DATA MINING
LECTURE 6

Dimensionality Reduction
PCA – SVD

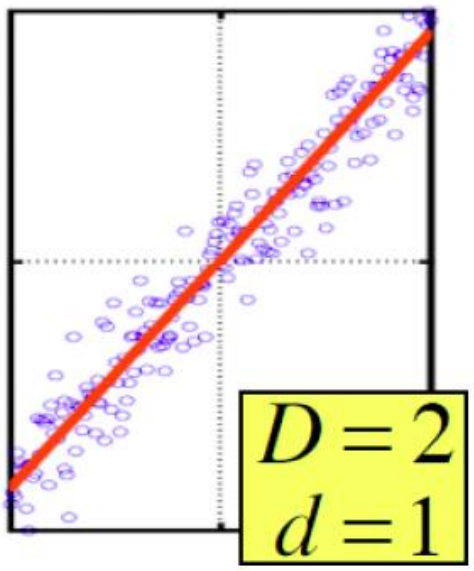
(Thanks to Jure Leskovec, Evimaria Terzi)

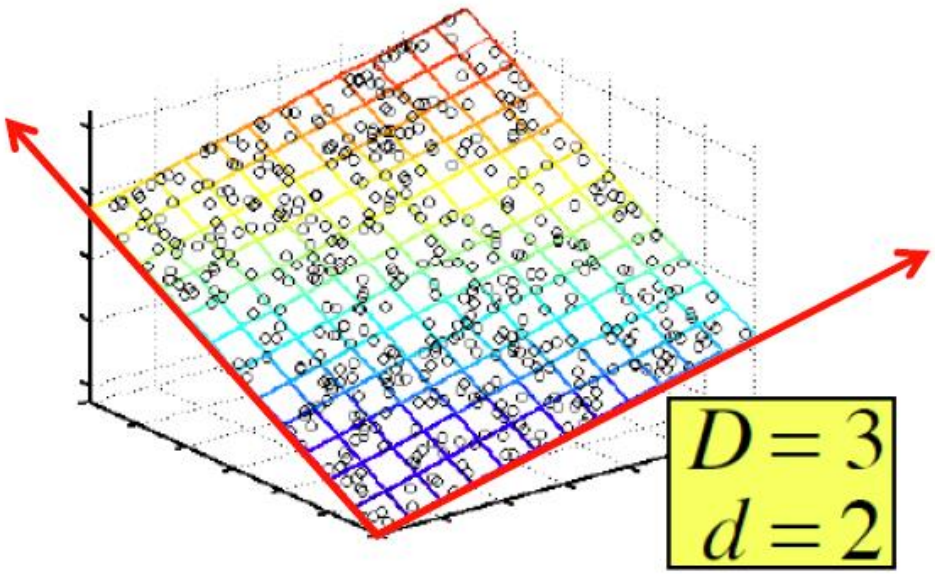
The curse of dimensionality

- Real data usually have thousands, or millions of dimensions
  - E.g., web documents, where the dimensionality is the vocabulary of words
  - Facebook graph, where the dimensionality is the number of users
- Huge number of dimensions causes problems
  - Data becomes very sparse, some algorithms become meaningless (e.g. density based clustering)
  - The complexity of several algorithms depends on the dimensionality and they become infeasible (e.g. nearest neighbor search).

Dimensionality Reduction

- Usually the data can be described with fewer dimensions, without losing much of the meaning of the data.
  - The data reside in a space of lower dimensionality





Example

TID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

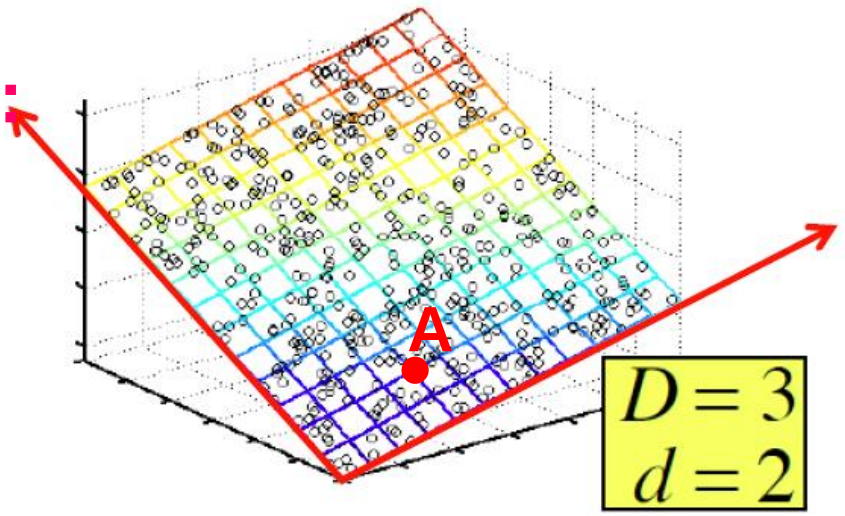
- In this data matrix the dimension is essentially 3
  - There are three types of products and three types of users

Example

- **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!**

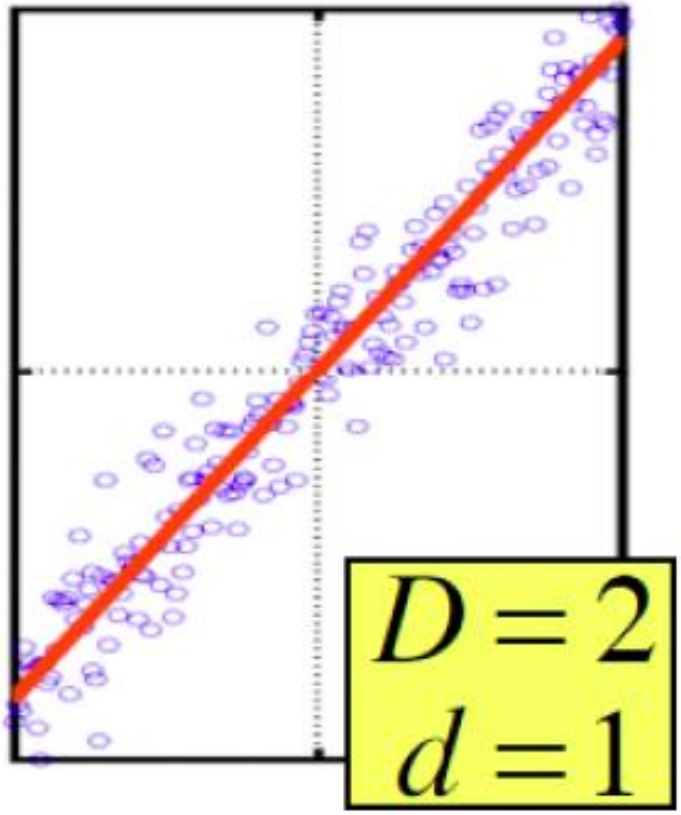
Dimensionality Reduction

- Find the "true dimension" of the data
  - In reality things are never as clear and simple as in this example, but we can still reduce the dimension.

- Essentially, we assume that some of the data is useful signal and some data is noise, and that we can approximate the useful part with a lower dimensionality space.
  - Dimensionality reduction does not just reduce the amount of data, it often brings out the useful part of the data

Dimensionality Reduction

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

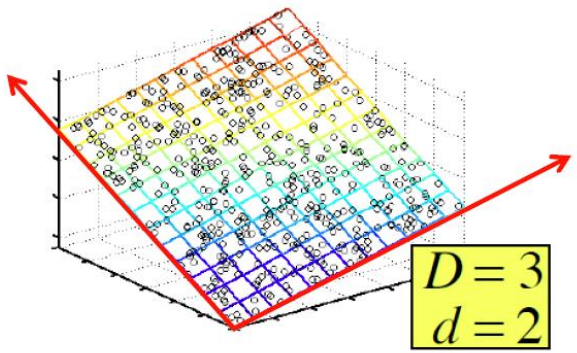


Rather than representing every point with 2 coordinates we represent each point with 1 coordinate (corresponding to the position of the point on the red line).

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

Why Reduce Dimensions?

- **Discover hidden correlations/topics**
  - E.g., in documents, words that occur commonly together
- **Remove redundant and noisy features**
  - E.g., in documents, not all words are useful
- **Interpretation and visualization**
- **Easier storage and processing of the data**



Dimensionality Reduction

- We have already seen a form of dimensionality reduction

- LSH, and random projections reduce the dimension while preserving the distances

Data in the form of a matrix

- We are given n objects and d attributes describing the objects. Each object has d numeric values describing it.

- We will represent the data as a $n \times d$ real matrix A.
  - We can now use tools from linear algebra to process the data matrix

- Our goal is to produce a new $n \times k$ matrix B such that
  - It preserves as much of the information in the original matrix A as possible
  - It reveals something about the structure of the data in A

Example: Document matrices

d terms
(e.g., theorem, proof, etc.)

n documents

$$A$$

A_{ij} = frequency of the **j**-th term in the **i**-th document

Find subsets of terms that bring documents together

Example: Recommendation systems

d movies

n customers

$$A$$

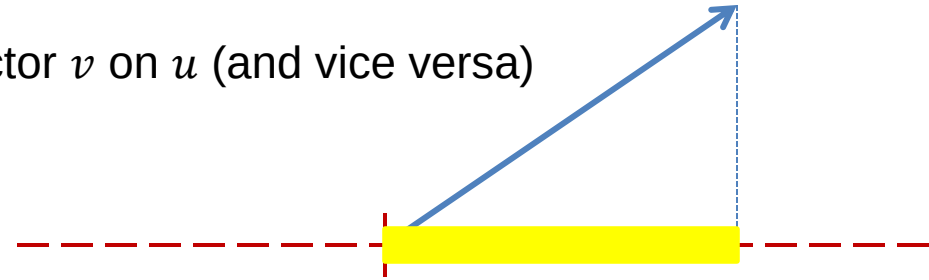
A_{ij} = rating of **j**-th product by the **i**-th customer

Find subsets of movies that capture the behavior or the customers

Linear algebra

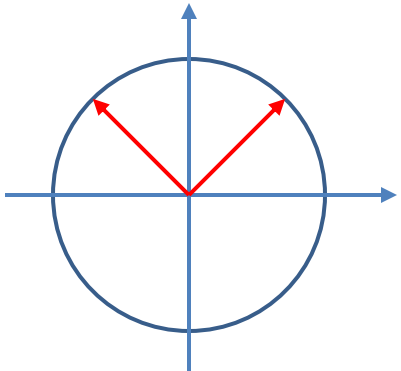
- We assume that vectors are column vectors.
- We use v^T for the transpose of vector v (row vector)

- Dot product: $u^T v$ $(1{\times}n, n{\times}1 \rightarrow 1{\times}1)$
  - The dot product is the projection of vector v on u (and vice versa)
  - $[1,2,3]\begin{bmatrix}4\\1\\2\end{bmatrix} = 12$



- $u^T v = \|v\|\|u\| \cos(u,v)$
  - If $||u|| = 1$ (unit vector) then $u^T v$ is the projection length of v on u

- $[-1,2,3]\begin{bmatrix}4\\-1\\2\end{bmatrix} = 0$: orthogonal vectors
  - Orthonormal vectors: two unit vectors that are orthogonal



Matrices

- An n×m matrix A is a collection of n row vectors and m column vectors

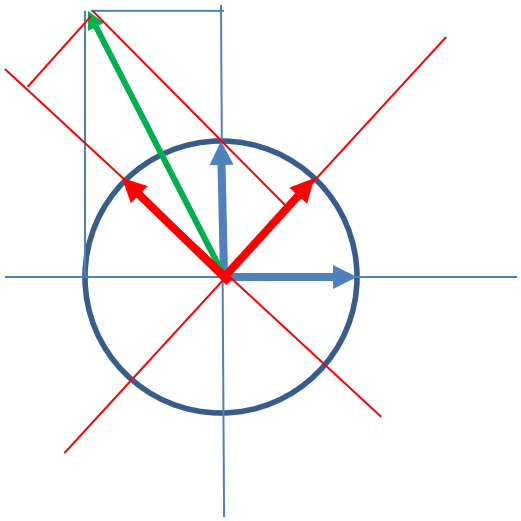
$$A = \begin{bmatrix} | & | & | \\ a_1 & a_2 & a_3 \\ | & | & | \end{bmatrix} \qquad A = \begin{bmatrix} - & \alpha_1^T & - \\ - & \alpha_2^T & - \\ - & \alpha_3^T & - \end{bmatrix}$$

- Matrix-vector multiplication
  - Right multiplication Au: projection of u onto the row vectors of A, or projection of row vectors of A onto u.
  - Left-multiplication $u^T A$: projection of u onto the column vectors of A, or projection of column vectors of A onto u
- Example:

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

Change of basis

- By default a vector is expressed in the axis-aligned basis.
  - For example, for vector [-1,2] we have:
  - $\begin{bmatrix} -1 \\ 2 \end{bmatrix} = -1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
- With a projection we can change the basis over which a vector is expressed.



$$\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} \frac{3\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix}$$

Rank

- Row space of A: The set of vectors that can be written as a linear combination of the rows of A
  - All vectors of the form $v = u^T A$
- Column space of A: The set of vectors that can be written as a linear combination of the columns of A
  - All vectors of the form $v = Au$.
- Rank of A: the number of linearly independent row (or column) vectors
  - These vectors define a basis for the row (or column) space of A
    - All vectors in the row (column) space are linear combinations of the basis vectors
- 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.

