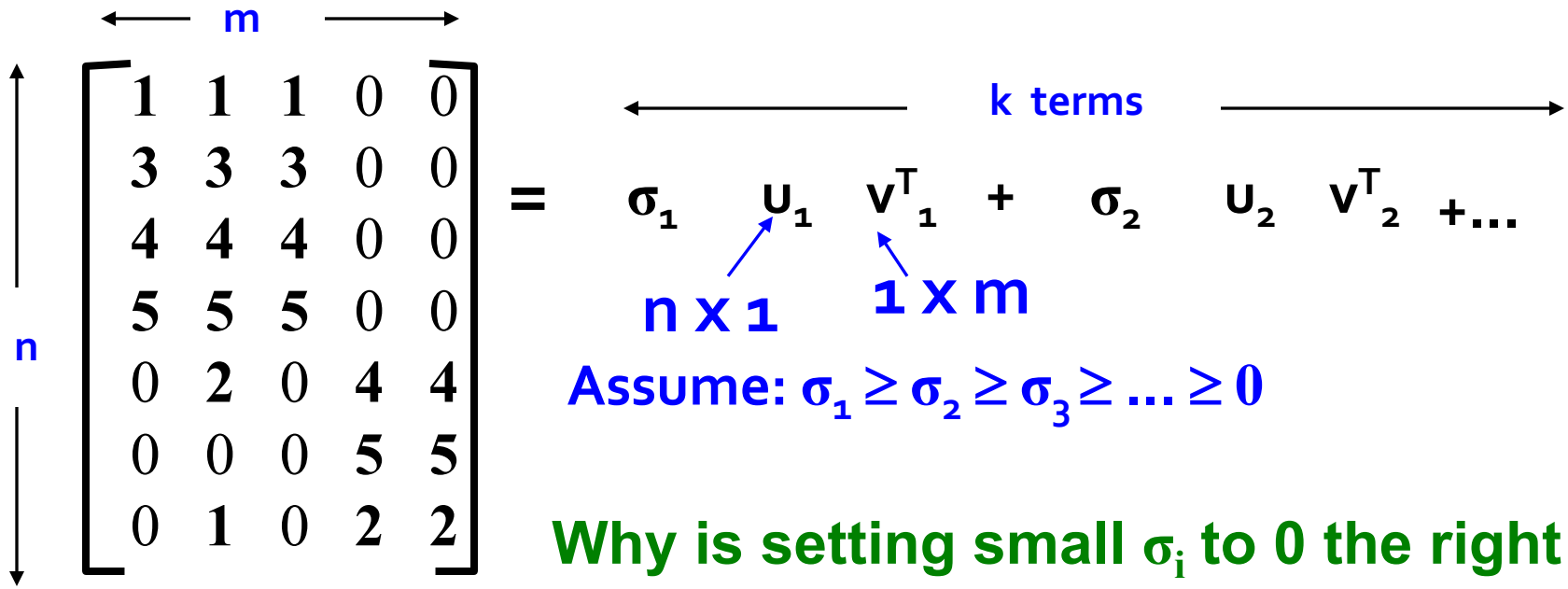
**'spectral decomposition' of the matrix:**



# SVD - Interpretation #2

**Equivalent:**

**'spectral decomposition' of the matrix**




**Why is setting small $\sigma_i$ to 0 the right thing to do?**

Vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ are unit length, so $\sigma_i$ scales them.

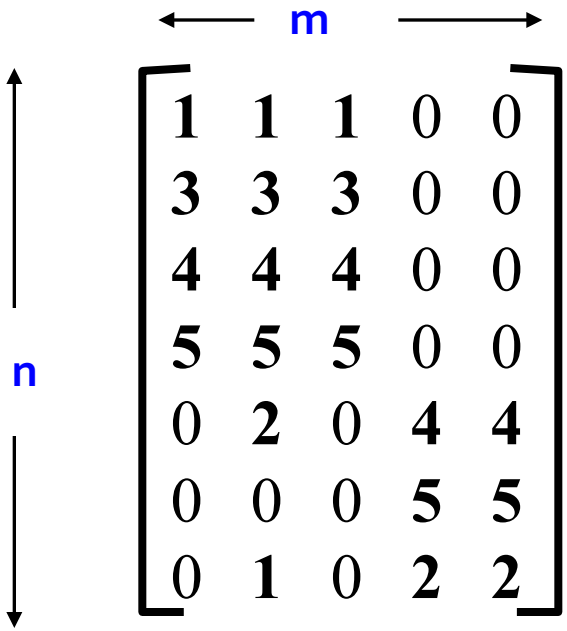
So, zeroing small $\sigma_i$ introduces less error.

# SVD - Interpretation #2

**Q: How many $\sigma_s$ to keep?**

**A:** Rule-of-a thumb:

**keep 80-90% of 'energy'** $= \sum_i \sigma_i^2$




$= \sigma_1 \; u_1 \; v^T_1 + \sigma_2 \; u_2 \; v^T_2 + \ldots$

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

# 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 ...

# SVD - Conclusions so far

- **SVD: A= U Σ $V^T$: unique**
  - **U**: user-to-concept similarities
  - **V**: movie-to-concept similarities
  - Σ : strength of each concept
- **Dimensionality reduction:**
  - keep the few largest singular values (80-90% of 'energy')
  - SVD: picks up linear correlations

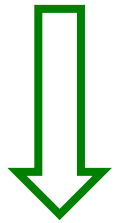
# Relation to Eigen-decomposition

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

# Relation to Eigen-decomposition

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

Shows how to compute SVD using eigenvalue decomposition!



$X \Lambda^2 X^T$

$X \Lambda^2 X^T$

# SVD: Properties

- $\mathbf{A}\,\mathbf{A}^T = \mathbf{U}\,\Sigma^2\,\mathbf{U}^T$
- $\mathbf{A}^T\mathbf{A} = \mathbf{V}\,\Sigma^2\,\mathbf{V}^T$

