

Online Social Networks and Media

Recommender Systems and Social Recommendations

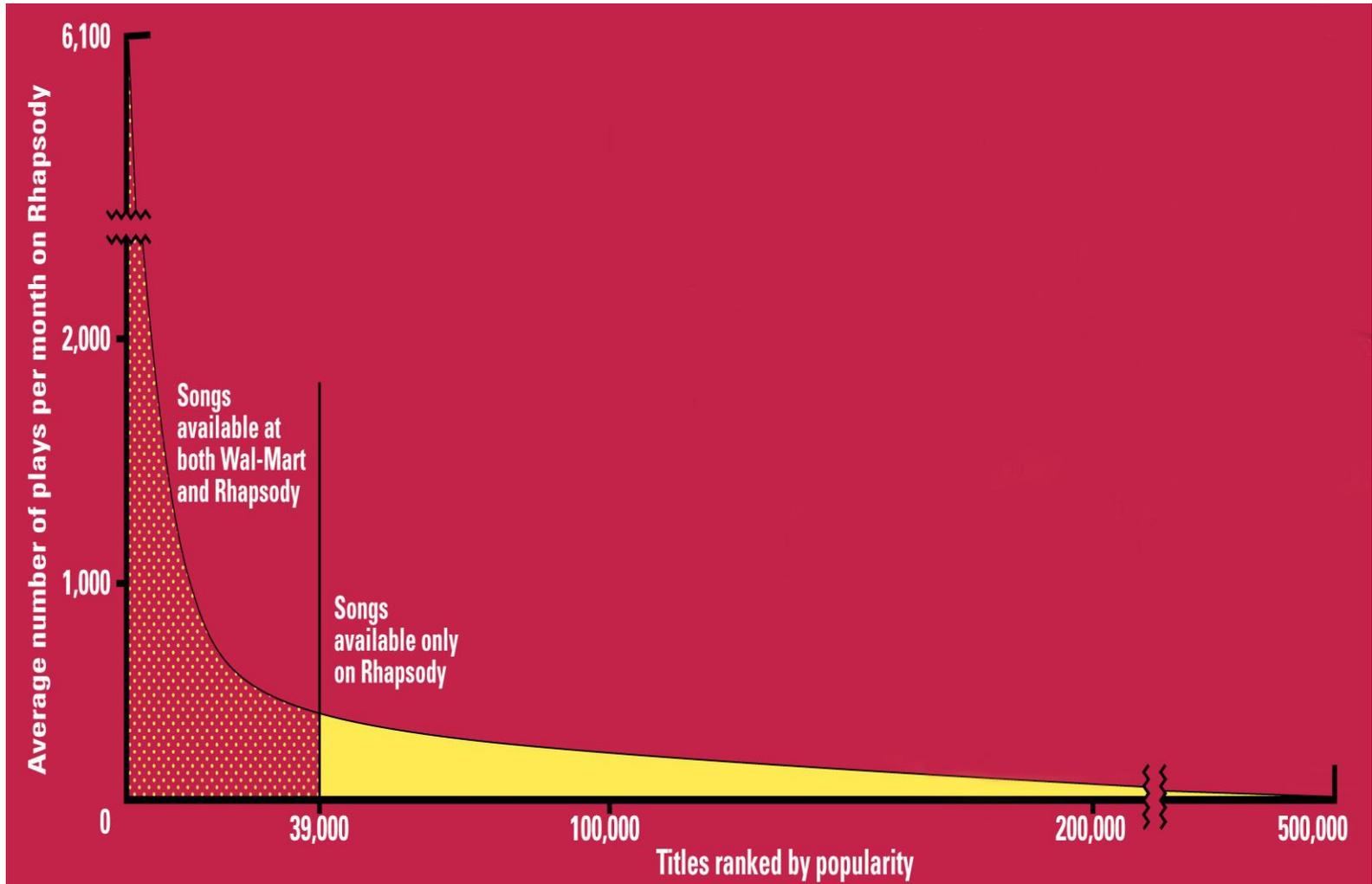
Thanks to:

Jure Leskovec, Anand Rajaraman, Jeff Ullman

Recommendation Systems

- **Recommendation** systems
 - When a user buys an **item** (initially books) we want to recommend other items that the user may like
 - When a user rates a **movie**, we want to recommend movies that the user may like
 - When a user likes a **song**, we want to recommend other songs that they may like
- A big success of data mining
- Exploits the **long tail**
 - How **Into Thin Air** made **Touching the Void** popular

The Long Tail



Source: Chris Anderson (2004)

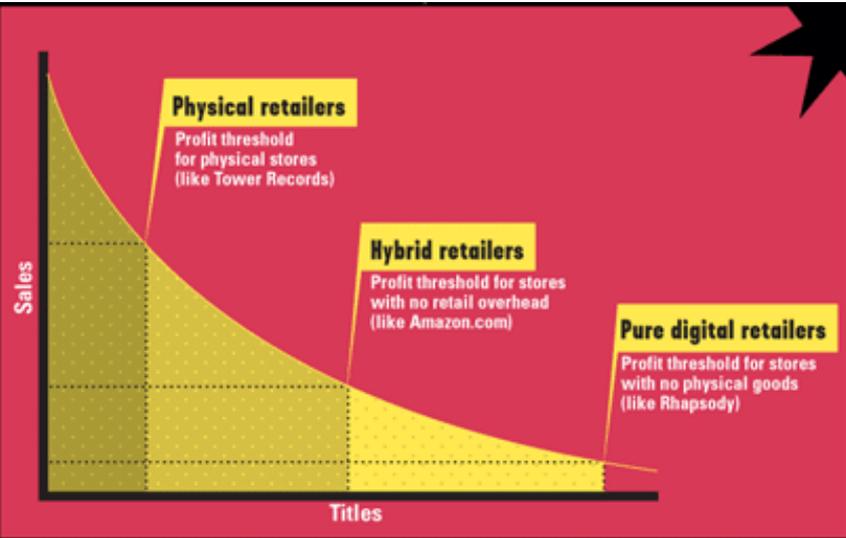
Sources: Erik Brynjolfsson and Jeffrey Hu, MIT, and Michael Smith, Carnegie Mellon; Barnes & Noble; Netflix; RealNetworks

Physical vs. Online

THE BIT PLAYER ADVANTAGE

Beyond bricks and mortar there are two main retail models – one that gets halfway down the Long Tail and another that goes all the way. The first is the familiar hybrid model of Amazon and Netflix, companies that sell physical goods online. Digital catalogs allow them to offer unlimited selection along with search, reviews, and recommendations, while the cost savings of massive warehouses and no walk-in customers greatly expands the number of products they can sell profitably.