Rank-1 matrices

- In a rank-1 matrix, all columns (or rows) are multiples of the same column (or row) vector

$$A = \begin{bmatrix} 1 & 2 & -1 \\ 2 & 4 & -2 \\ 3 & 6 & -3 \end{bmatrix}$$

- All rows are multiples of $r^T = [1,2,-1]$
- All columns are multiples of $c = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$
- External product: uv^T $(n\times 1\ ,\ 1\times m\ \rightarrow\ n\times m)$
  - The resulting $n\times m$ has rank 1: all rows (or columns) are linearly dependent
  - $A = cr^T$

Eigenvectors

- (Right) Eigenvector of matrix A: a vector v such that $Av = \lambda v$
- λ: eigenvalue of eigenvector v

- A square symmetric matrix A of rank r, has r orthonormal eigenvectors $u_1, u_2, \ldots, u_r$ with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_r$.
- Eigenvectors define an orthonormal basis for the column space of A
- We can write:

$$A = U \Lambda U^T$$

Singular Value Decomposition

$$A = U \ \ \Sigma \ \ V^T = [u_1, u_2, \cdots, u_r] \begin{bmatrix} \sigma_1 & & & 0 \\ & \sigma_2 & & \\ & & \ddots & \\ 0 & & & \sigma_r \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_r^T \end{bmatrix}$$

$[n \times m] = [n \times r] \ [r \times r] \ [r \times m]$

r: rank of matrix A

- $\sigma_1, \geq \sigma_2 \geq \cdots \geq \sigma_r$: singular values of matrix A
- $u_1, u_2, \ldots, u_r$: left singular vectors of A
- $v_1, v_2, \ldots, v_r$: right singular vectors of A

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T$$

Singular Value Decomposition

- The left singular vectors are an orthonormal basis for the row space of A.
- The right singular vectors are an orthonormal basis for the column space of A.

- If A has rank r, then A can be written as the sum of r rank-1 matrices

- There are r "linear components" (trends) in A.
  - Linear trend: the tendency of the row vectors of A to align with vector **v**
  - Strength of the i-th linear trend: $||A\boldsymbol{v}_i|| = \boldsymbol{\sigma}_i$

Symmetric matrices

- Special case: A is symmetric positive definite matrix

$$A = \lambda_1 u_1 u_1^T + \lambda_2 u_2 u_2^T + \cdots + \lambda_r u_r u_r^T$$

- $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r \geq 0$: Eigenvalues of A
- $u_1, u_2, \ldots, u_r$: Eigenvectors of A

Singular Values and Eigenvalues

- Singular Value Decomposition

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

- The left singular vectors of A are also the eigenvectors of the (symmetric) matrix AA^T

$$AA^T = U\Sigma^2 U^T$$

- The right singular vectors of A are also the eigenvectors of the (symmetric) matrix $A^T A$

$$A^T A = V\Sigma^2 V^T$$

- The singular values of matrix A are also the square roots of eigenvalues of AA^T and $A^T A$

$$\lambda_i(A^T A) = \lambda_i(AA^T) = \sigma_i^2$$

SVD properties

- Singular Value Decomposition has three useful properties that we will study now:
  - It provides the important (principal) directions (dimensions) in the data – Principal Component Analysis
  - It provides the best low rank approximation for our matrix
  - It minimizes the reconstruction error (squared distance between real data points and their estimates)

Principal Component Analysis

- Goal: reduce the dimensionality while preserving the "information in the data".
- In the new space we want to:
  - Maximize the amount of information
  - Minimize redundancy – remove the redundant dimensions
  - Minimize the noise in the data.

Variability

- Information in the data: variability in the data
  - We measure variability using the covariance matrix.
  - Sample variance for variable X:

$$\sigma_X^2 = \frac{1}{N}\sum_i (x_i - \mu_X)(x_i - \mu_X) = \frac{1}{N}(x - \mu_X)^T(x - \mu_X)$$

  - Sample covariance of variables X and Y

$$\sigma_{XY}^2 = \frac{1}{N}\sum_i (x_i - \mu_X)\,(y_i - \mu_Y) = \frac{1}{N}(x - \mu_X)^T(y - \mu_Y)$$

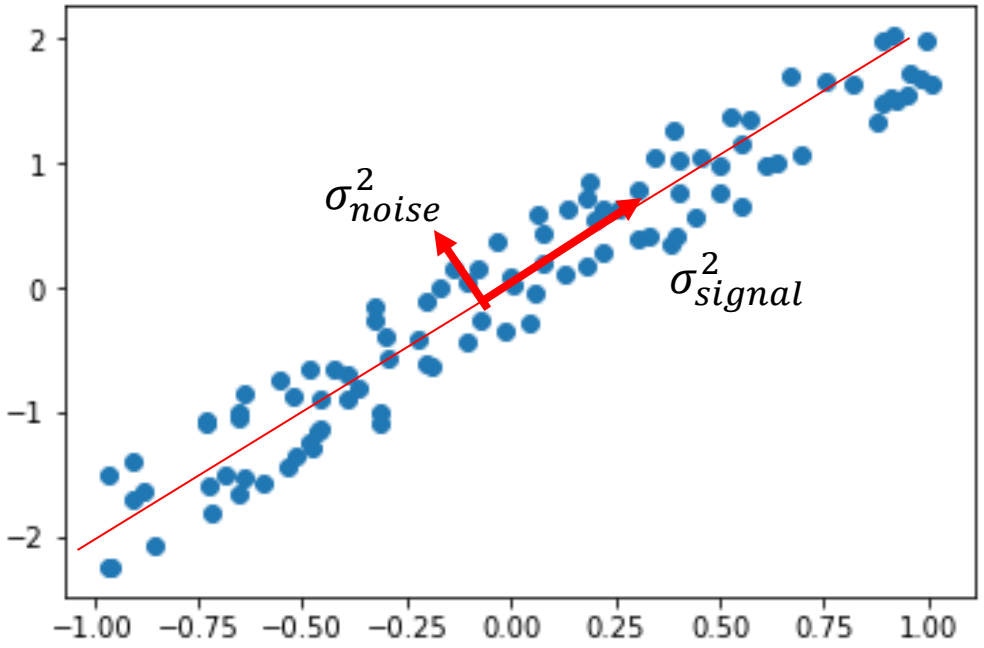
- High variance σ_X^2 means high information in dimension (attribute) X
  - We want to maximize the signal-to-noise ratio $\frac{\sigma_{signal}^2}{\sigma_{noise}^2}$

- High co-variance σ_{XY}^2 means high correlation between attributes X,Y, and thus high redundancy.
  - Ideally we want $\sigma_{XY}^2 = 0$ for all pairs X,Y

Example

- In the data below the data are essentially one-dimensional, but what is the axis we should use?
  - The direction in which the variance is maximized.

The variance along the orthogonal direction is small and captures the noise in the data



Covariance matrix

- We are given the data matrix A, with n rows and m columns, where the rows correspond to data samples over a set of features defined by the columns.
- Remove the mean of each column from the column vectors to get the centered matrix $\bar{A}$
- The matrix $C = \bar{A}^T \bar{A}$ is the covariance matrix of the column vectors of $\bar{A}$.
- We want to change the basis of the data so that the matrix becomes diagonal
  - All the values are in the diagonal and the off-diagonal entries are zero

PCA: Principal Component Analysis

- We will project the rows of matrix $\bar{A}$ onto a new set of attributes (dimensions) such that:
  - The attributes have zero covariance to each other (they are orthogonal)
  - Each attribute captures the most remaining variance in the data, while orthogonal to the existing attributes
    - The first attribute should capture the most variance in the data

- For matrix $\bar{A}$, the variance of the columns of $\bar{A}$ when projected to vector v is given by $\sigma^2 = \left|\left|\bar{A}v\right|\right|^2$
  - The first right singular vector of $\bar{A}$ maximizes σ^2!

PCA and SVD

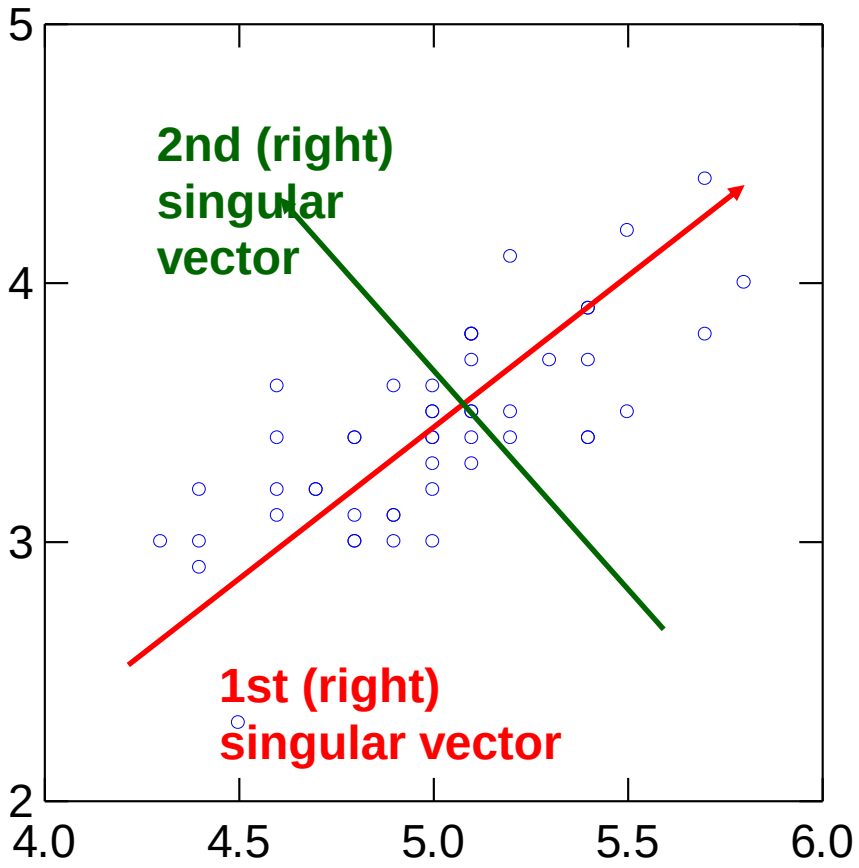
- PCA is a special case of SVD on the centered matrix.
- After projecting the centered matrix $\bar{A}$ to the singular vectors in V we have that the covariance matrix of the new dataset $\bar{A}V$ is:

$$(\bar{A}V)^T(\bar{A}V) = \Sigma$$

- We have achieved to make the matrix diagonal!