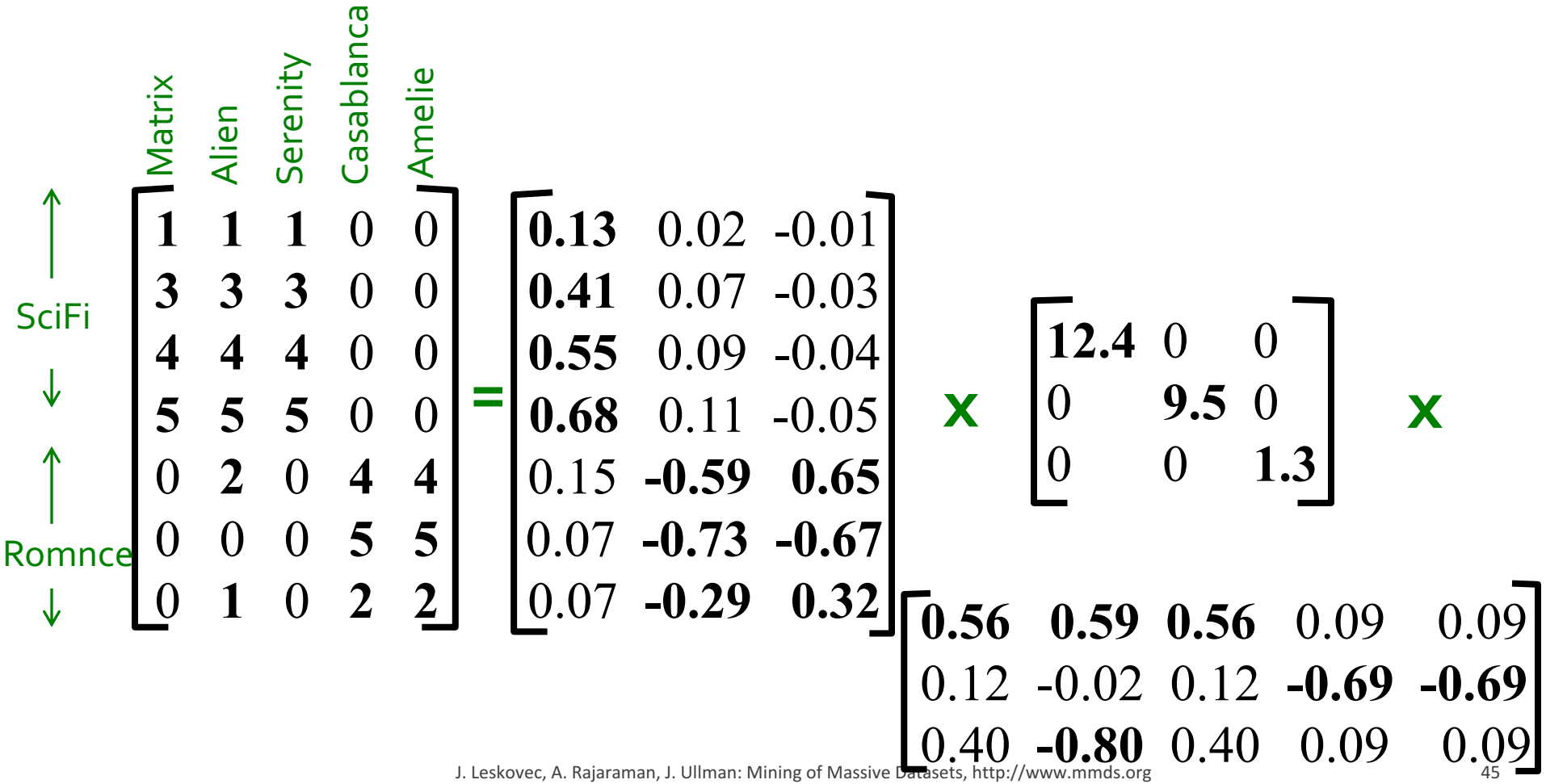
- $(\mathbf{A}^T\mathbf{A})^k = \mathbf{V}\,\Sigma^{2k}\,\mathbf{V}^T$
  - E.g.: $(\mathbf{A}^T\mathbf{A})^2 = \mathbf{V}\,\Sigma^2\,\mathbf{V}^T\,\mathbf{V}\,\Sigma^2\,\mathbf{V}^T = \mathbf{V}\,\Sigma^4\,\mathbf{V}^T$

- $(\mathbf{A}^T\mathbf{A})^k \sim v_1\,\sigma_1^{2k}\,v_1^T$ for $k>>1$

# Example of SVD & Conclusion

# Case study: How to query?

- **Q: Find users that like 'Matrix'**
- **A: Map query into a 'concept space' – how?**

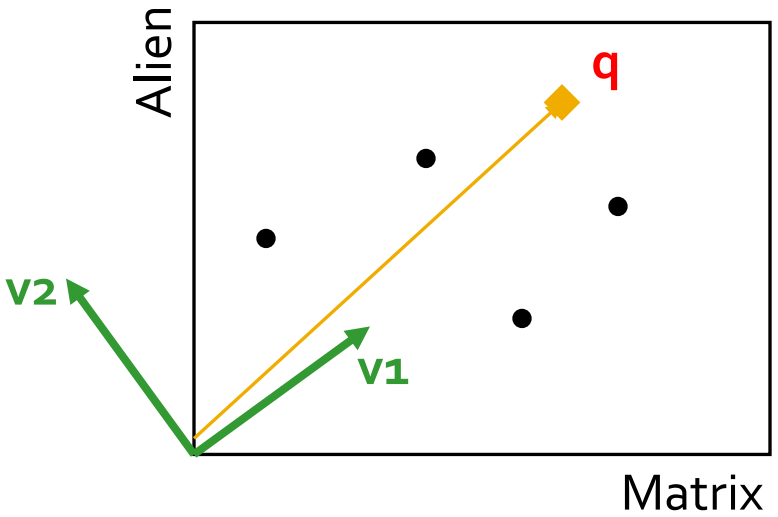


# Case study: How to query?

- **Q: Find users that like 'Matrix'**
- **A: Map query into a 'concept space' – how?**

$$q = \begin{bmatrix} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\ 5 & 0 & 0 & 0 & 0 \end{bmatrix}$$

**Project into concept space:**
Inner product with each 'concept' vector $\mathbf{v}_i$

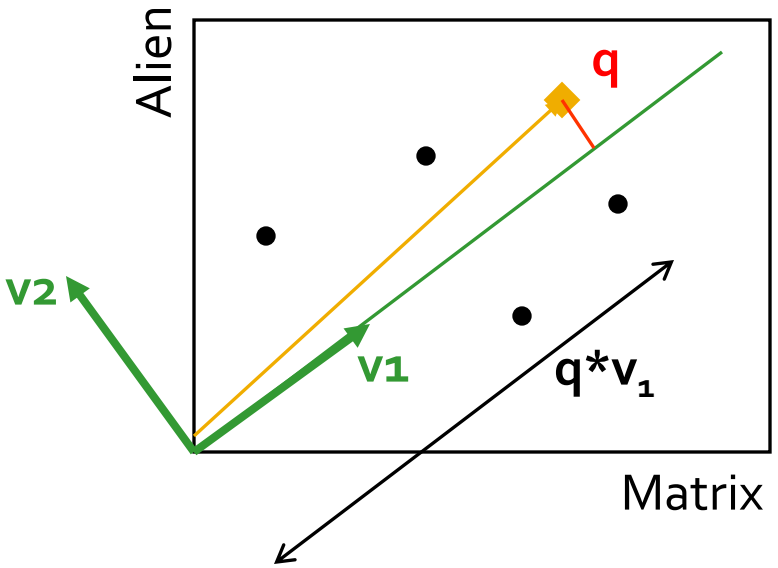


# Case study: How to query?

- **Q: Find users that like 'Matrix'**
- **A: Map query into a 'concept space' – how?**

$$q = \begin{bmatrix} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\ 5 & 0 & 0 & 0 & 0 \end{bmatrix}$$