Pushing this even further are pure digital services, such as iTunes, which offer the additional savings of delivering their digital goods online at virtually no marginal cost. Since an extra database entry and a few megabytes of storage on a server cost effectively nothing, these retailers have no economic reason not to carry *everything* available.



“IF YOU LIKE BRITNEY, YOU’LL LOVE ...”

Just as lower prices can entice consumers down the Long Tail, recommendation engines drive them to obscure content they might not find otherwise.



Source: Amazon.com

Read <http://www.wired.com/wired/archive/12.10/tail.html> to learn more!

<http://www.mmds.org>

Utility (Preference) Matrix

	Harry Potter 1	Harry Potter 2	Harry Potter 3	Twilight	Star Wars 1	Star Wars 2	Star Wars 3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

How can we fill the empty entries of the matrix?

Recommendation Systems

- **Content-based:**
 - Represent the items into a **feature space** and recommend items to customer C **similar** to previous items rated highly by C
 - Movie recommendations: recommend movies with same actor(s), director, genre, ...
 - Websites, blogs, news: recommend other sites with “similar” content

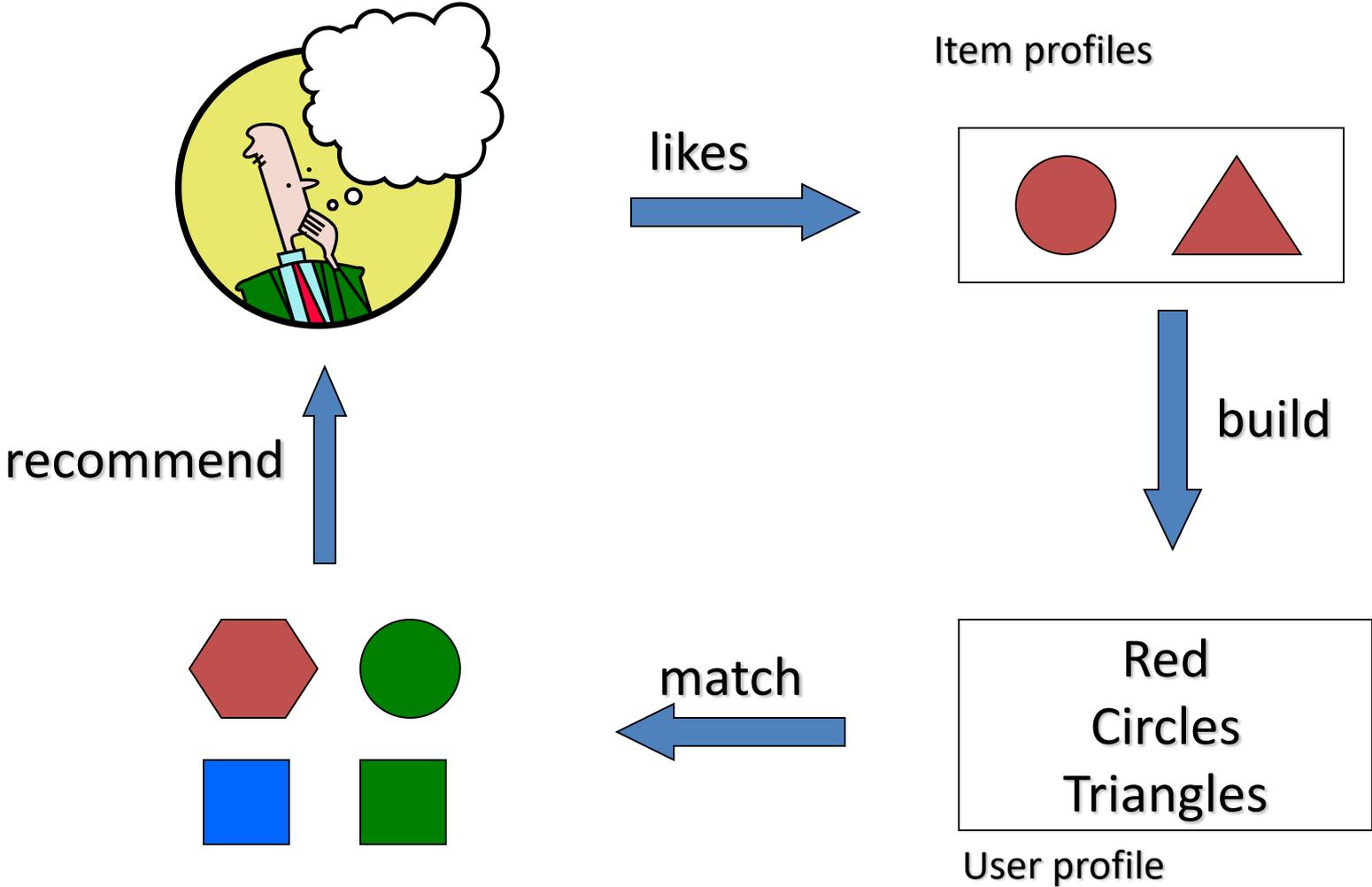
Content-based prediction

	Harry Potter 1	Harry Potter 2	Harry Potter 3	Twilight	Star Wars 1	Star Wars 2	Star Wars 3
A	4	4	4	5	1	1	1
B	5	5	4				
C				2	4	5	5
D		3					3

Someone who likes one of the Harry Potter (or Star Wars) movies is likely to like the rest

- Same actors, similar story, same genre

Intuition



Extracting features for Items

- Map items into a **feature space**:
 - For movies:
 - Actors, directors, genre, rating, year,...
 - For documents?
- Items are now **real vectors** in a multidimensional feature space

	Year	Action	Sci-Fi	Romance	Lucas	H. Ford	Pacino
Star Wars	1977	1	1	0	1	1	0

- Challenge: Make all feature values compatible
- Alternatively we can view a movie as a **set of features**:
 - Star Wars = {1977, Action, Sci-Fi, Lucas, H.Ford}

Extracting Features for Users

- To compare items with users we need to **map** users to the same feature space. How?
 - Take all the movies that the user has seen and take **the average vector**
 - Other aggregation functions are also possible.
- Recommend to user C the **most similar** item i computing similarity in the common feature space
 - How do we measure similarity?