- Dimensionality reduction: Don't keep all the singular vectors in V just the k first ones.

PCA



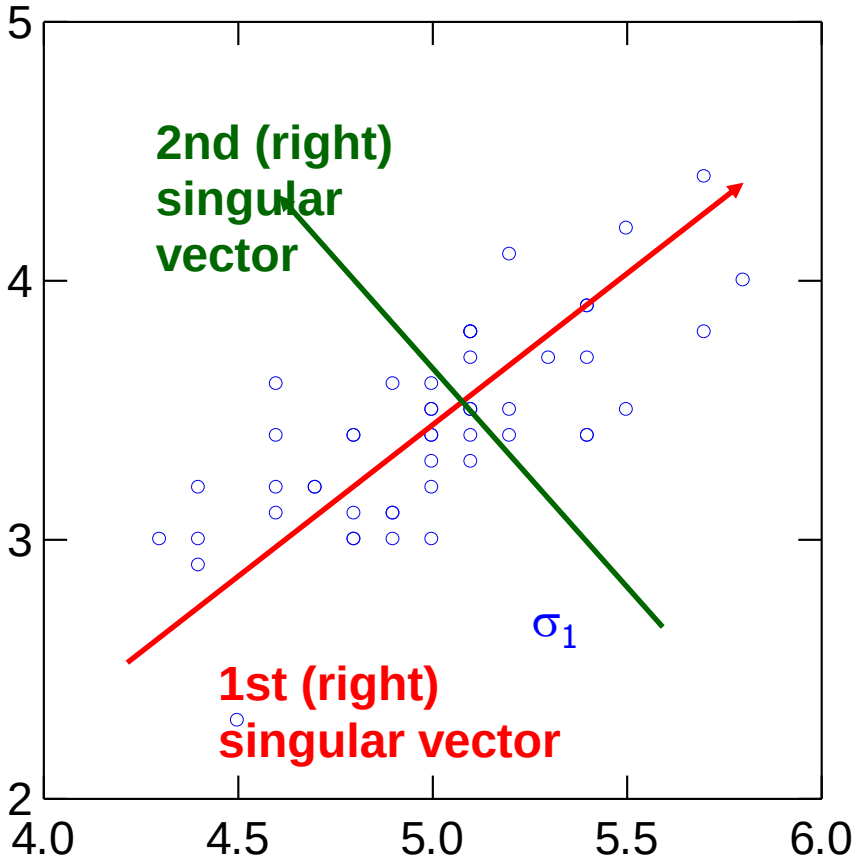
Input: 2-d dimensional points

Output:

1st (right) singular vector: direction of maximal variance,

2nd (right) singular vector: direction of maximal variance, after removing the projection of the data along the first singular vector.

Singular values



σ_1: measures data variance along the first singular vector.

σ_2: measures how much of the data variance is explained by the second singular vector.

Singular values tell us something about the variance

- The variance in the direction of the **k**-th principal component is given by the corresponding singular value σ_k^2

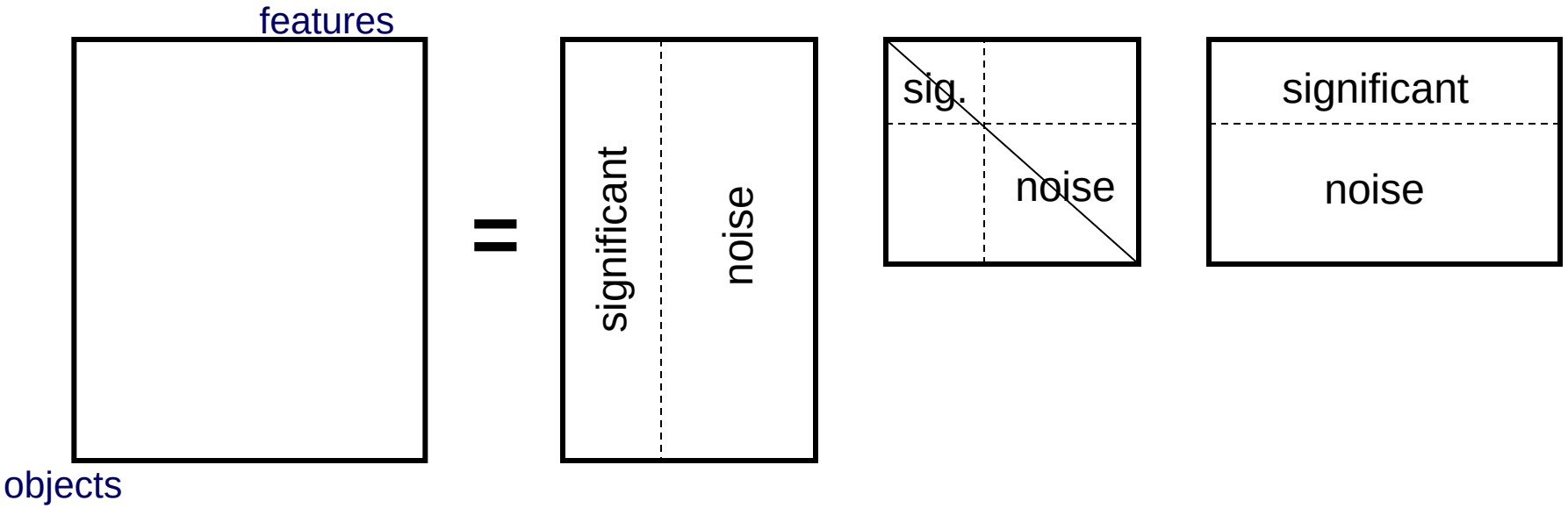
- Singular values can be used to estimate how many components to keep

- ***Rule of thumb:*** keep enough to explain ***85%*** of the variation:

$$\frac{\sum_{j=1}^{k} \sigma_j^2}{\sum_{j=1}^{n} \sigma_j^2} \approx 0.85$$

SVD and Rank-k approximations

A = U Σ V^T



We keep the k most important singular vectors
The matrix $U_k \Sigma_k V_k^T$ is a rank-k approximation of A
The idea is that this is the part that has the useful information and noise is removed
We will show that this is also the best rank-k approximation (closest to the original A)

Example

drugs

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$$

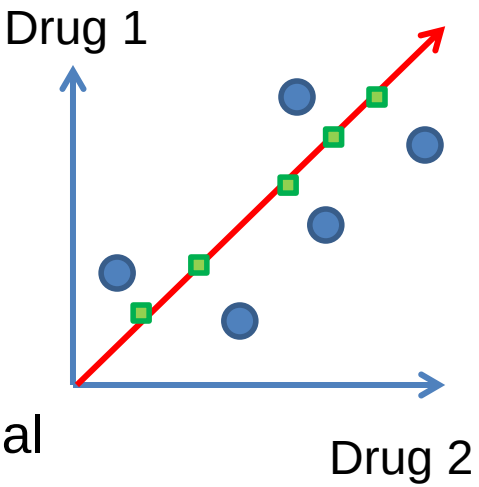
students

legal illegal

a_{ij}: usage of student i of drug j

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

- First right singular vector v_1
  - More or less same weight to all drugs
  - Discriminates heavy from light users
- Second right singular vector
  - Positive values for legal drugs, negative for illegal



SVD for matrix reconstruction

- We will now see how we can use the fact that SVD gives the best rank-k approximation for a data matrix A.
- The idea is that we assume that the “true” matrix is rank-k, and rank is increased due to noise
- We use SVD to find the best rank-k approximation for A, and thus the best approximation of the “true” matrix

An (extreme) example

- User-Movie matrix
  - Blue and Red rows (colums) are linearly dependent

A = 

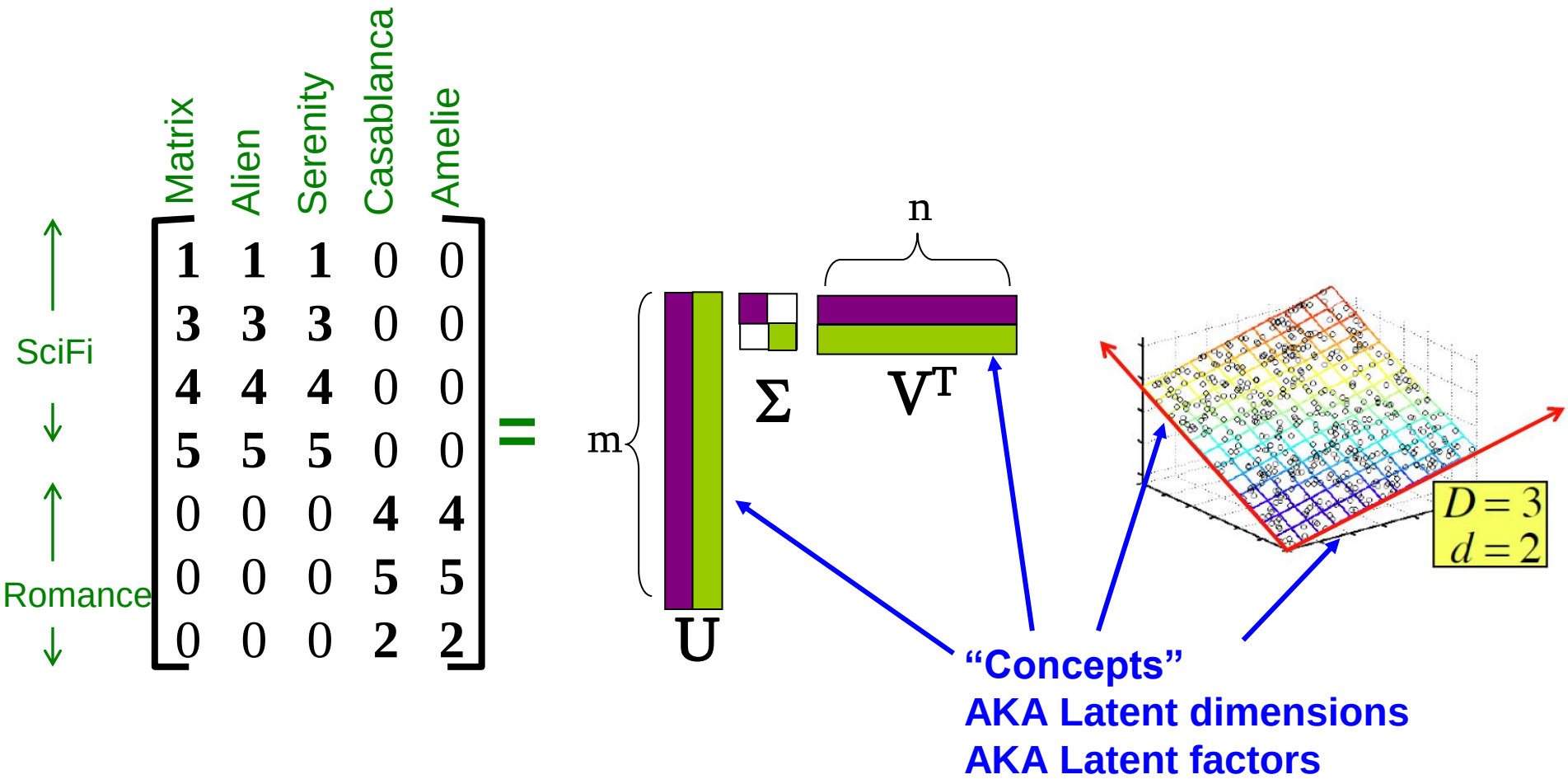
- There are two prototype users (vectors of movies): blue and red
  - To describe the data is enough to describe the two prototypes, and the projection weights for each row