**Project into concept space:**
Inner product with each 'concept' vector $\mathbf{v}_i$

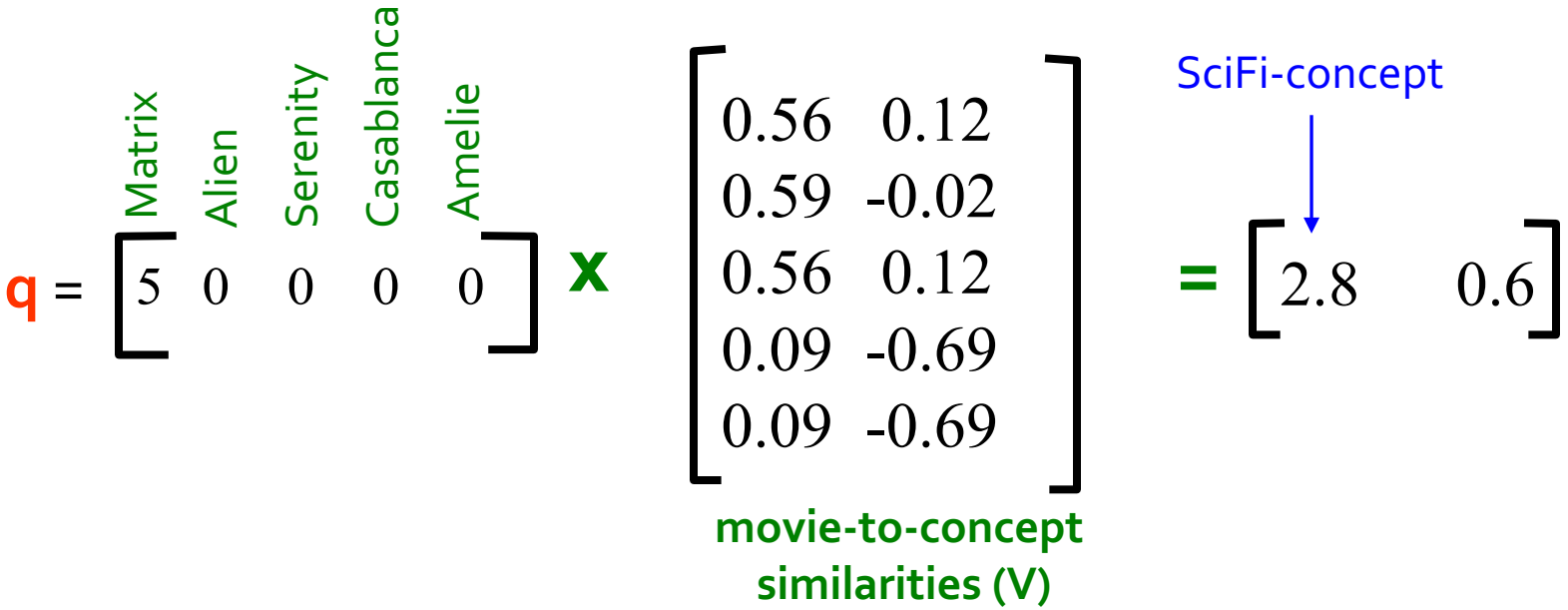


# Case study: How to query?

**Compactly, we have:**

$q_{concept} = q\ V$

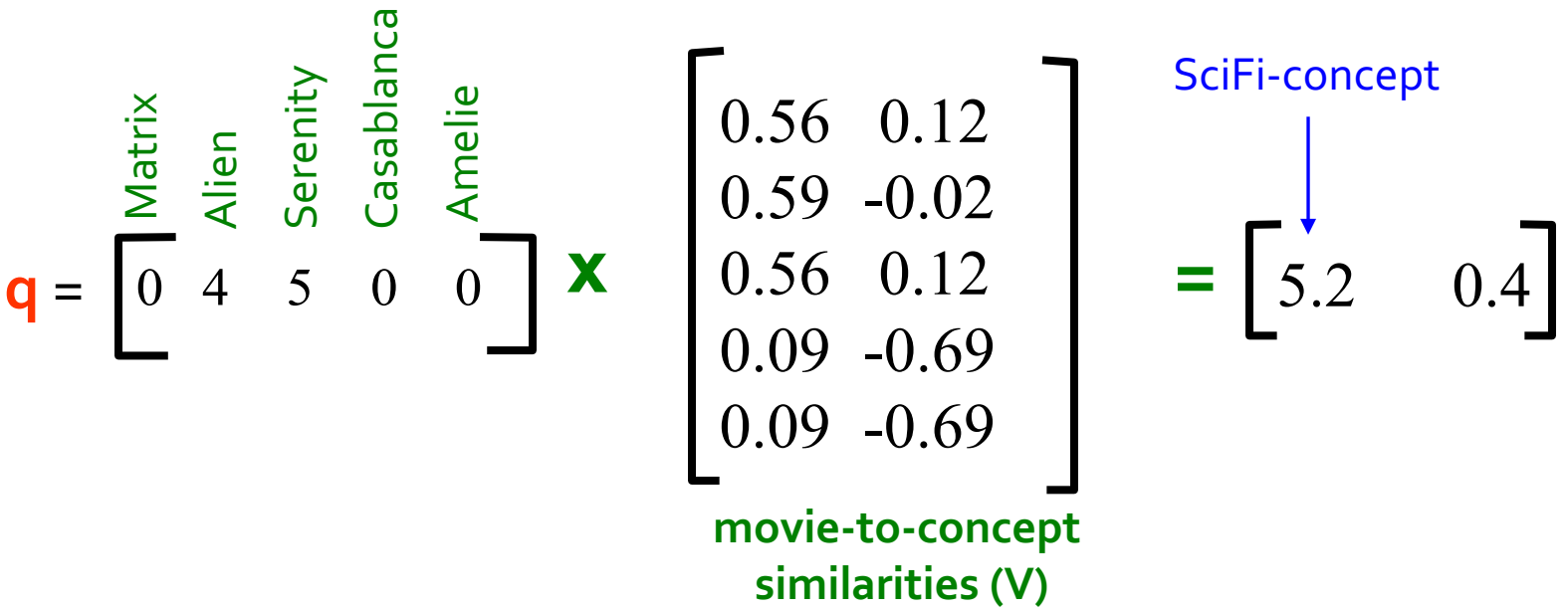
**E.g.:**



# Case study: How to query?

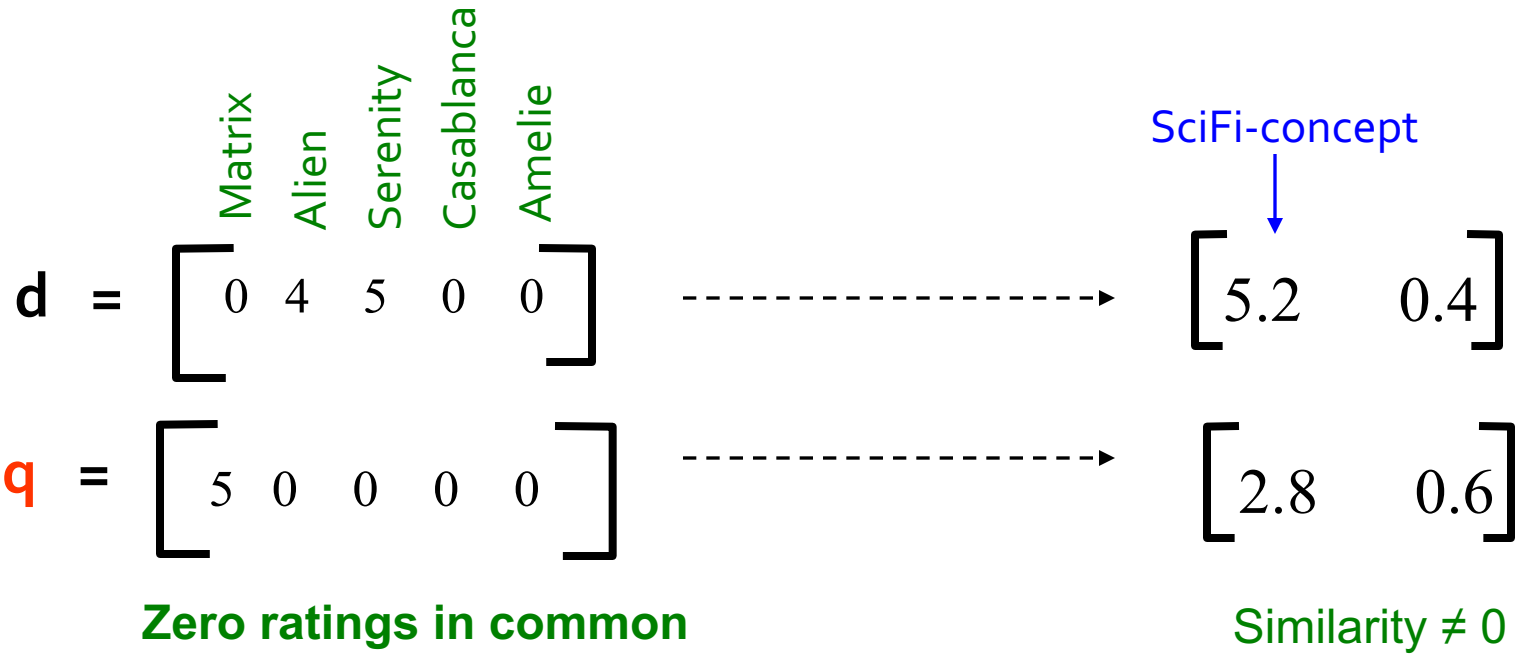