Similarity

- Typically similarity between **vectors** is measured by the **Cosine Similarity**

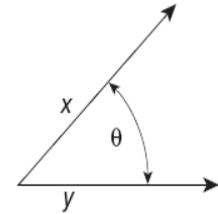
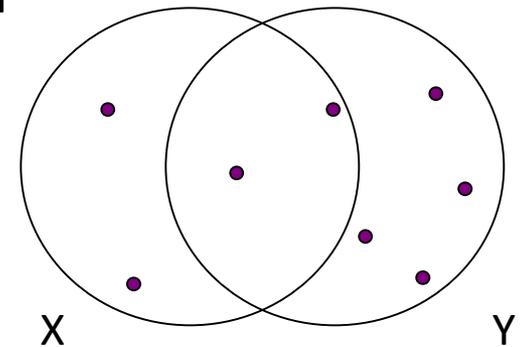


Figure 2.16. Geometric illustration of the cosine measure.

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} = \frac{\sum_{i=1}^d x_i y_i}{\sqrt{\sum_{i=1}^d x_i^2} \sqrt{\sum_{i=1}^d y_i^2}}$$

- If we view the items as **sets** then we can use the **Jaccard Similarity**

$$\text{JSim}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$



Classification approach

- Using the user and item features we can construct a classifier that tries to predict if a user will like a new item

Limitations of content-based approach

- Finding the appropriate features
 - e.g., images, movies, music
- Overspecialization
 - Never recommends items outside user's content profile
 - People might have multiple interests

Collaborative filtering

	Harry Potter 1	Harry Potter 2	Harry Potter 3	Twilight	Star Wars 1	Star Wars 2	Star Wars 3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Two users are similar if they rate the **same items** in a **similar way**

Recommend to user C, the items liked by **many** of the **most similar users**.

User Similarity

	Harry Potter 1	Harry Potter 2	Harry Potter 3	Twilight	Star Wars 1	Star Wars 2	Star Wars 3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Which pair of users do you consider as the most similar?

What is the right definition of similarity?

User Similarity

	Harry Potter 1	Harry Potter 2	Harry Potter 3	Twilight	Star Wars 1	Star Wars 2	Star Wars 3
A	1			1	1		
B	1	1	1				
C				1	1	1	
D		1					1

Jaccard Similarity: users are sets of movies

Disregards the ratings.

$$J_{\text{sim}}(A,B) = 1/5$$

$$J_{\text{sim}}(A,C) = 1/2$$

$$J_{\text{sim}}(B,D) = 1/4$$

User Similarity

	Harry Potter 1	Harry Potter 2	Harry Potter 3	Twilight	Star Wars 1	Star Wars 2	Star Wars 3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Cosine Similarity:

Assumes zero entries are negatives:

$$\text{Cos}(A,B) = 0.38$$

$$\text{Cos}(A,C) = 0.32$$

User Similarity

	Harry Potter 1	Harry Potter 2	Harry Potter 3	Twilight	Star Wars 1	Star Wars 2	Star Wars 3
A	2/3			5/3	-7/3		
B	1/3	1/3	-2/3				
C				-5/3	1/3	4/3	
D		0					0

Normalized Cosine Similarity:

- Subtract the mean rating per user and then compute Cosine (**correlation coefficient**)

$$\text{Corr}(A,B) = 0.092$$

$$\text{Corr}(A,C) = -0.559$$

User-User Collaborative Filtering

- For a user u , find the set $TopK(u)$ of the K users whose ratings are most “similar” to u ’s ratings
- Estimate u ’s ratings based on ratings of users in $TopK$ using some aggregation function. For item i :

$$\widehat{r}_{ui} = \frac{1}{Z} \sum_{v \in TopK(u)} \text{sim}(u, v) r_{vi}$$

$$Z = \sum_{v \in TopK(u)} \text{sim}(u, v)$$

- Modeling deviations:

$$\widehat{r}_{ui} = \overline{r}_u + \frac{1}{Z} \sum_{v \in TopK(u)} \text{sim}(u, v) (\overline{r}_v - r_{vi})$$

Mean rating of u Deviation from mean for v Mean deviation of similar users

- Advantage: for each user we have small amount of computation.

Item-Item Collaborative Filtering

- We can **transpose (flip)** the matrix and perform the same computation as before to define similarity between items
 - Intuition: Two items are similar if they are **rated in the same way by many users**.
 - Better defined similarity since it captures the notion of **genre** of an item
 - Users may have multiple interests.
- Algorithm: For each user u and item i
 - Find the set $TopK_u(i)$ of **most similar items** to item i that have been rated by user u .
 - **Aggregate** their ratings to predict the rating for item i .
- Disadvantage: we need to consider each user-item pair separately

Evaluation

- Split the data into **train** and **test** set
 - Keep a fraction of the ratings to test the accuracy of the predictions
- Metrics:
 - **Root Mean Square Error** (RMSE) for measuring the quality of **predicted ratings**:

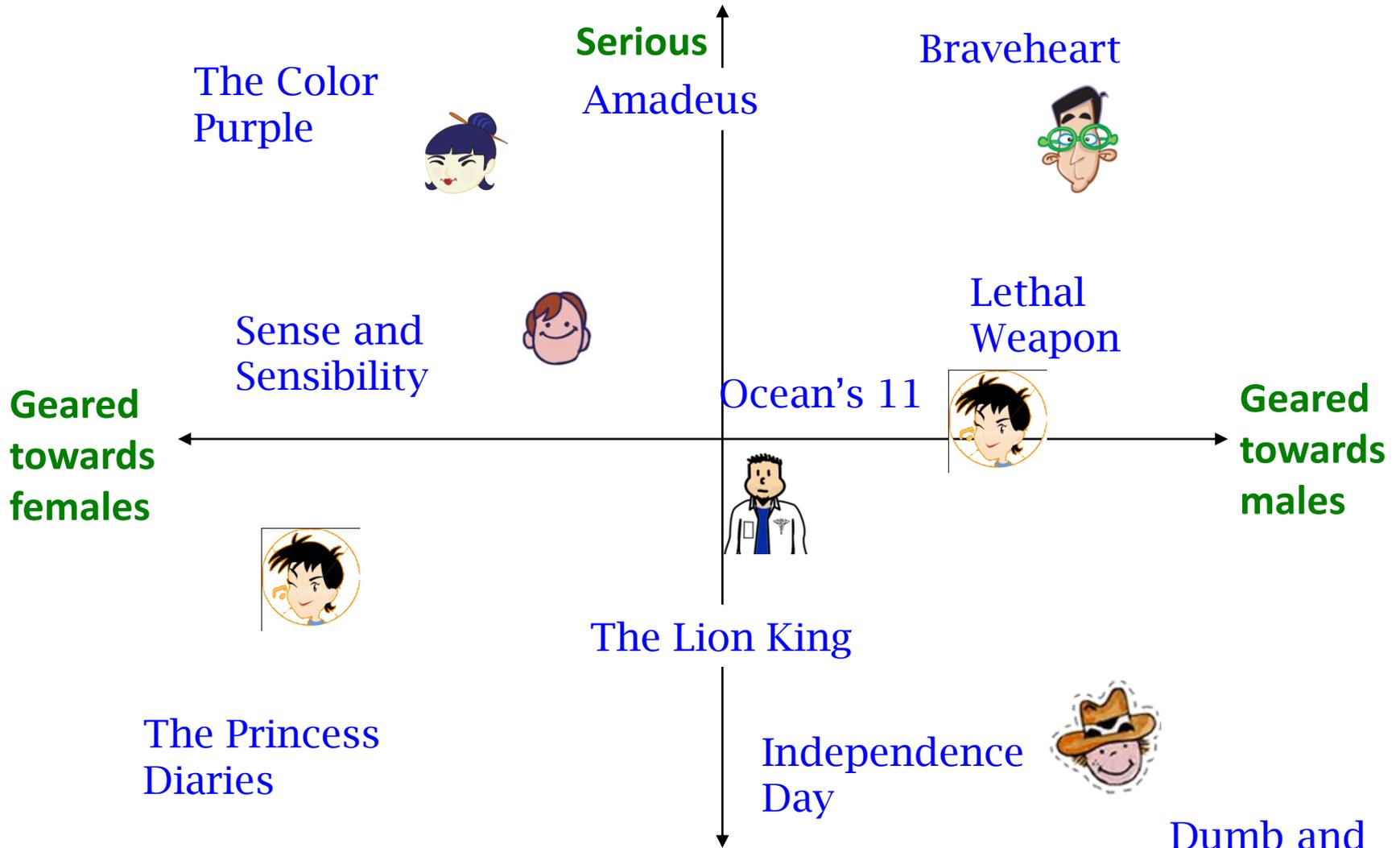
$$RMSE = \frac{1}{n} \sqrt{\sum_{i,j} (\widehat{r}_{ij} - r_{ij})^2}$$

- **Precision/Recall** for measuring the quality of **binary (action/no action) predictions**:
 - Precision = fraction of predicted actions that were correct
 - Recall = fraction of actions that were predicted correctly
- **Kendal' tau** for measuring the quality of predicting the **ranking of items**:
 - The fraction of pairs of items that are ordered correctly
 - The fraction of pairs that are ordered incorrectly

Model-Based Collaborative Filtering

- So far we have looked at specific user-item combinations
- A different approach looks at the full user-item matrix and tries to find a **model** that explains how the ratings of the matrix are **generated**
 - If we have such a model, we can **predict** the ratings that we do not observe.
- **Latent factor model:**
 - (Most) models assume that there are **few (K) latent factors** that define the behavior of the users and the characteristics of the items

Latent Factor Models

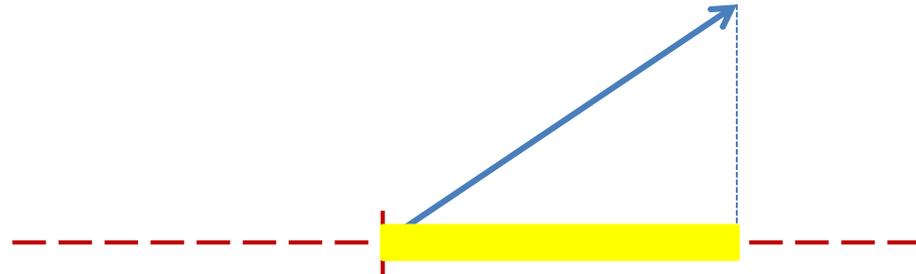


Linear algebra

- 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
 - If both u and v are unit vectors dot product is the **cosine similarity** between u and v .