- A is a rank-2 matrix

$$A = [w_1, w_2] \begin{bmatrix} d_1^T \\ d_2^T \end{bmatrix}$$

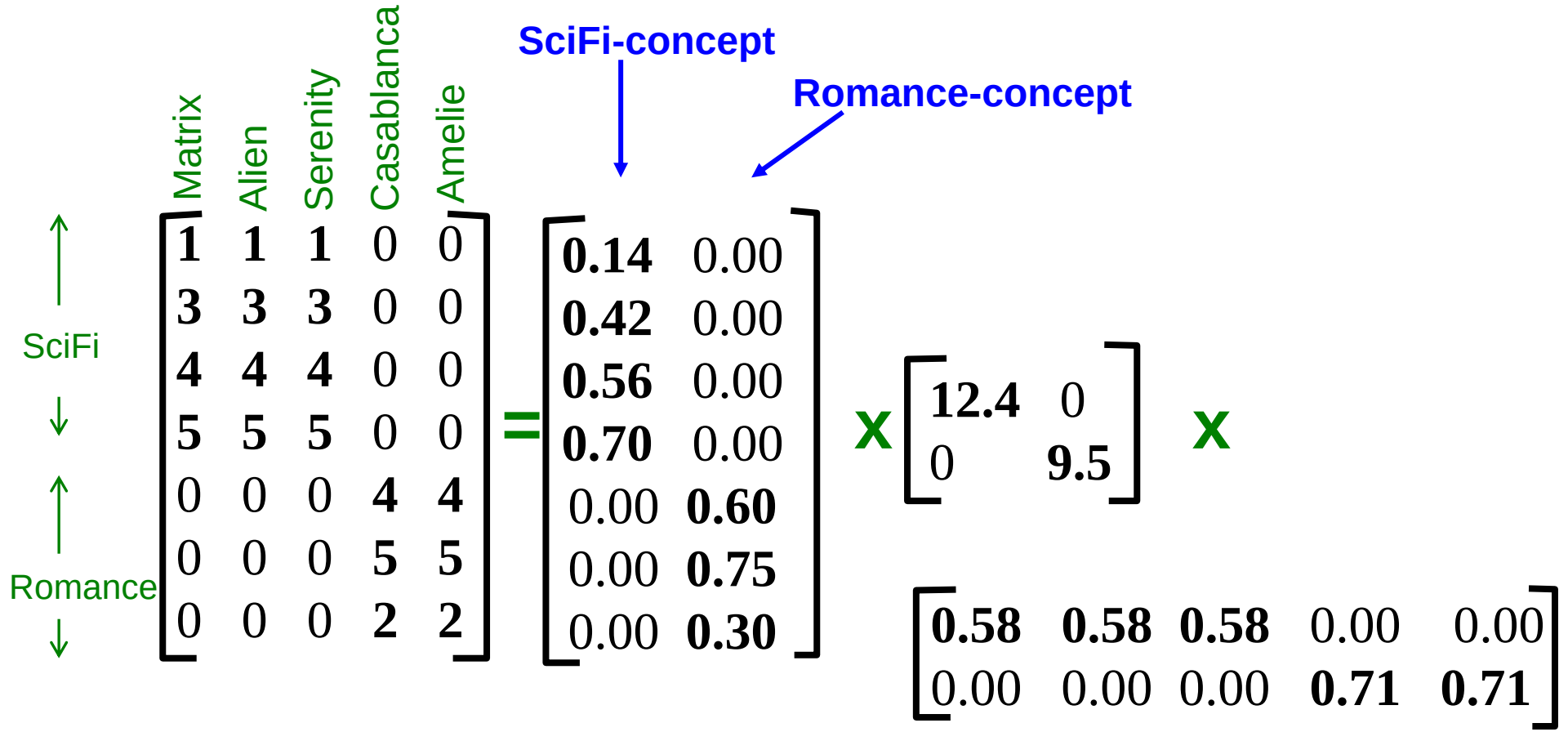
SVD – Example: Users-to-Movies

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



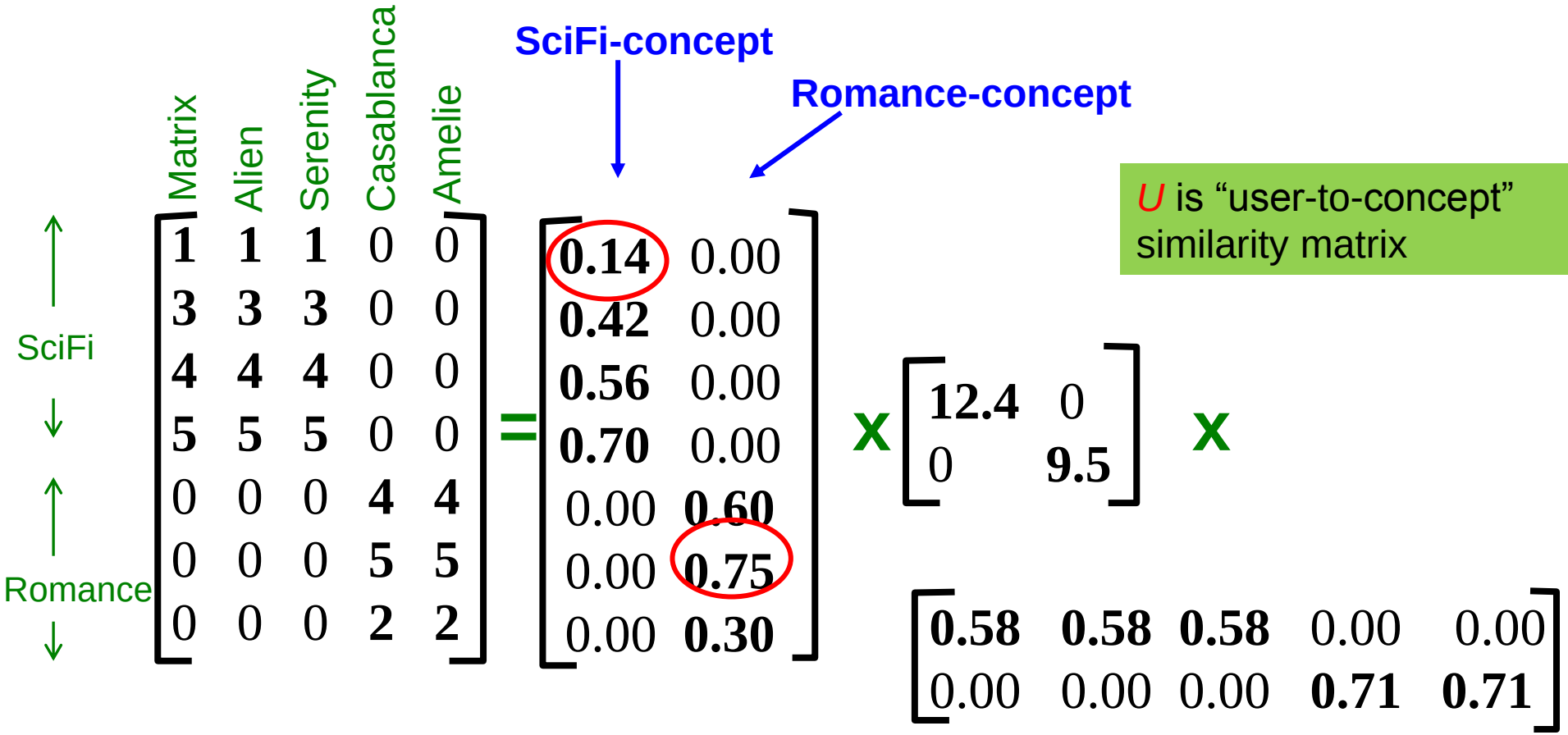
SVD – Example: Users-to-Movies

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



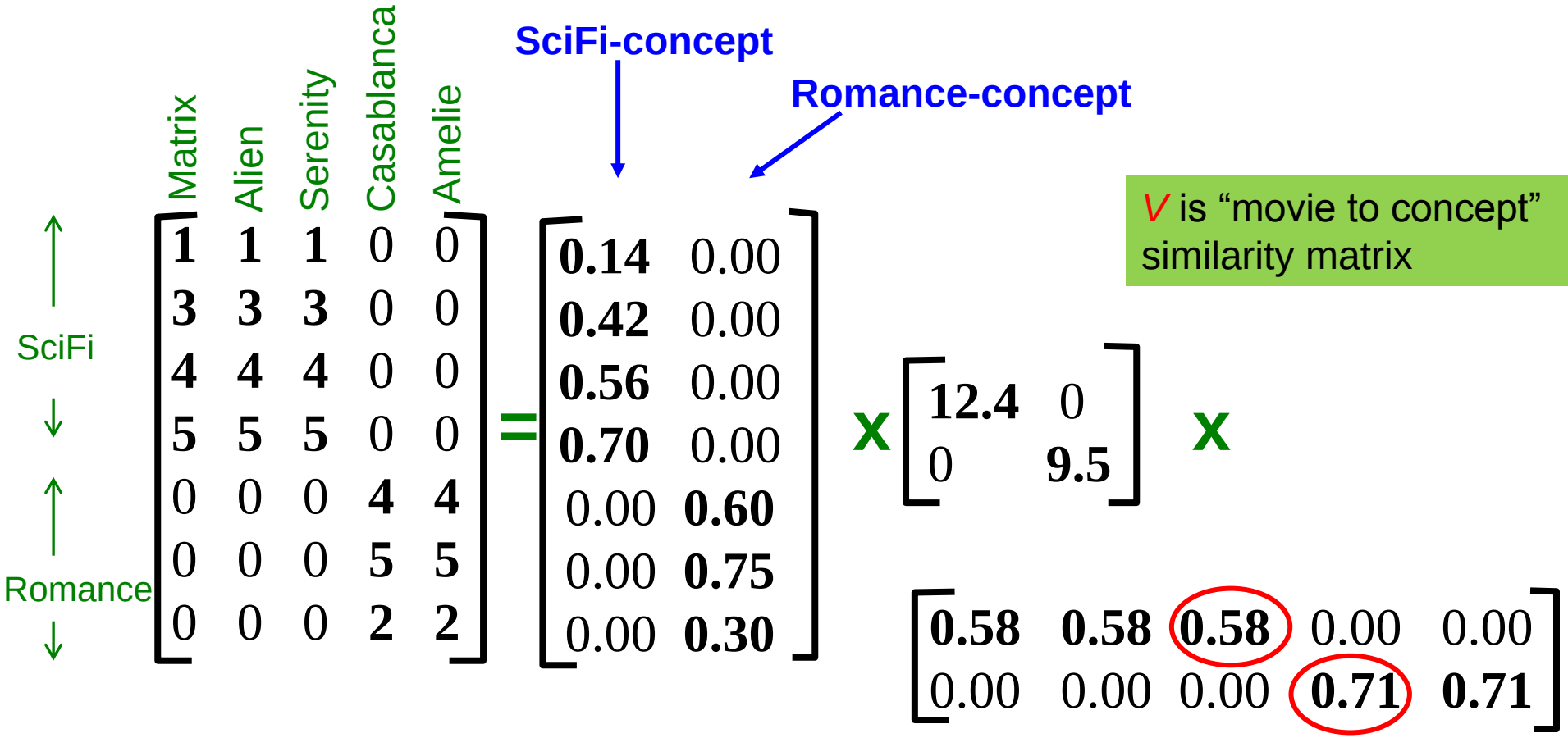
SVD – Example: Users-to-Movies

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



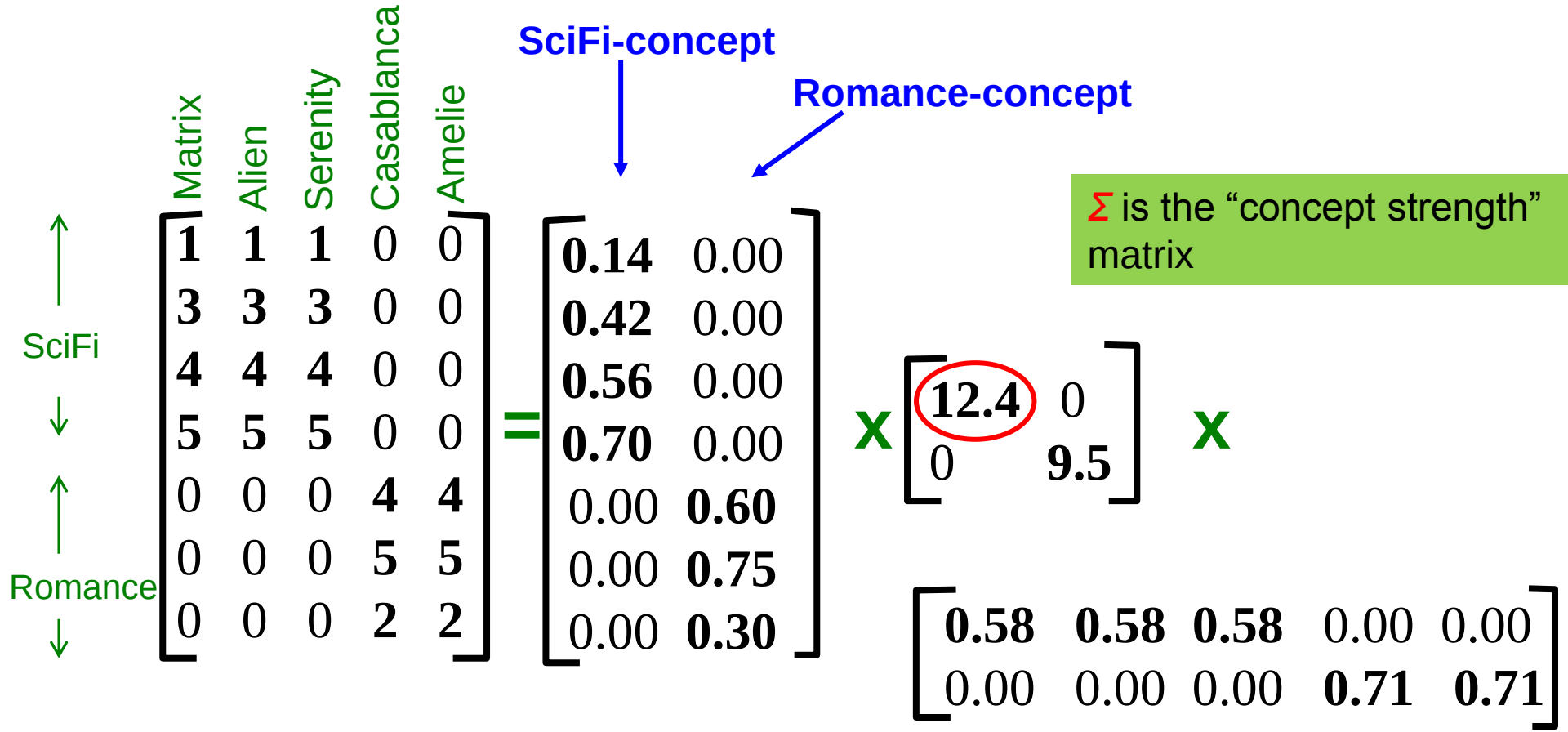
SVD – Example: Users-to-Movies

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



SVD – Example: Users-to-Movies

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



Σ is the "concept strength" matrix

SVD – Example: Users-to-Movies

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