- **How would the user *d* that rated ('Alien', 'Serenity') be handled?**
  **$d_{concept} = d\ V$**

**E.g.:**



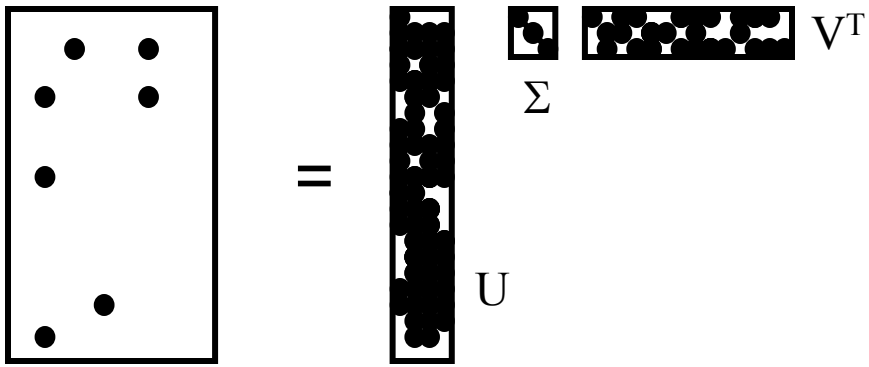
# Case study: How to query?

- **Observation:** User ***d*** that rated ('*Alien*', '*Serenity*') will be **similar** to user **q** that rated ('*Matrix*'), although ***d*** and **q** 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!**