- $[-1, 2, 3] \begin{bmatrix} 4 \\ -1 \\ 2 \end{bmatrix} = 0$: **orthogonal** vectors

- **Orthonormal** vectors: two unit vectors that are orthogonal

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
- **Example**
 - Matrix $\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$ has rank $\mathbf{r=2}$
 - **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 $c = [1, 2, 3]^T$
- **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**

$$- A = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} [1, 2, -1] = cr^T$$

Singular Value Decomposition

- The **left singular vectors** are an **orthonormal basis** for the **column space** of A .
- The **right singular vectors** are an **orthonormal basis** for the **row 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 \mathbf{v}
 - Strength of the i -th linear trend: $\|A\mathbf{v}_i\| = \sigma_i$

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, \dots, u_r : **Eigenvectors** of A

An (extreme) example

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

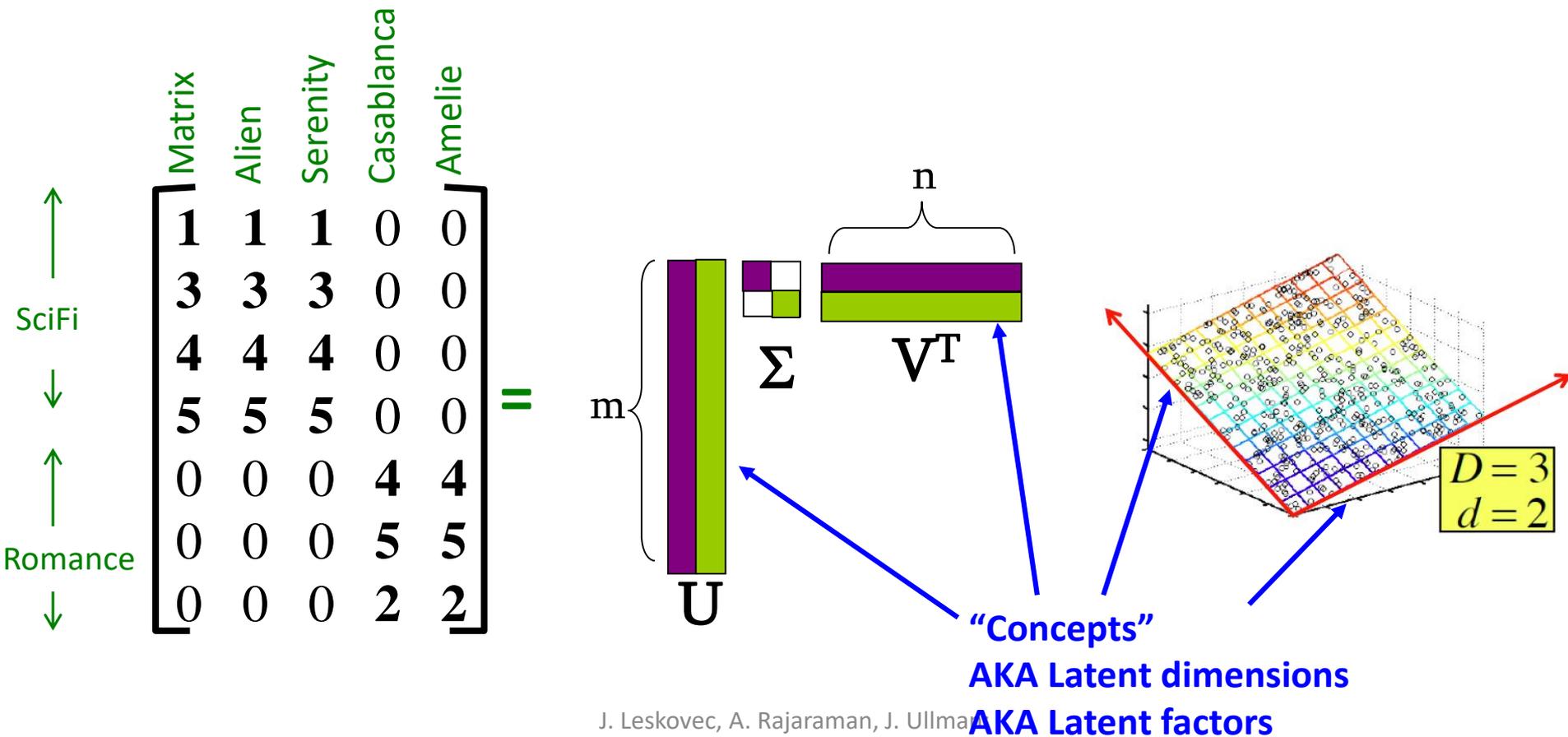
$$A = \begin{array}{|c|c|} \hline \text{Blue} & \text{White} \\ \hline \text{White} & \text{Red} \\ \hline \end{array}$$

- 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

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



SVD – Example: Users-to-Movies

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

$$\begin{matrix}
 \uparrow \\
 \text{SciFi} \\
 \downarrow \\
 \uparrow \\
 \text{Romance} \\
 \downarrow
 \end{matrix}
 \begin{bmatrix}
 \text{Matrix} \\
 \text{Alien} \\
 \text{Serenity} \\
 \text{Casablanca} \\
 \text{Amelie}
 \end{bmatrix}
 \begin{bmatrix}
 1 & 1 & 1 & 0 & 0 \\
 3 & 3 & 3 & 0 & 0 \\
 4 & 4 & 4 & 0 & 0 \\
 5 & 5 & 5 & 0 & 0 \\
 0 & 0 & 0 & 4 & 4 \\
 0 & 0 & 0 & 5 & 5 \\
 0 & 0 & 0 & 2 & 2
 \end{bmatrix}
 =
 \begin{bmatrix}
 0.14 & 0.00 \\
 0.42 & 0.00 \\
 0.56 & 0.00 \\
 0.70 & 0.00 \\
 0.00 & 0.60 \\
 0.00 & 0.75 \\
 0.00 & 0.30
 \end{bmatrix}
 \begin{matrix}
 \text{SciFi-concept} \\
 \text{Romance-concept}
 \end{matrix}
 \times
 \begin{bmatrix}
 12.4 & 0 \\
 0 & 9.5
 \end{bmatrix}
 \times
 \begin{bmatrix}
 0.58 & 0.58 & 0.58 & 0.00 & 0.00 \\
 0.00 & 0.00 & 0.00 & 0.71 & 0.71
 \end{bmatrix}$$