SciFi ↑ / ↓ Romance ↑ / ↓

	Matrix	Alien	Serenity	Casablanca	Amelie
	1	1	1	0	0
	3	3	3	0	0
	4	4	4	0	0
	5	5	5	0	0
	0	0	0	4	4
	0	0	0	5	5
	0	0	0	2	2

=

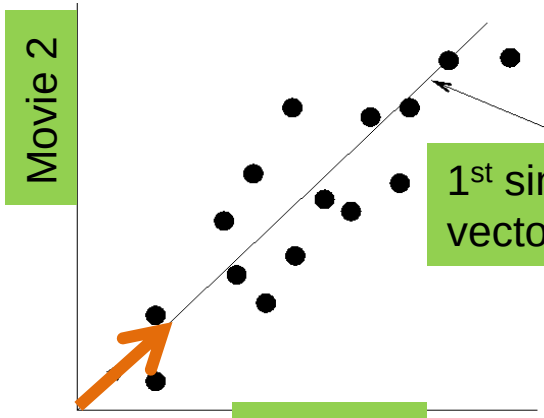
0.14	0.00
0.42	0.00
0.56	0.00
0.70	0.00
0.00	**0.60**
0.00	**0.75**
0.00	**0.30**

x

12.4	0
0	**9.5**

x

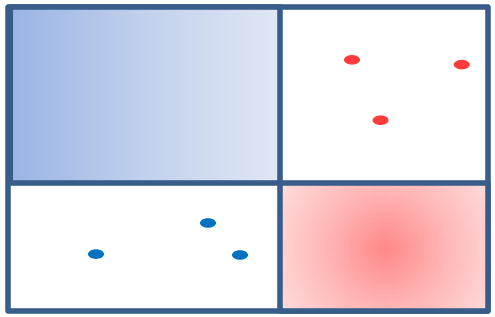
0.58	**0.58**	**0.58**	0.00	0.00
0.00	0.00	0.00	**0.71**	**0.71**



Σ is the "spread (variance)" matrix

An (more realistic) example

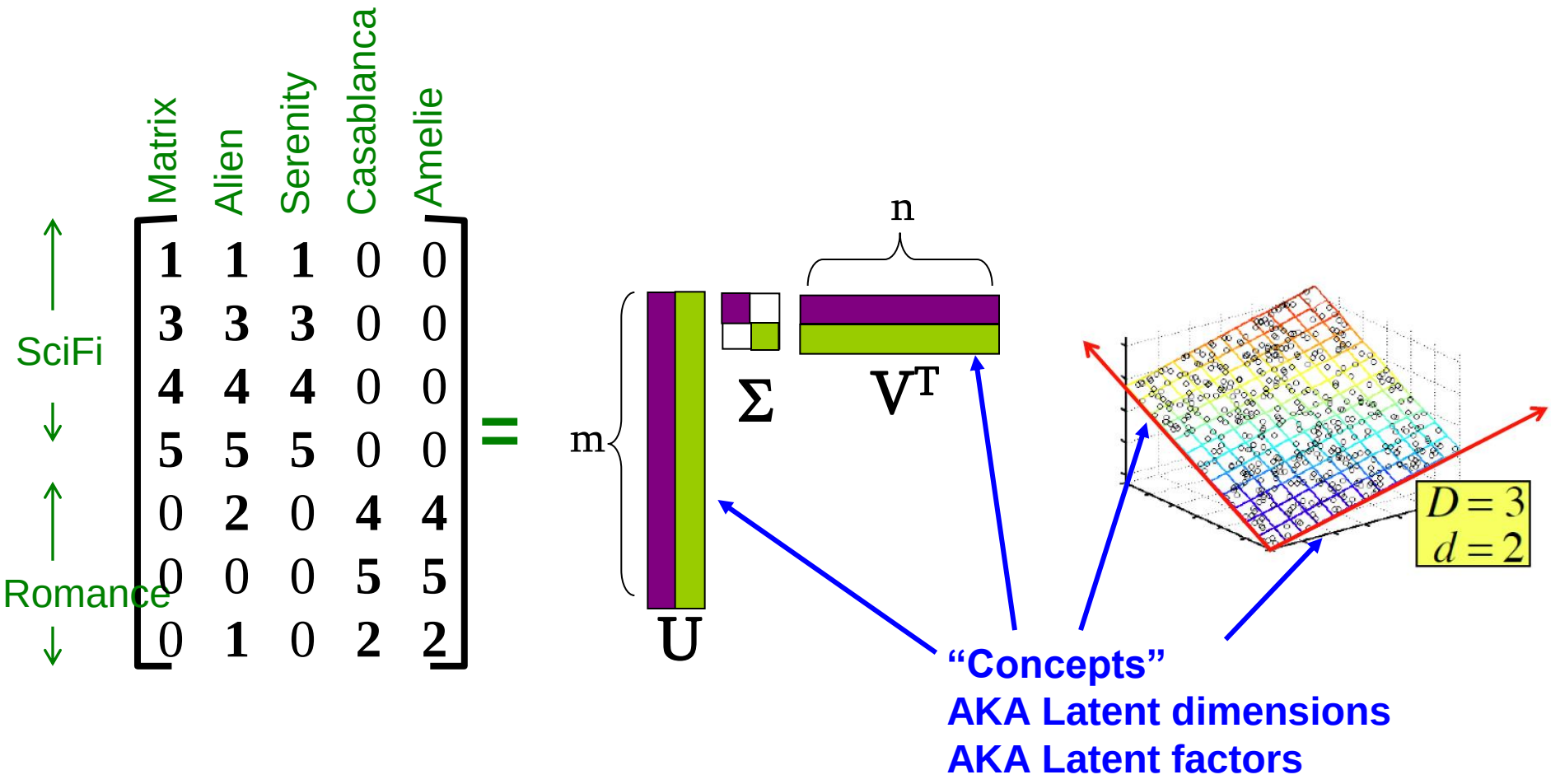
- User-Movie matrix

A = 

- There are two prototype users and movies but they are noisy

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**

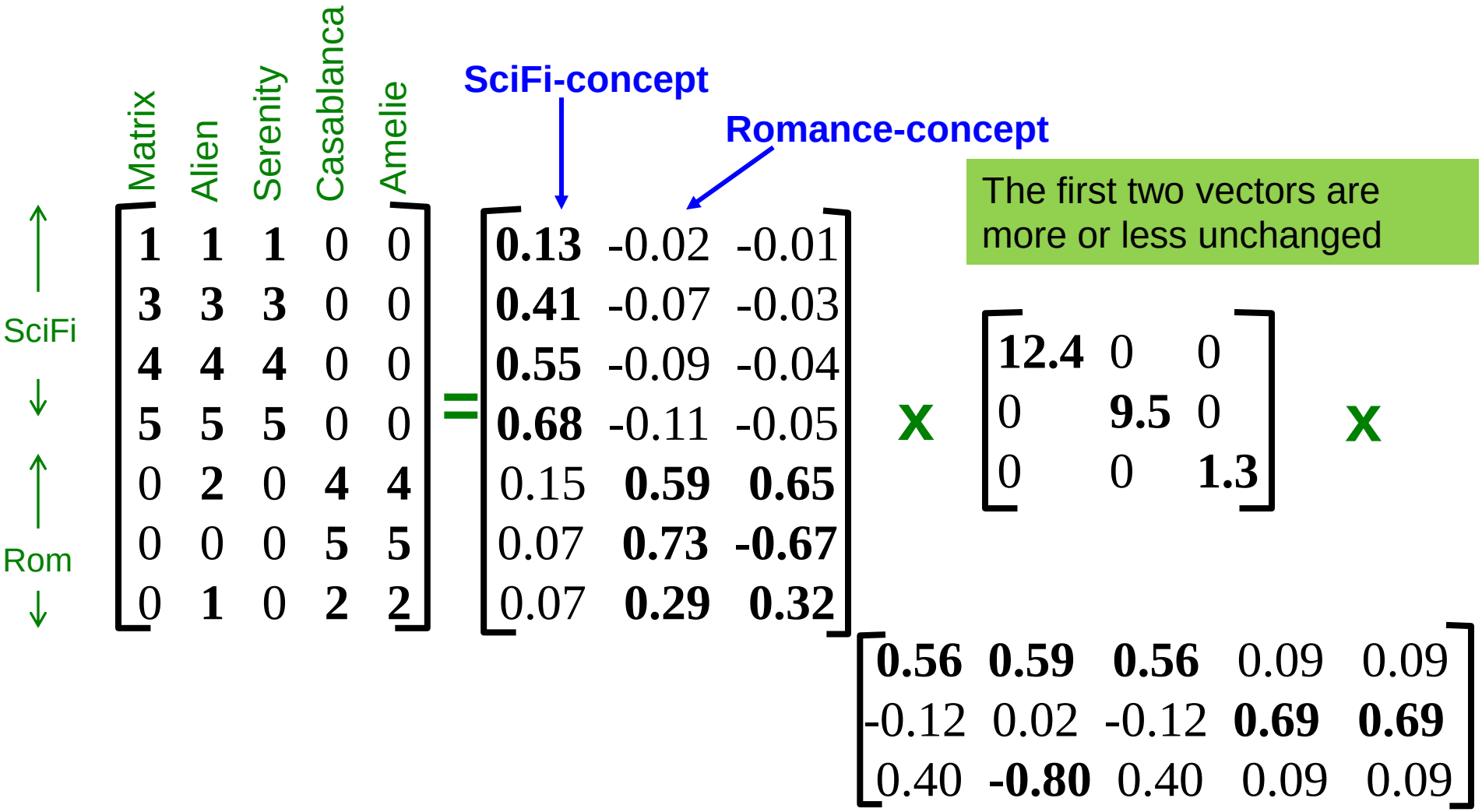
	Matrix	Alien	Serenity	Casablanca	Amelie
SciFi	1	1	1	0	0
	3	3	3	0	0
	4	4	4	0	0
	5	5	5	0	0
Romance	0	2	0	4	4
	0	0	0	5	5
	0	1	0	2	2