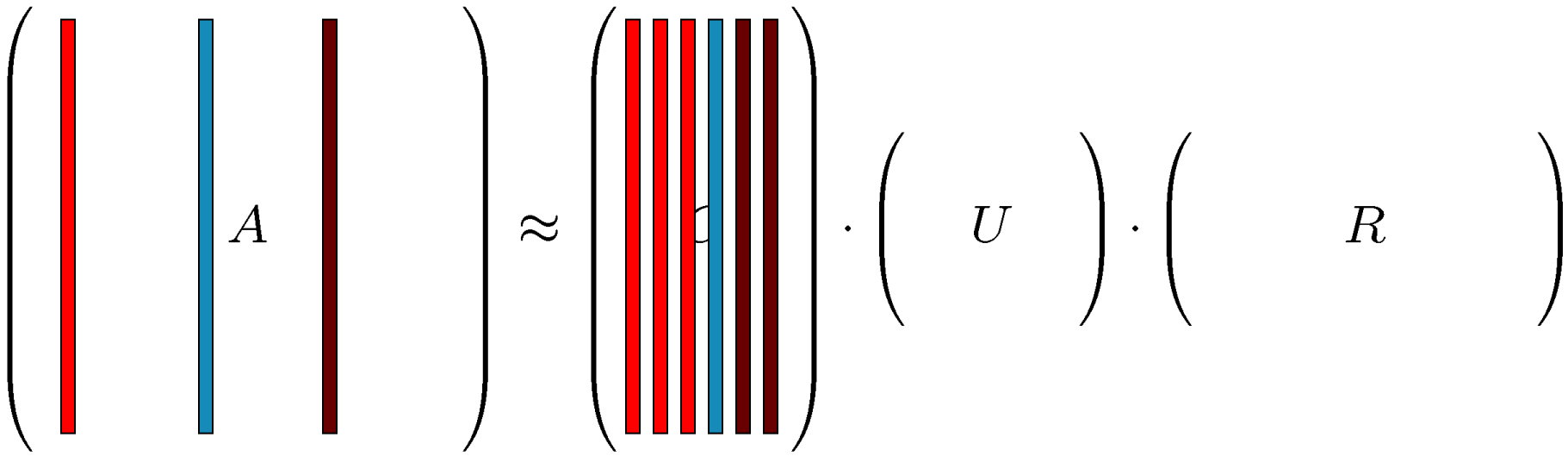


# CUR Decomposition

# CUR Decomposition

Frobenius norm:
$\|X\|_F = \sqrt{\Sigma_{ij} X_{ij}^2}$

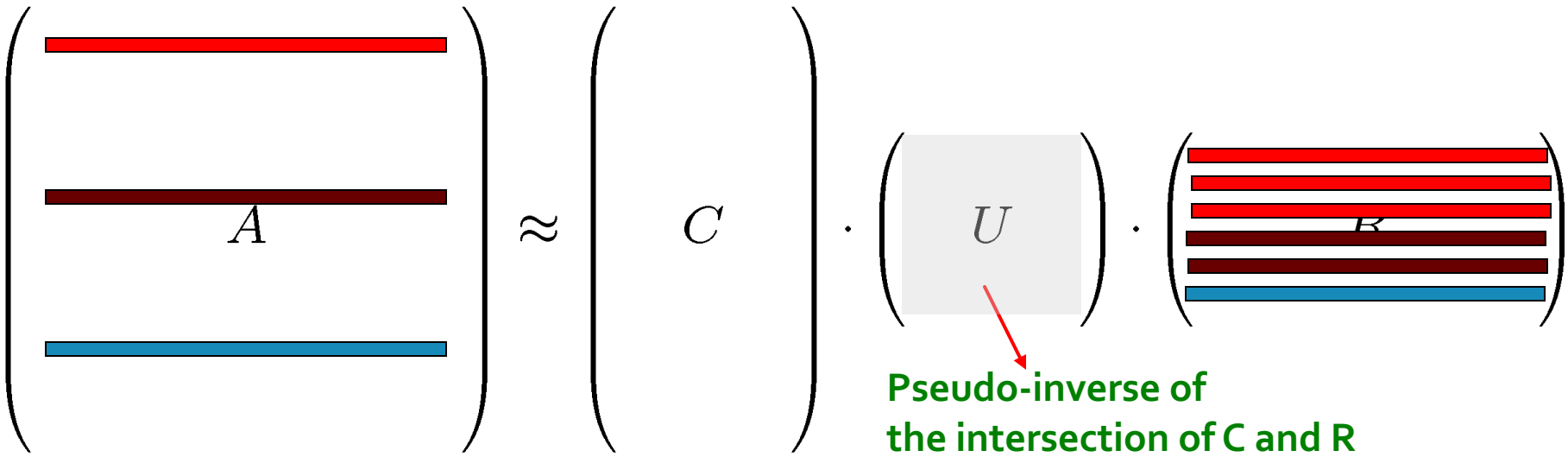
- **Goal: Express A as a product of matrices C,U,R**
  **Make $\|A-C\cdot U\cdot R\|_F$ small**
- **"Constraints" on C and R:**



# CUR Decomposition

Frobenius norm:
$\|X\|_F = \sqrt{\Sigma_{ij} X_{ij}^2}$

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



# CUR: Provably good approx. to SVD

- **Let:**

  $\mathbf{A_k}$ be the "best" rank $\boldsymbol{k}$ approximation to $\mathbf{A}$ (that is, $\mathbf{A_k}$ is SVD of A)

**Theorem** [Drineas et al.]

**CUR** in O(**m·n**) time achieves

- $\|\mathbf{A-CUR}\|_F \leq \|\mathbf{A-A_k}\|_F + \varepsilon\|\mathbf{A}\|_F$

with probability at least **1-δ**, by picking

- $O(k \log(1/\delta)/\varepsilon^2)$ columns, and
- $O(k^2 \log^3(1/\delta)/\varepsilon^6)$ rows

**In practice:**
Pick 4*k* cols/rows

# CUR: How it Works

- **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. $P(x) = \sum_i \mathbf{A}(i,x)^2 / \sum_{i,j} \mathbf{A}(i,j)^2$
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

# CUR: How it Works

- Removing duplicates:

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

**Output**: $\mathbf{C}_s \in \mathbb{R}^{m \times c'}$

1. Compute $\mathbf{C}_d$ using the intial subspace construction
2. Let $\mathbf{C} \in \mathbb{R}^{m \times c'}$ be the unique columns of $\mathbf{C}_d$
3. For $i = 1 : c'$
4. Let $u$ be the number of $\mathbf{C}(:, i)$ in $\mathbf{C}_d$
5. Compute $\mathbf{C}_s(:, i) \leftarrow \sqrt{u} \cdot \mathbf{C}(:, i)$

# Proof

THEOREM 4.1. (DUPLICATE COLUMNS) *Matrices* $\mathbf{C}_S$ *and* $\mathbf{C}_D$, *defined in Table 2, have the same singular values and left singular vectors.*

*Proof.* It is easy to see $\mathbf{C}_d = \mathbf{C}\mathbf{D}^T$. Then we have

$$\mathbf{C}_d\mathbf{C}_d^T = \mathbf{C}\mathbf{D}^T(\mathbf{C}\mathbf{D}^T)^T = \mathbf{C}\mathbf{D}^T\mathbf{D}\mathbf{C}^T \tag{4.1}$$

$$= \mathbf{C}\Lambda\mathbf{C}^T = \mathbf{C}\Lambda^{1/2}\Lambda^{1/2}\mathbf{C}^T \tag{4.2}$$

$$= \mathbf{C}\Lambda^{1/2}(\mathbf{C}\Lambda^{1/2})^T = \mathbf{C}_s\mathbf{C}_s^T \tag{4.3}$$

where $\Lambda \in \mathbb{R}^{k\times k}$ is defined in Table 2[3].

Now we can diagonalize either the product $\mathbf{C}_d\mathbf{C}_d^T$ or $\mathbf{C}_s\mathbf{C}_s^T$ to find the same singular values and left singular vectors for both $\mathbf{C}_d$ and $\mathbf{C}_s$.

# CUR: How it Works

Approximate Multiplication:

**Input**: matrix $\mathbf{A} \in \mathbb{R}^{c \times m}$, $\mathbf{B} \in \mathbb{R}^{m \times n}$, sample size $r$

**Output**: $\mathbf{C}_s \in \mathbb{R}^{c \times r'}$ and $\mathbf{R}_s \in \mathbb{R}^{r' \times n}$

1. for $x = 1 : m$ [row distribution of $\mathbf{B}$]
2. $Q(x) = \sum_i \mathbf{B}(x,i)^2 / \sum_{i,j} \mathbf{B}(i,j)^2$
3. for $i = 1 : r$
4. Pick $j \in 1 : r$ based on distribution $Q(x)$
5. Set $\mathbf{R}_d(i,:) = \mathbf{B}(j,:) / \sqrt{rQ(j)}$
6. Set $\mathbf{C}_d(:,i) = \mathbf{A}(:,j) / \sqrt{rQ(j)}$
7. $\mathbf{R} \in \mathbb{R}^{r' \times n}$ are the unique rows of $\mathbf{R}_d$
8. $\mathbf{C} \in \mathbb{R}^{c \times r'}$ are the unique columns of $\mathbf{C}_d$
9. for $i = 1 : r'$
10. $u$ is the number of $\mathbf{R}(i,:)$ in $\mathbf{R}_d$
11. Set $\mathbf{R}_s(i,:) \leftarrow u \cdot \mathbf{R}(i,:)$
12. Set $\mathbf{C}_s(:,i) \leftarrow \mathbf{C}(:,i)$

# CUR: How it Works

Final Algorithm:

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

1. find $\mathbf{C}$ from CMD subspace construction
2. diagonalize $\mathbf{C}^T\mathbf{C}$ to find $\mathbf{\Sigma}_C$ and $\mathbf{V}_C$
3. find $\mathbf{C}_s$ and $\mathbf{R}_s$ using ApprMultiplication on $\mathbf{C}^T$ and $\mathbf{A}$
4. $\mathbf{U} = \mathbf{V}_C \mathbf{\Sigma}_C^{-2} \mathbf{V}_C^T \mathbf{C}_s$