SVD – Example: Users-to-Movies

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

SciFi-concept ↓
Romance-concept ↙

$$= \begin{bmatrix} 0.14 & 0.00 \\ 0.42 & 0.00 \\ 0.56 & 0.00 \\ 0.70 & 0.00 \\ 0.00 & 0.60 \\ 0.00 & 0.75 \\ 0.00 & 0.30 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.71 & 0.71 \end{bmatrix}$$

U is “user-to-concept” similarity matrix

SVD – Example: Users-to-Movies

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

Matrix Alien Serenity Casablanca Amelie

SciFi

Romance

SciFi-concept

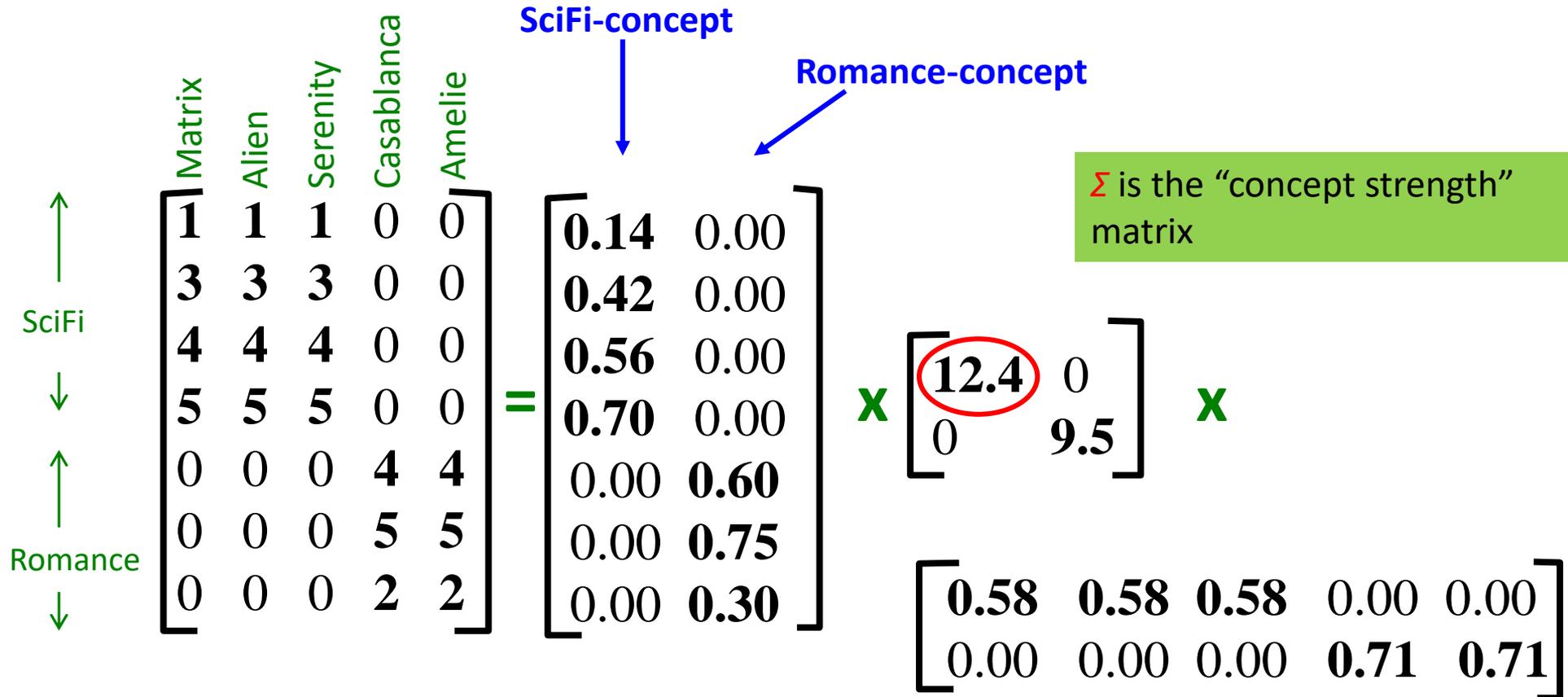
Romance-concept

V is "movie to concept" similarity matrix

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.14 & 0.00 \\ 0.42 & 0.00 \\ 0.56 & 0.00 \\ 0.70 & 0.00 \\ 0.00 & 0.60 \\ 0.00 & 0.75 \\ 0.00 & 0.30 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.71 & 0.71 \end{bmatrix}$$