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

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

SVD – Example: Users-to-Movies

- **$A = U \Sigma V^T$ - 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
SciFi ↓	3	3	3	0	0
	4	4	4	0	0
	5	5	5	0	0
Rom ↑	0	2	0	4	4
Rom ↓	0	0	0	5	5
	0	1	0	2	2

$=$

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**

$\times$

12.4	0	0
0	**9.5**	0
0	0	**1.3**

$\times$

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

The third vector has a very low singular value

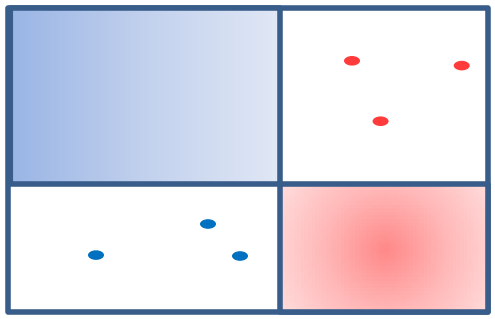
SVD - Interpretation #1

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

- $\boldsymbol{U}$: user-to-concept similarity matrix
- $\boldsymbol{V}$: movie-to-concept similarity matrix
- Σ: its diagonal elements:
  'strength' of each concept

Rank-k approximation

- In this User-Movie matrix

A = 

- We have more than two singular vectors, but the strongest ones are still about the two types.
  - The third models the noise in the data
- By keeping the two strongest singular vectors we obtain most of the information in the data.
  - This is a rank-2 approximation of the matrix A

Rank-**k** approximations (A_k)

$$\left(A_k \right) = \left(U_k \right) \cdot \left(\Sigma_k \right) \cdot \left(V_k^T \right)$$

n x d **n x k** **k x k** **k x d**

$\boldsymbol{U_k}$ $(\boldsymbol{V_k})$: orthogonal matrix containing the top ***k*** left (right) singular vectors of **A**.

$\boldsymbol{\Sigma_k}$**:** diagonal matrix containing the top ***k*** singular values of **A**

$\mathbf{A_k}$ is an approximation of **A**

$\mathbf{A_k}$ is the **best** approximation of **A**

SVD as an optimization

- The rank-k approximation matrix A_k produced by the top-k singular vectors of A minimizes the Frobenious norm of the difference with the matrix A

$$A_k = \arg \max_{B:rank(B)=k} \|A - B\|_F^2$$

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

Explanation: The (i,j) cell in A_k is close on average with the (i,j) cell of A

What does this mean?

- We can project the row (and column) vectors of the matrix A into a k-dimensional space and preserve most of the information
- (Ideally) The k dimensions reveal latent features/aspects/topics of the term (document) space.
- (Ideally) The A_k approximation of matrix A, contains all the useful information, and what is discarded is noise

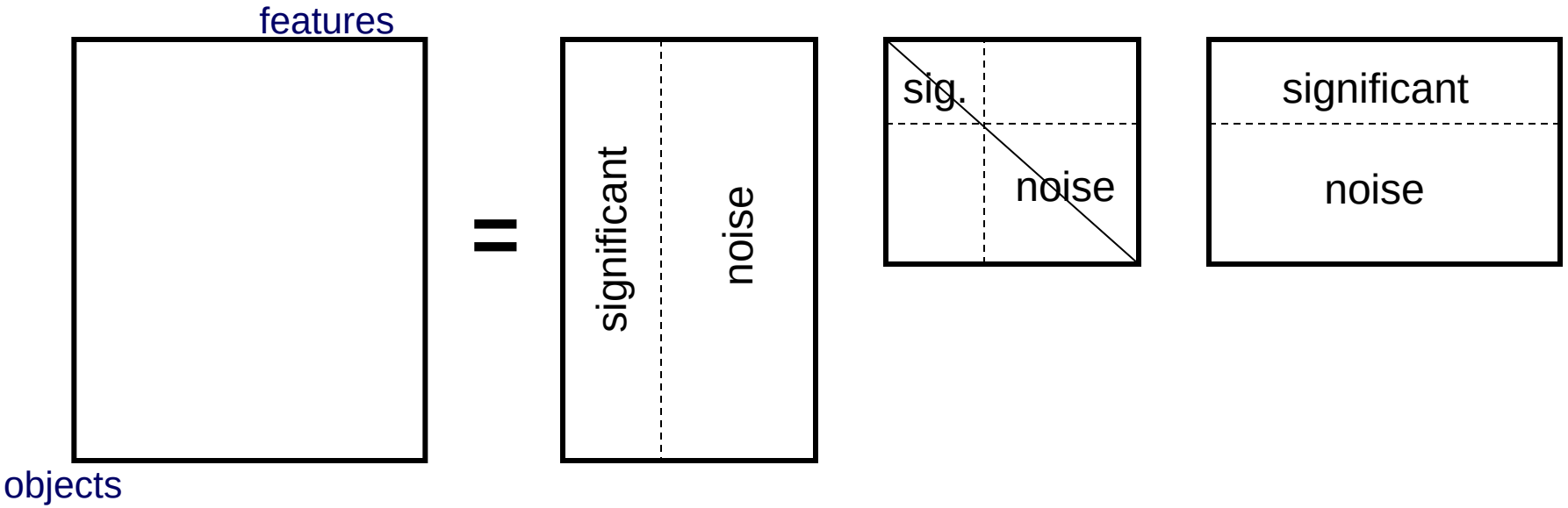
Latent factor model

- Rows (columns) are linear combinations of k latent factors
  - E.g., in our extreme document example there are two factors
- Some noise is added to this rank-k matrix resulting in higher rank

- SVD retrieves the latent factors (hopefully).

SVD and Rank-k approximations

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



Example

More details

- Q: **How exactly is dim. reduction done?**
- A: **Compute SVD**

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

Example

More details

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

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

Example

More details

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

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

Example

More details

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

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

Example

More details

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

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

Example

More details

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

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

Frobenius norm:

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

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

Application: Recommender systems

- Data: Users rating movies
  - Sparse and often noisy
- Assumption: There are k basic user profiles, and each user is a linear combination of these profiles
  - E.g., action, comedy, drama, romance
  - Each user is a weighted combination of these profiles
  - The "true" matrix has rank k
- If we had the matrix A with all ratings of all users for all movies, the matrix A_k would tell us the true preferences of the users for the movies

Model-based Recommendation Systems

- What we observe is a noisy, and incomplete version of this matrix $\tilde{A}$
- Given matrix $\tilde{A}$ and we would like to get the missing ratings that A_k would produce

- Algorithm: compute the rank-k approximation $\tilde{A}_k$ of and matrix $\tilde{A}$ predict for user u and movie m, the value $\tilde{A}_k[m, u]$.
  - The rank-k approximation $\tilde{A}_k$ is provably close to A_k

- Model-based collaborative filtering

Example

Missing ratings and noise

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 3 & 3 & 0 & 0 \\ 4 & 4 & 0 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} \mathbf{0.14} & -0.06 & -0.04 \\ \mathbf{0.30} & -0.11 & \mathbf{-0.61} \\ \mathbf{0.43} & -0.16 & \mathbf{0.76} \\ \mathbf{0.74} & -0.31 & -0.18 \\ 0.15 & \mathbf{0.53} & 0.02 \\ 0.07 & \mathbf{0.70} & -0.03 \\ 0.07 & \mathbf{0.27} & 0.01 \end{bmatrix} \times \begin{bmatrix} \mathbf{12.4} & 0 & 0 \\ 0 & \mathbf{9.5} & 0 \\ 0 & 0 & \mathbf{1.3} \end{bmatrix} \times$$

$$\begin{bmatrix} \mathbf{0.51} & \mathbf{0.66} & \mathbf{0.44} & 0.23 & 0.23 \\ \mathbf{-0.24} & -0.13 & -0.21 & \mathbf{0.66} & \mathbf{0.66} \\ 0.59 & 0.08 & \mathbf{-0.80} & 0.01 & 0.01 \end{bmatrix}$$

Example

Missing ratings and noise

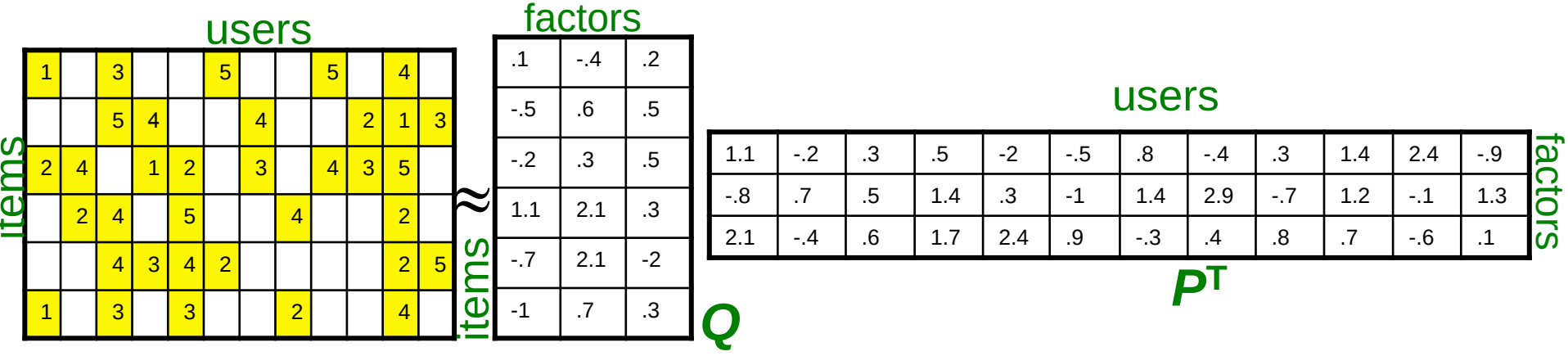
$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 3 & 3 & 0 & 0 \\ 4 & 4 & 0 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} \mathbf{0.14} & -0.06 & -0.04 \\ \mathbf{0.30} & -0.11 & \mathbf{-0.61} \\ \mathbf{0.43} & -0.16 & \mathbf{0.76} \\ \mathbf{0.74} & -0.31 & -0.18 \\ 0.15 & \mathbf{0.53} & 0.02 \\ 0.07 & \mathbf{0.70} & -0.03 \\ 0.07 & \mathbf{0.27} & 0.01 \end{bmatrix} \times \begin{bmatrix} \mathbf{12.4} & 0 & 0 \\ 0 & \mathbf{9.5} & 0 \\ 0 & 0 & \mathbf{1.3} \end{bmatrix} \times \begin{bmatrix} \mathbf{0.51} & \mathbf{0.66} & \mathbf{0.44} & 0.23 & 0.23 \\ \mathbf{-0.24} & -0.13 & -0.21 & \mathbf{0.66} & \mathbf{0.66} \\ 0.59 & 0.08 & \mathbf{-0.80} & 0.01 & 0.01 \end{bmatrix}$$