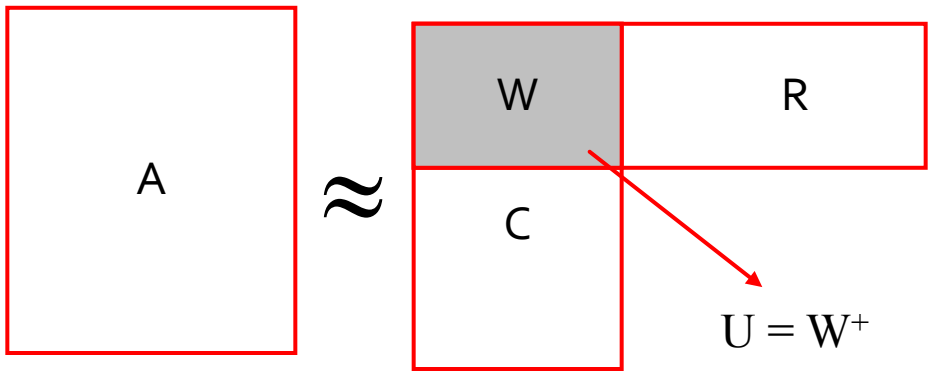
# CUR: How it Works

Why it works:

$$\tilde{\mathbf{A}} = \mathbf{U}_c\mathbf{U}_c^T\mathbf{A} = \mathbf{C}\mathbf{V}_C\mathbf{\Sigma}_C^{-1}(\mathbf{C}\mathbf{V}_C\mathbf{\Sigma}_C^{-1})^T\mathbf{A}$$
$$= \mathbf{C}(\mathbf{V}_C\mathbf{\Sigma}_C^{-2}\mathbf{V}_C^T\mathbf{C}^T)\mathbf{A} = \mathbf{C}\mathbf{T}\mathbf{A}$$

# Computing U

- Let **W** be the "intersection" of sampled columns **C** and rows **R**
  - Let SVD of **W** = **X Z** $\mathbf{Y}^T$
- **Then: U** = $\mathbf{W}^+$ = **Y** $\mathbf{Z}^+$ $\mathbf{X}^T$
  - $Z^+$: **reciprocals of non-zero singular values:** $Z^+_{ii} = 1/Z_{ii}$
  - $W^+$ is the "**pseudoinverse**"




**Why pseudoinverse works?**

W = X Z Y then $W^{-1} = X^{-1} Z^{-1} Y^{-1}$

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

# Proof

THEOREM 5. *Suppose $A \in \mathbb{R}^{m \times n}$; let a description of $\tilde{H}_\ell$ be constructed from the* CONSTANTTIMESVD *algorithm by sampling $c$ columns of $A$ with probabilities $\{p_i\}_{i=1}^n$ and $w$ rows of $C$ with probabilities $\{q_j\}_{j=1}^m$ where $p_i = |A^{(i)}|^2 / \|A\|_F^2$ and $q_j = |C_{(j)}|^2 / \|C\|_F^2$. Let $\eta = 1 + \sqrt{8 \log(2/\delta)}$ and $\epsilon > 0$.*

*If a Frobenius norm bound is desired, and hence the* CONSTANTTIMESVD *algorithm is run with $\gamma = \epsilon/100k$, then by choosing $c = \Omega(k^2\eta^2/\epsilon^4)$ columns of $A$ and $w = \Omega(k^2\eta^2/\epsilon^4)$ rows of $C$ we have that with probability at least $1 - \delta$,*

$$\left\|A - \tilde{H}_\ell \tilde{H}_\ell^T A\right\|_F^2 \leq \|A - A_k\|_F^2 + \epsilon \|A\|_F^2 . \tag{35}$$

*If a spectral norm bound is desired, and hence the* CONSTANTTIMESVD *algorithm is run with $\gamma = \epsilon/100$, then by choosing $c = \Omega(\eta^2/\epsilon^4)$ columns of $A$ and $w = \Omega(\eta^2/\epsilon^4)$ rows of $C$ we have that with probability at least $1 - \delta$,*

$$\left\|A - \tilde{H}_\ell \tilde{H}_\ell^T A\right\|_2^2 \leq \|A - A_k\|_2^2 + \epsilon \|A\|_F^2 . \tag{36}$$

# CUR: Provably good approx. to SVD

- Algorithm:

1. Run ColumnSelect on $A$ with $c = O(k \log k/\epsilon^2)$ to choose columns of $A$ and construct the matrix $C$.
2. Run ColumnSelect on $A^T$ with $r = O(k \log k/\epsilon^2)$ to choose rows of $A$ (columns of $A^T$) and construct the matrix $R$.
3. Define the matrix $U$ as $U = C^+AR^+$, where $X^+$ denotes a Moore–Penrose generalized inverse of the matrix $X$ (17).

$$\|A - CUR\|_F \leq (2+\epsilon)\,\|A - A_k\|_F$$

SVD error

CUR error

**In practice:**
Pick $4k$ cols/rows
for a "rank-k" approximation

# ColumnSelect

- Given matrix A, compute leverage scores:

$$\pi_j = \frac{1}{k} \sum_{\xi=1}^{k} (v_j^{\xi})^2,$$

- Algorithm:

1. Compute $v^1, \ldots, v^k$ (the top $k$ right singular vectors of $A$) and the normalized statistical leverage scores of Eq. **3**.
2. Keep the $j$th column of $A$ with probability $p_j = \min\{1, c\pi_j\}$, for all $j \in \{1, \ldots, n\}$, where $c = O(k \log k/\epsilon^2)$.
3. Return the matrix $C$ consisting of the selected columns of $A$.

# Bound

- By algorithm:

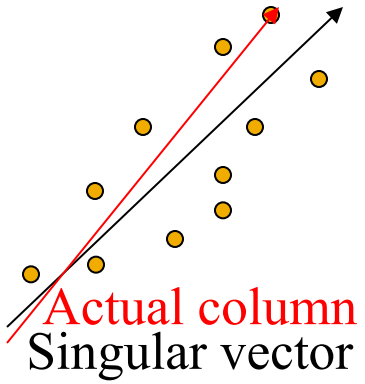
$$\|A - CUR\|_F = \left\|A - CC^+AR^+R\right\|_F .$$

- Hence:

$$\begin{aligned}\|A - CUR\|_F &\leq \left\|A - CC^+A\right\|_F + \left\|CC^+A - CC^+AR^+R\right\|_F \\ &\leq \left\|A - CC^+A\right\|_F + \left\|A - AR^+R\right\|_F \\ &= \|A - P_C A\|_F + \|A - AP_R\|_F .\end{aligned}$$