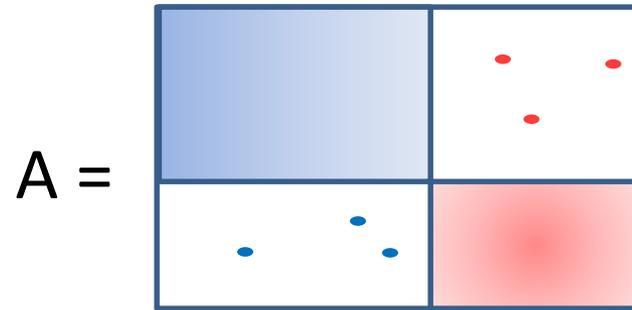
SVD – Example: Users-to-Movies

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



An (more realistic) example

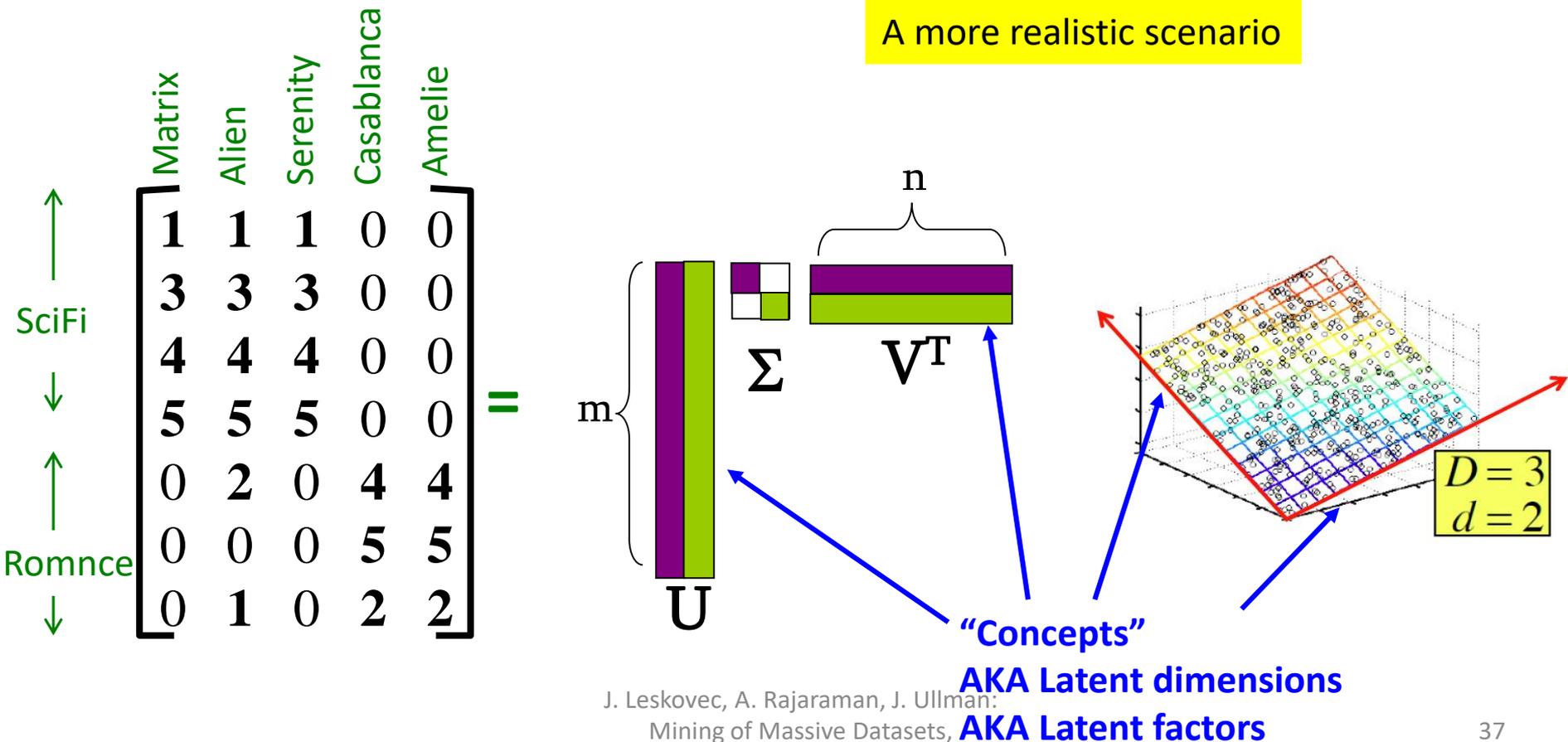
- User-Movie matrix



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

SVD – Example: Users-to-Movies

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



J. Leskovec, A. Rajaraman, J. Ullman:

Mining of Massive Datasets, AKA Latent factors

<http://www.mmms.org>

SVD – Example: Users-to-Movies

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

	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	2	0	4	4				
	0	0	0	5	5				
	0	1	0	2	2				

SciFi

Romance

SciFi-concept

Romance-concept

The first two vectors are more or less unchanged

J. Leskovec, A. Rajaraman, J. Ullman:
Mining of Massive Datasets,
<http://www.mmms.org>

SVD - Interpretation

‘**movies**’, ‘**users**’ and ‘**concepts**’:

- U : user-to-concept similarity matrix
- V : movie-to-concept similarity matrix
- Σ : its diagonal elements:
 ‘**strength**’ of each concept

Rank-k approximation

- In the last User-Movie matrix 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
 - This is **the best rank-2** approximation of A
 - The best two latent factors

SVD as an optimization

- The rank-k approximation matrix A_k produced by the top-k singular vectors of A minimizes the sum of square errors for the entries of matrix A

$$A_k = \arg \max_{B: \text{rank}(B)=k} \sum_{i,j} (A_{ij} - B_{ij})^2$$

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

Example

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

Example

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

Example

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

Frobenius norm:

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

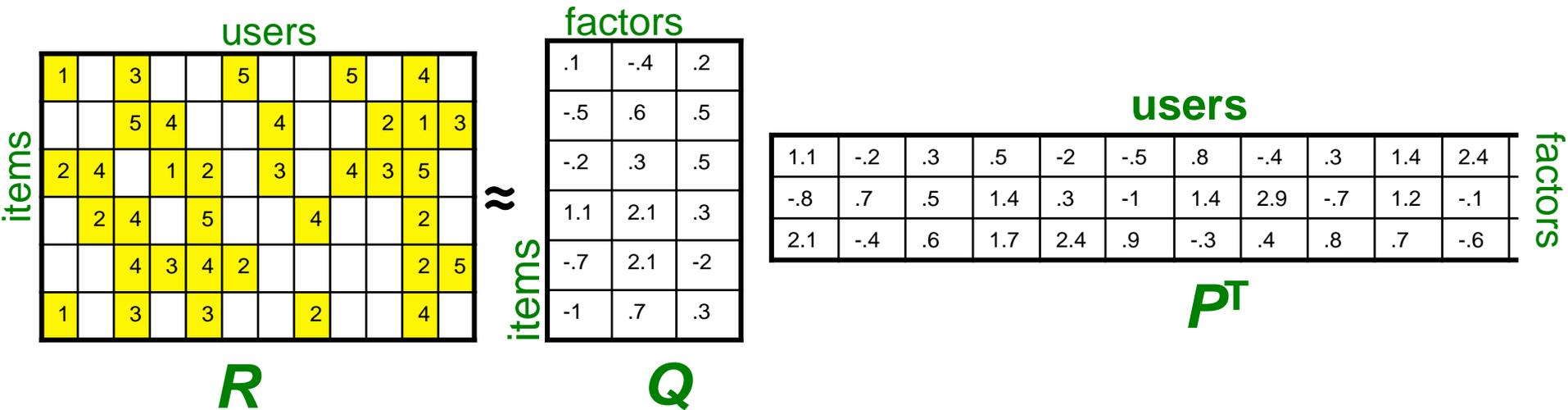
$$\|A-B\|_F = \sqrt{\sum_{ij} (A_{ij}-B_{ij})^2}$$

is "small"

Latent Factor Models

- “SVD” on Netflix data: $R \approx Q \cdot P^T$

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



- For now let's assume we can approximate the rating matrix R as a product of “thin” $Q \cdot P^T$

– R has missing entries but let's ignore that for now!