Example

- Reconstruction of missing ratings

$$\begin{bmatrix} \mathbf{0.96} & \mathbf{1.14} & \mathbf{0.82} & -0.01 & -0.01 \\ \mathbf{1.94} & \mathbf{2.32} & \mathbf{1.66} & 0.07 & 0.07 \\ \mathbf{2.77} & \mathbf{3.32} & \mathbf{2.37} & 0.08 & 0.08 \\ \mathbf{4.84} & \mathbf{5.74} & \mathbf{4.14} & -0.08 & 0.08 \\ 0.40 & \mathbf{1.42} & 0.33 & \mathbf{4.06} & \mathbf{4.06} \\ -0.42 & 0.63 & -0.38 & \mathbf{4.92} & \mathbf{4.92} \\ 0.20 & \mathbf{0.71} & 0.16 & \mathbf{2.03} & \mathbf{2.03} \end{bmatrix}$$

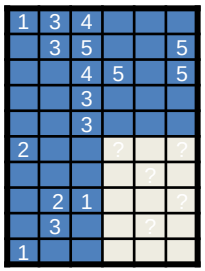
Latent Factor Models



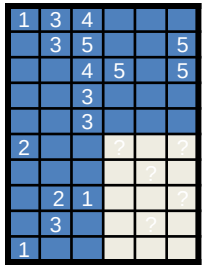
- **SVD also considers entries that are missing!**
- **Use specialized methods to find *P*, *Q***
  - $\min_{P,Q} \sum_{(i,x)\in R} \left(r_{xi} - q_i^T \cdot p_x\right)^2$ $\qquad \hat{r}_{xi} = q_i^T \cdot p_x$
  - **Note:**
    - We don't require cols of ***P, Q*** to be orthogonal/unit length
    - ***P, Q*** map users/movies to a latent space

Computing the latent factors

- **Want to minimize SSE for unseen test data**
- **Idea: Minimize SSE on training data**
  - Want large ***k*** (# of factors) to capture all the signals
  - But, **SSE** on **test data** begins to rise for ***k*** > 2
- This is a classical example of **overfitting:**
  - With too much freedom (too many free parameters) the model starts fitting noise
    - That is it fits too well the training data and thus **not generalizing** well to unseen test data



Dealing with Missing Entries



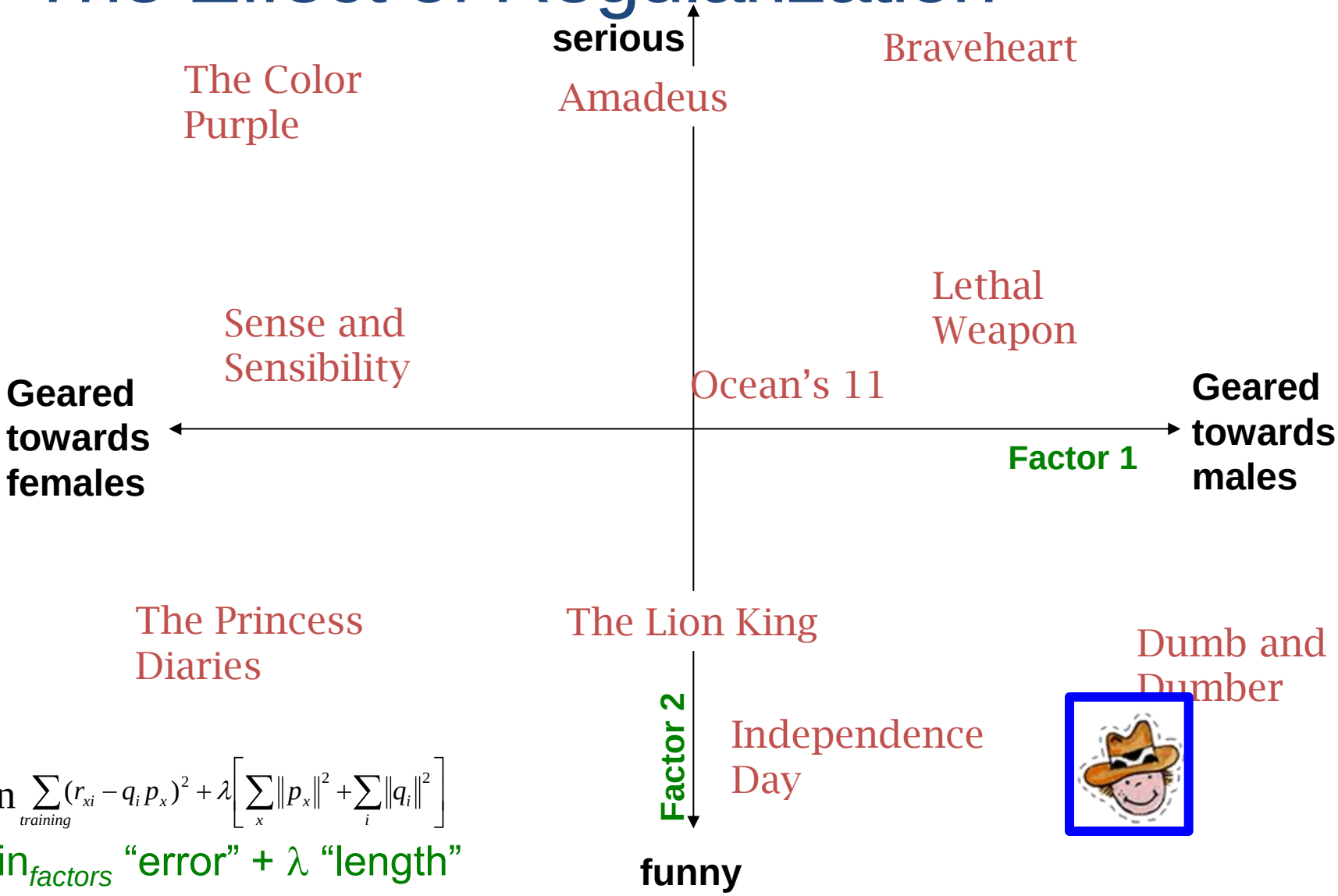
- **To solve overfitting we introduce regularization:**
  - Allow rich model where there are sufficient data
  - Shrink aggressively where data are scarce

$$\min_{P,Q} \underbrace{\sum_{training} (r_{xi} - q_i^T p_x)^2}_{\text{“error”}} + \underbrace{\left[\lambda_1 \sum_x \|p_x\|^2 + \lambda_2 \sum_i \|q_i\|^2 \right]}_{\text{“length”}}$$

λ_1, λ_2 … user set regularization parameters

Note: We do not care about the “raw” value of the objective function, but we care in P,Q that achieve the minimum of the objective

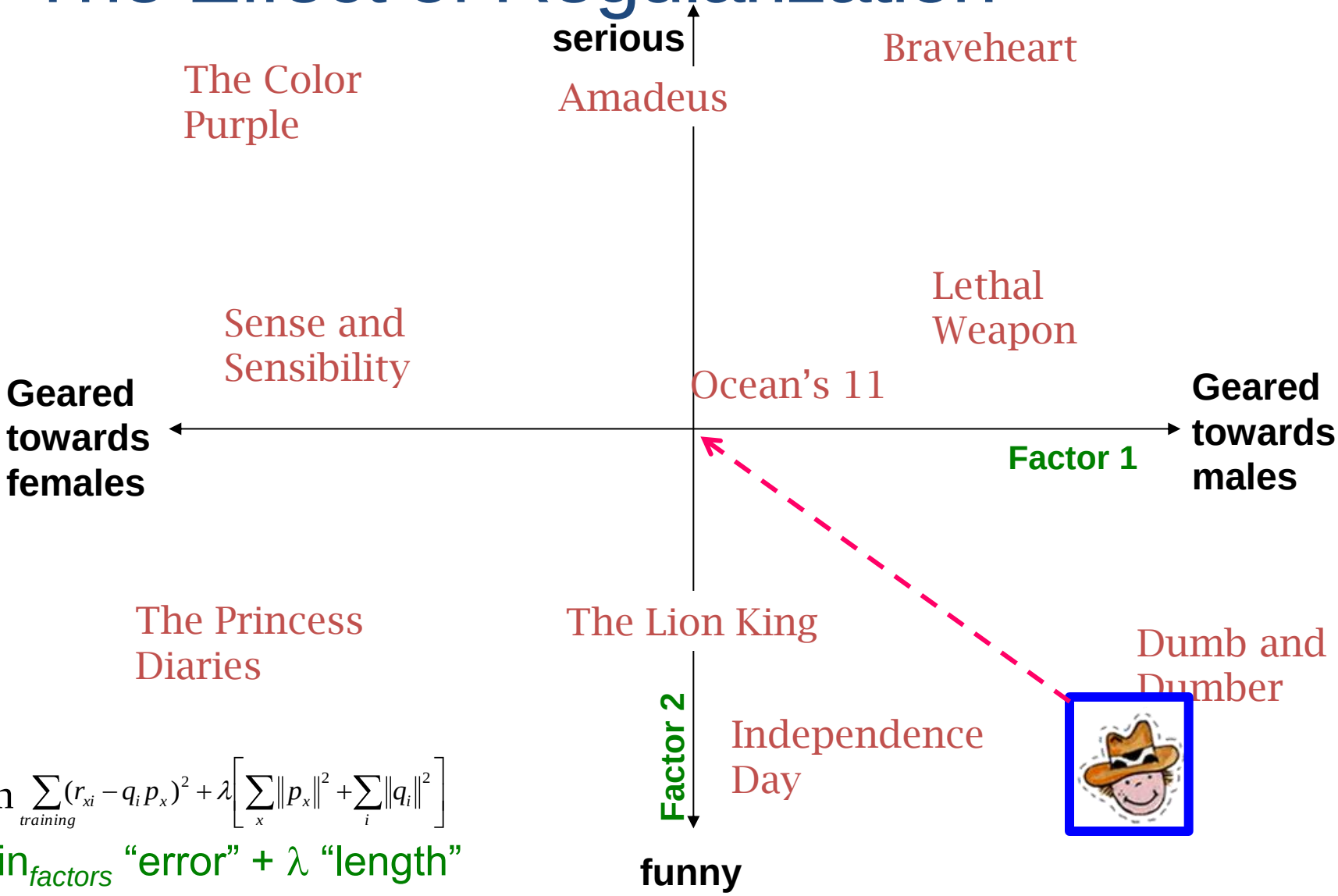
The Effect of Regularization



$$\min_{P,Q} \sum_{training} (r_{xi} - q_i p_x)^2 + \lambda \left[\sum_x \|p_x\|^2 + \sum_i \|q_i\|^2 \right]$$

$\min_{factors}$ "error" + λ "length"