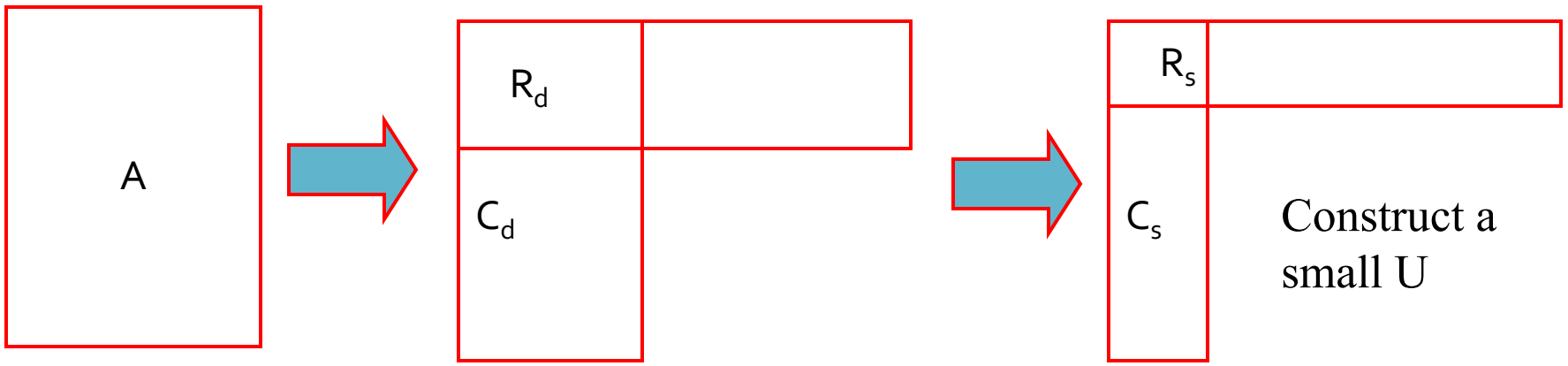
# CUR: Pros & Cons

+ **Easy interpretation**
  - Since the basis vectors are actual columns and rows
+ **Sparse basis**
  - Since the basis vectors are actual columns and rows
- **Duplicate columns and rows**
  - Columns of large norms will be sampled many times

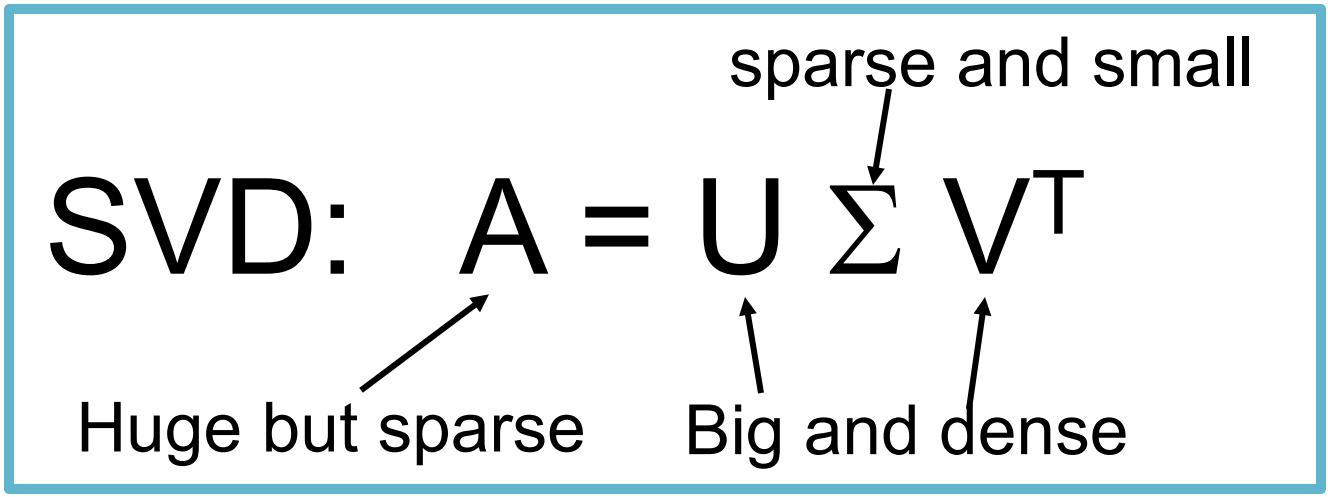


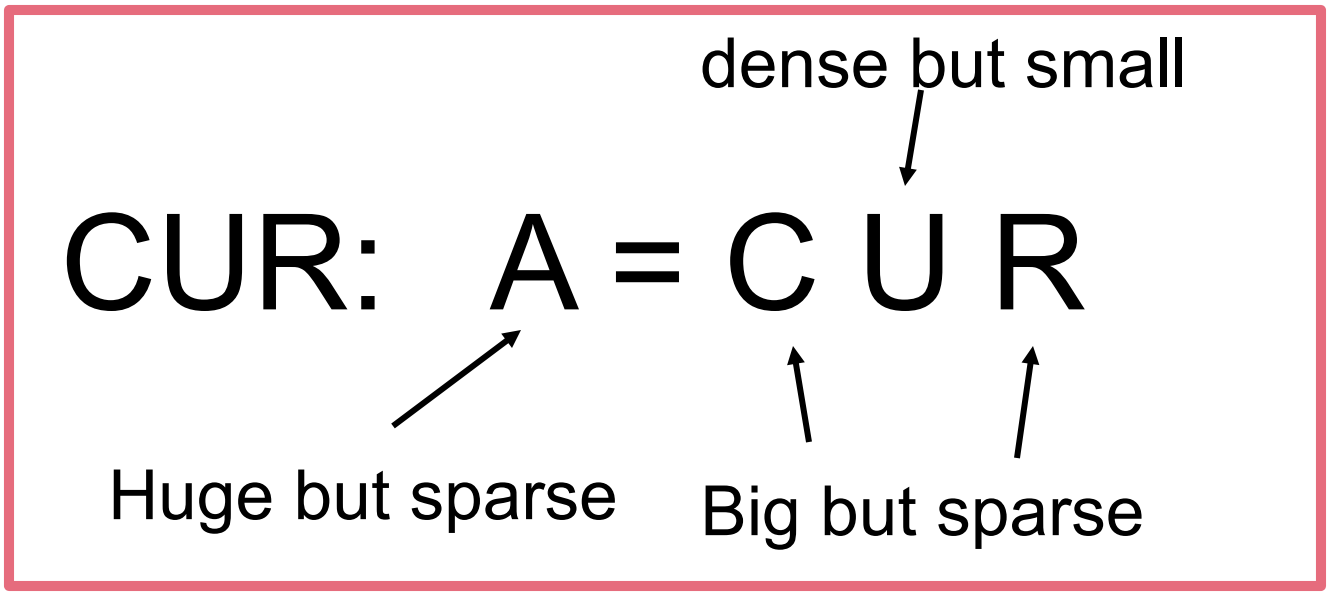
# Solution

- **If we want to get rid of the duplicates:**
  - Throw them away
  - Scale (multiply) the columns/rows by the square root of the number of duplicates