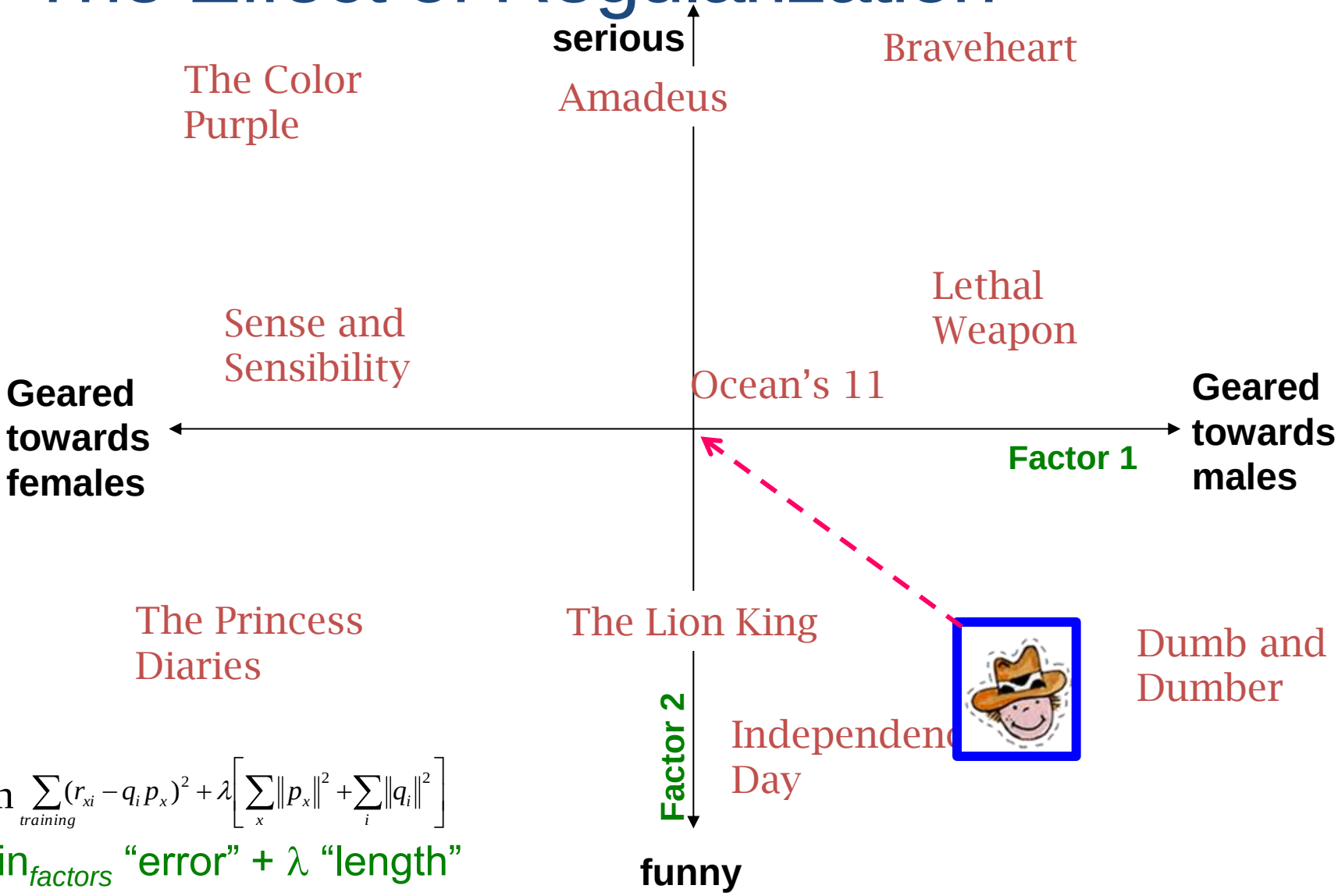
The Effect of Regularization



$$\min_{P,Q} \sum_{training} (r_{xi} - q_i p_x)^2 + \lambda \left[\sum_x \|p_x\|^2 + \sum_i \|q_i\|^2 \right]$$

$\min_{factors}$ "error" + λ "length"

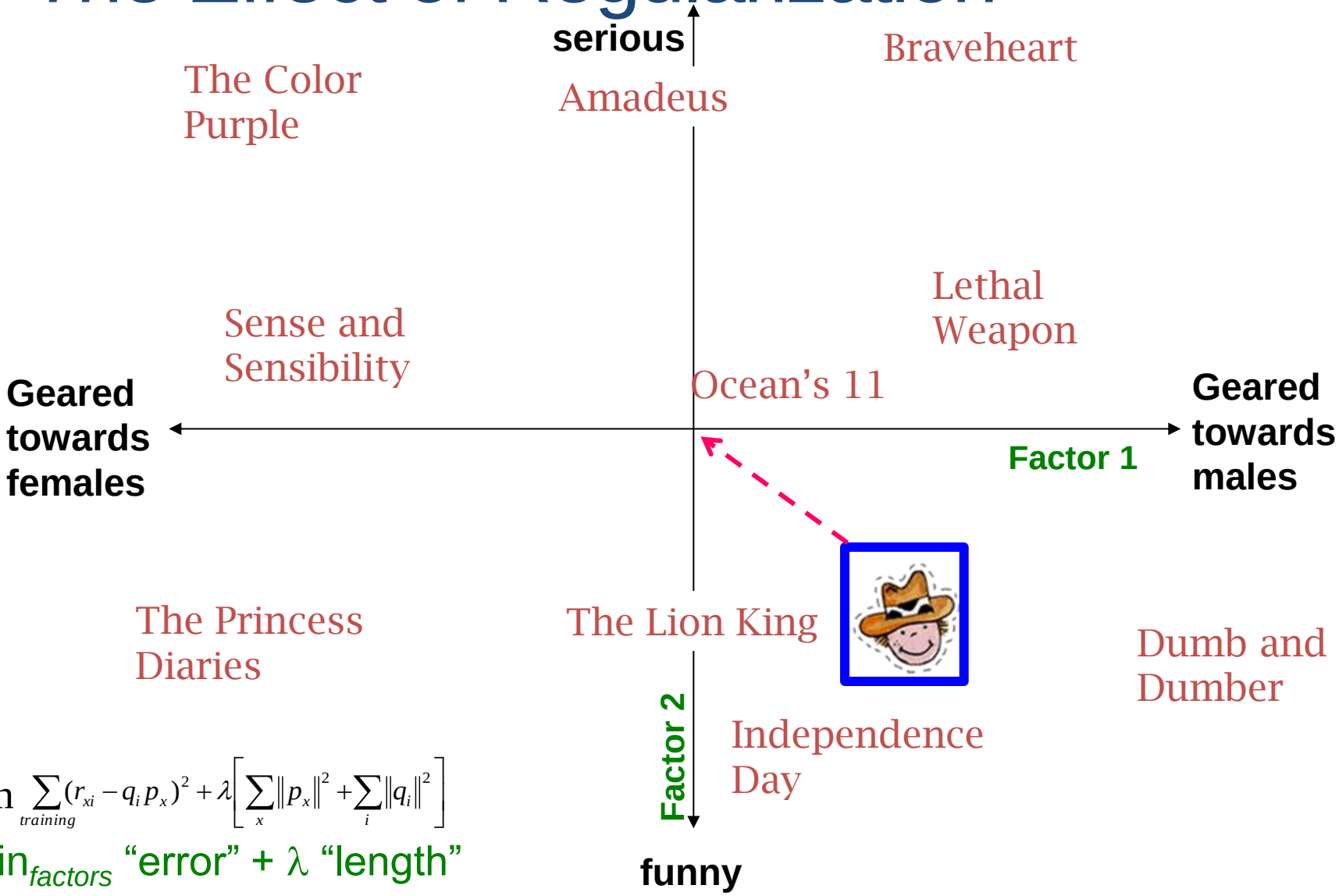
The Effect of Regularization



$$\min_{P,Q} \sum_{training} (r_{xi} - q_i p_x)^2 + \lambda \left[\sum_x \|p_x\|^2 + \sum_i \|q_i\|^2 \right]$$

$\min_{factors}$ "error" + λ "length"

The Effect of Regularization



$$\min_{P,Q} \sum_{training} (r_{xi} - q_i p_x)^2 + \lambda \left[\sum_x \|p_x\|^2 + \sum_i \|q_i\|^2 \right]$$

$\min_{factors}$ "error" + λ "length"

Latent factors

- To find the P,Q that minimize the error function we can use (stochastic) gradient descent

- We can define different latent factor models that apply the same idea in different ways
  - Probabilistic/Generative models.

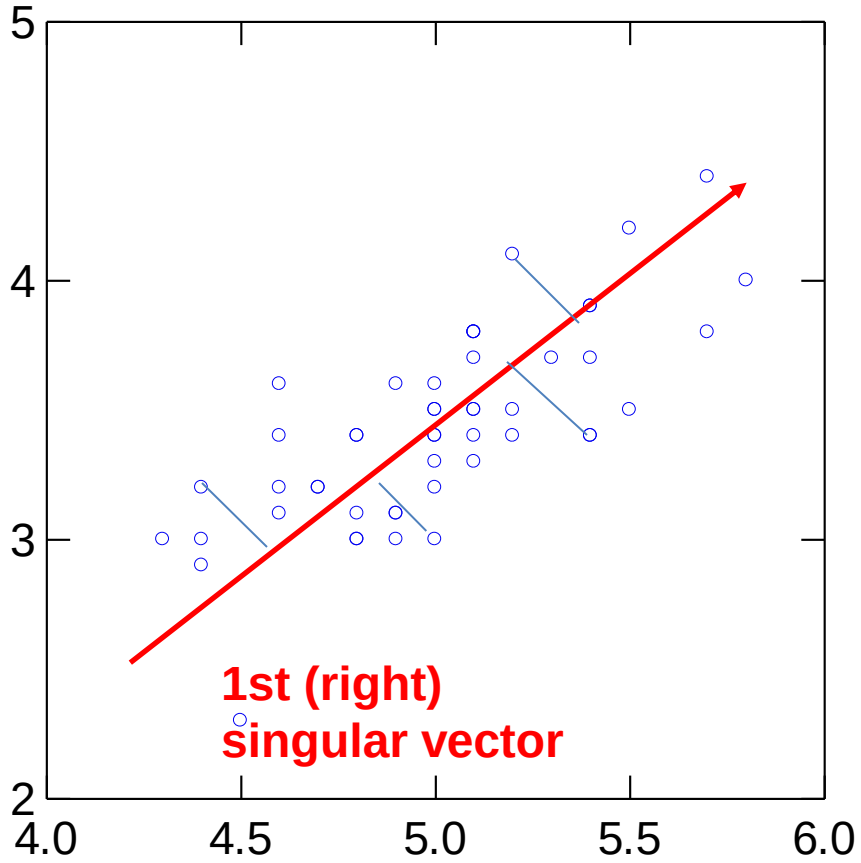
- The latent factor methods work well in practice, and they are employed by most sophisticated recommendation systems

Another Application

- Latent Semantic Indexing (LSI):
  - Apply PCA on the document-term matrix, and index the k-dimensional vectors
  - When a query comes, project it onto the k-dimensional space and compute cosine similarity in this space
  - Principal components capture main topics, and enrich the document representation

Another property of PCA/SVD

- The chosen vectors are such that minimize the sum of square differences between the data vectors and the low-dimensional projections



SVD is “**the Rolls-Royce and the Swiss Army Knife of Numerical Linear Algebra.**”*

*Dianne O’Leary, MMDS ’06

Computation of eigenvectors

- Consider a symmetric square matrix M
- Power-method:
  - Start with the vector v of all 1's
  - Compute $v = Mv$
  - Normalize by the length of v
  - Repeat until the vector does not change

- This will give us the first eigenvector.
- The first eigenvalue is $\lambda = v^T M v$

- For the second one, compute the first eigenvector of the matrix $M^* = M - \lambda v v^T$

Singular Values and Eigenvalues

- The left singular vectors of A are also the eigenvectors of AA^T
- The right singular vectors of A are also the eigenvectors of A^TA
- The singular values of matrix A are also the square roots of eigenvalues of AA^T and A^TA

Computing singular vectors

- Compute the eigenvectors and eigenvalues of the matrices MM^T and M^TM