# SVD vs. CUR

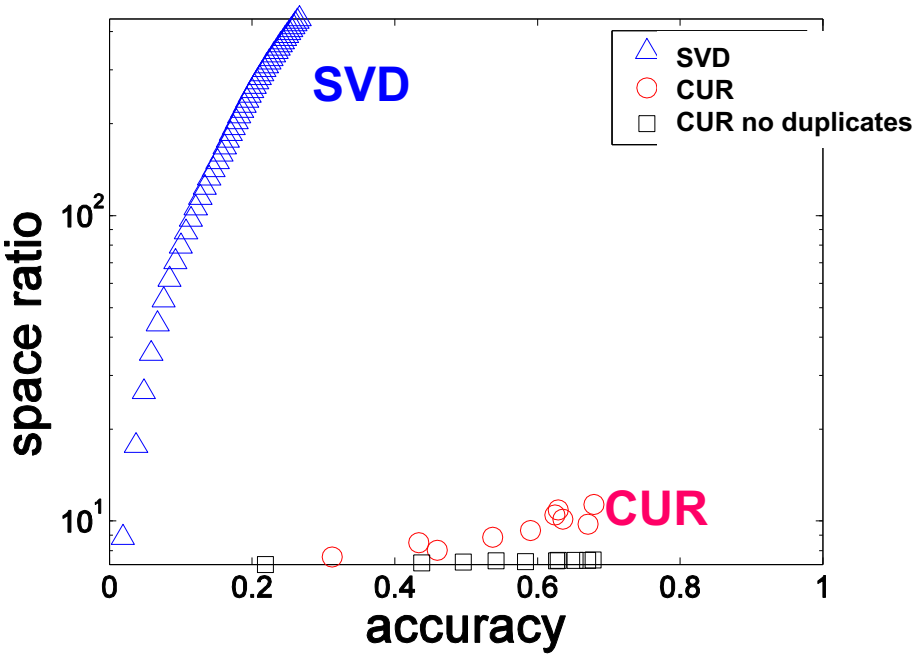

sparse and small
SVD: A = U Σ V^T
Huge but sparse
Big and dense

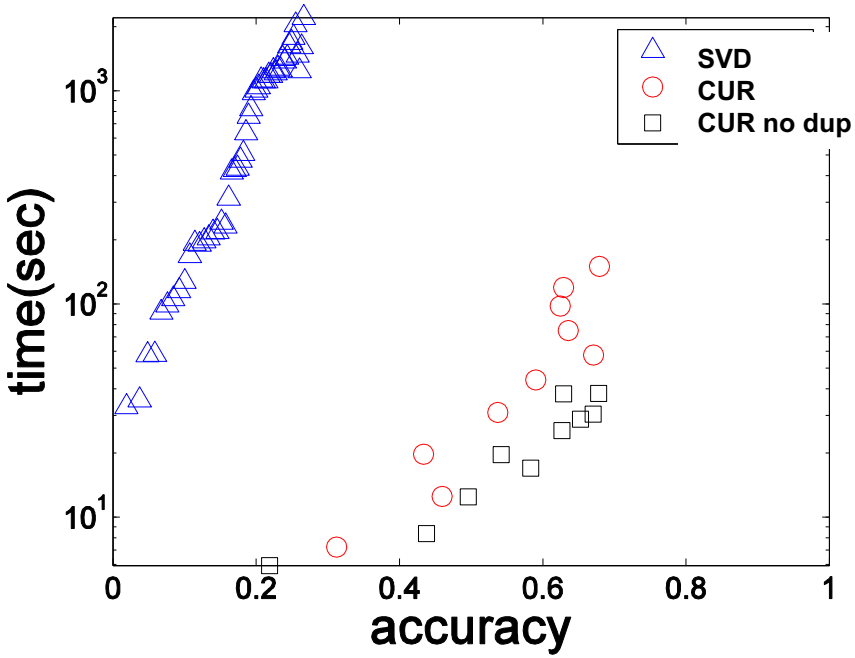

dense but small
CUR: A = C U R
Huge but sparse
Big but sparse

# SVD vs. CUR: Simple Experiment

- **DBLP bibliographic data**
  - Author-to-conference big sparse matrix
  - $A_{ij}$: Number of papers published by author *i* at conference *j*
  - 428K authors (rows), 3659 conferences (columns)
    - **Very sparse**
- **Want to reduce dimensionality**
  - How much time does it take?
  - What is the reconstruction error?
  - How much space do we need?

# Results: DBLP- big sparse matrix



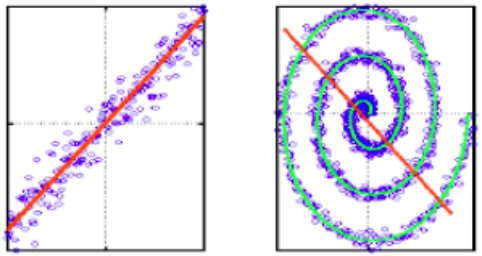




- **Accuracy:**
  - 1 – relative sum squared errors
- **Space ratio:**
  - #output matrix entries / #input matrix entries
- **CPU time**

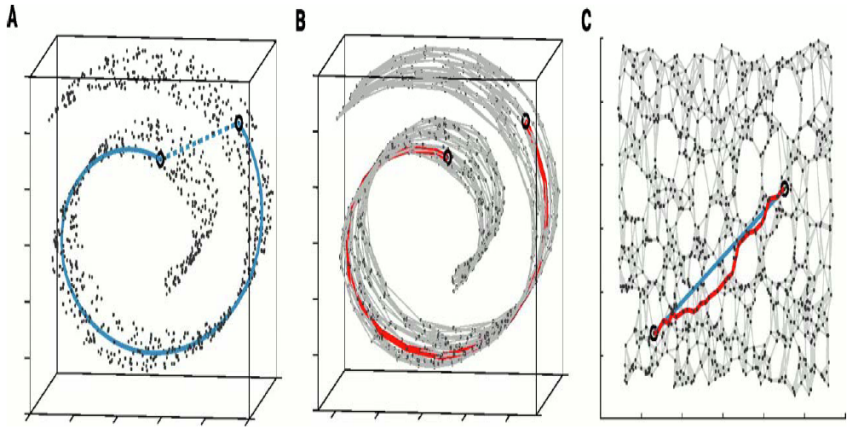
Sun, Faloutsos: *Less is More: Compact Matrix Decomposition for Large Sparse Graphs*, SDM '07.

# What about linearity assumption?

- **SVD is limited to linear projections:**
  - Lower-dimensional linear projection that preserves Euclidean distances



- Non-linear methods: **Isomap**
  - Data lies on a nonlinear low-dim curve aka manifold
    - Use the distance as measured along the manifold
  - **How?**
    - Build adjacency graph
    - Geodesic distance is graph distance
    - SVD/PCA the graph pairwise distance matrix



# Further Reading: CUR

- Drineas et al., *Fast Monte Carlo Algorithms for Matrices III: Computing a Compressed Approximate Matrix Decomposition*, SIAM Journal on Computing, 2006.
- J. Sun, Y. Xie, H. Zhang, C. Faloutsos: *Less is More: Compact Matrix Decomposition for Large Sparse Graphs*, SDM 2007
- *Intra- and interpopulation genotype reconstruction from tagging SNPs*, P. Paschou, M. W. Mahoney, A. Javed, J. R. Kidd, A. J. Pakstis, S. Gu, K. K. Kidd, and P. Drineas, Genome Research, 17(1), 96-107 (2007)
- *Tensor-CUR Decompositions For Tensor-Based Data*, M. W. Mahoney, M. Maggioni, and P. Drineas, Proc. 12-th Annual SIGKDD, 327-336 (2006)