- Basically, we will want the reconstruction error to be small on known ratings and we don't care about the values on the missing ones

Ratings as Products of Factors

- How to estimate the missing rating of user x for item i ?

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

≈

$$\hat{r}_{xi} = \mathbf{q}_i^T \cdot \mathbf{p}_x$$

$$= \sum_f q_{if} \cdot p_{xf}$$

\mathbf{q}_i = row i of \mathbf{Q}
 \mathbf{p}_x = column x of \mathbf{P}^T

items

factors

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

factors

users

\mathbf{P}^T

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



Ratings as Products of Factors

- How to estimate the missing rating of user x for item i ?

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

≈

$$\hat{r}_{xi} = \mathbf{q}_i^T \cdot \mathbf{p}_x$$

$$= \sum_f \mathbf{q}_{if} \cdot \mathbf{p}_{xf}$$

\mathbf{q}_i = row i of \mathbf{Q}
 \mathbf{p}_x = column x of \mathbf{P}^T

items

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

factors

factors

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

\mathbf{P}^T



Ratings as Products of Factors

- How to estimate the missing rating of user x for item i ?

users

	1		3		5		5		4		
items			5	4	2.4	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

≈

$$\hat{r}_{xi} = q_i^T \cdot p_x$$

$$= \sum_f q_{if} \cdot p_{xf}$$

q_i = row i of Q
 p_x = column x of P^T

items

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

f factors

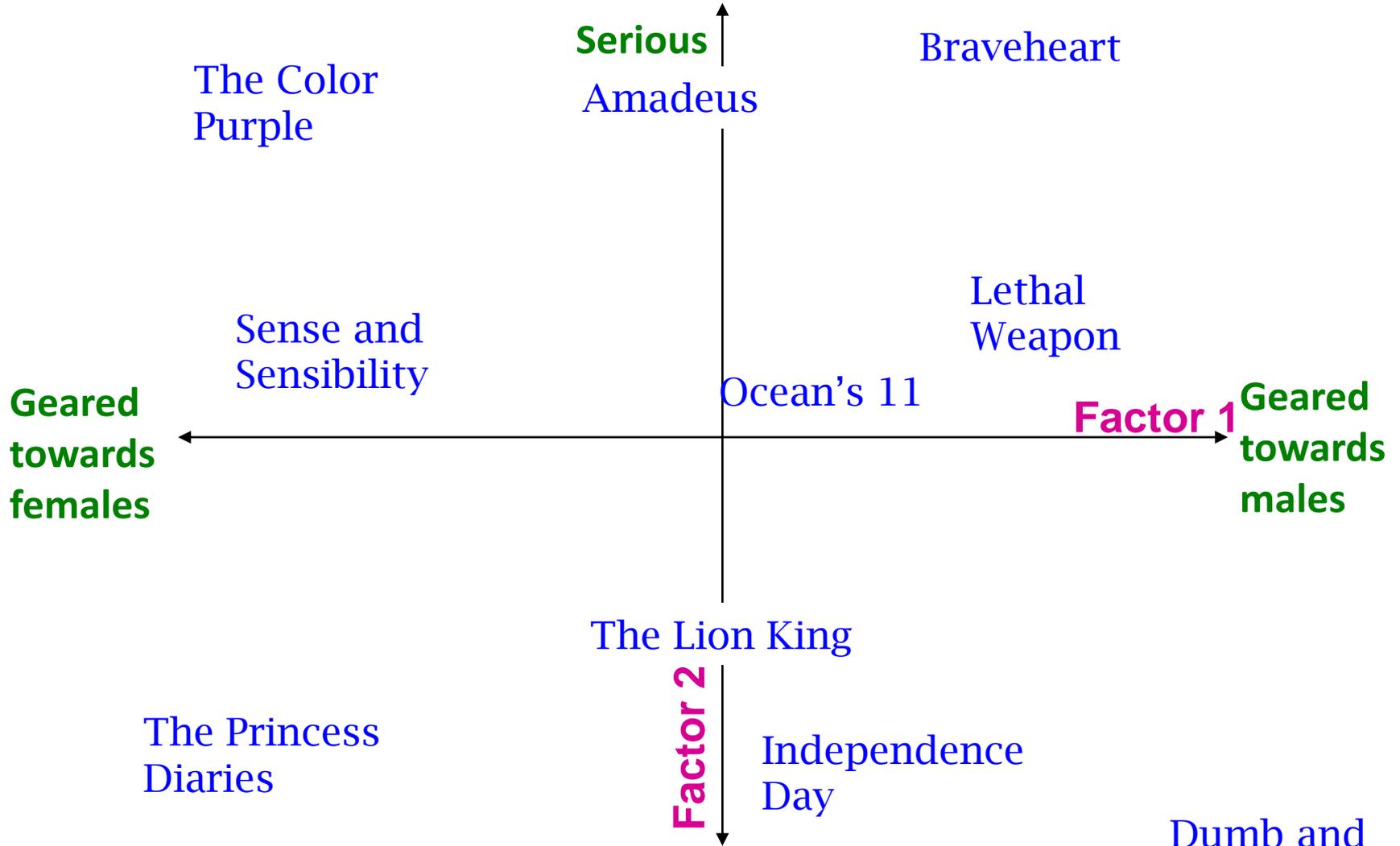
users

f factors

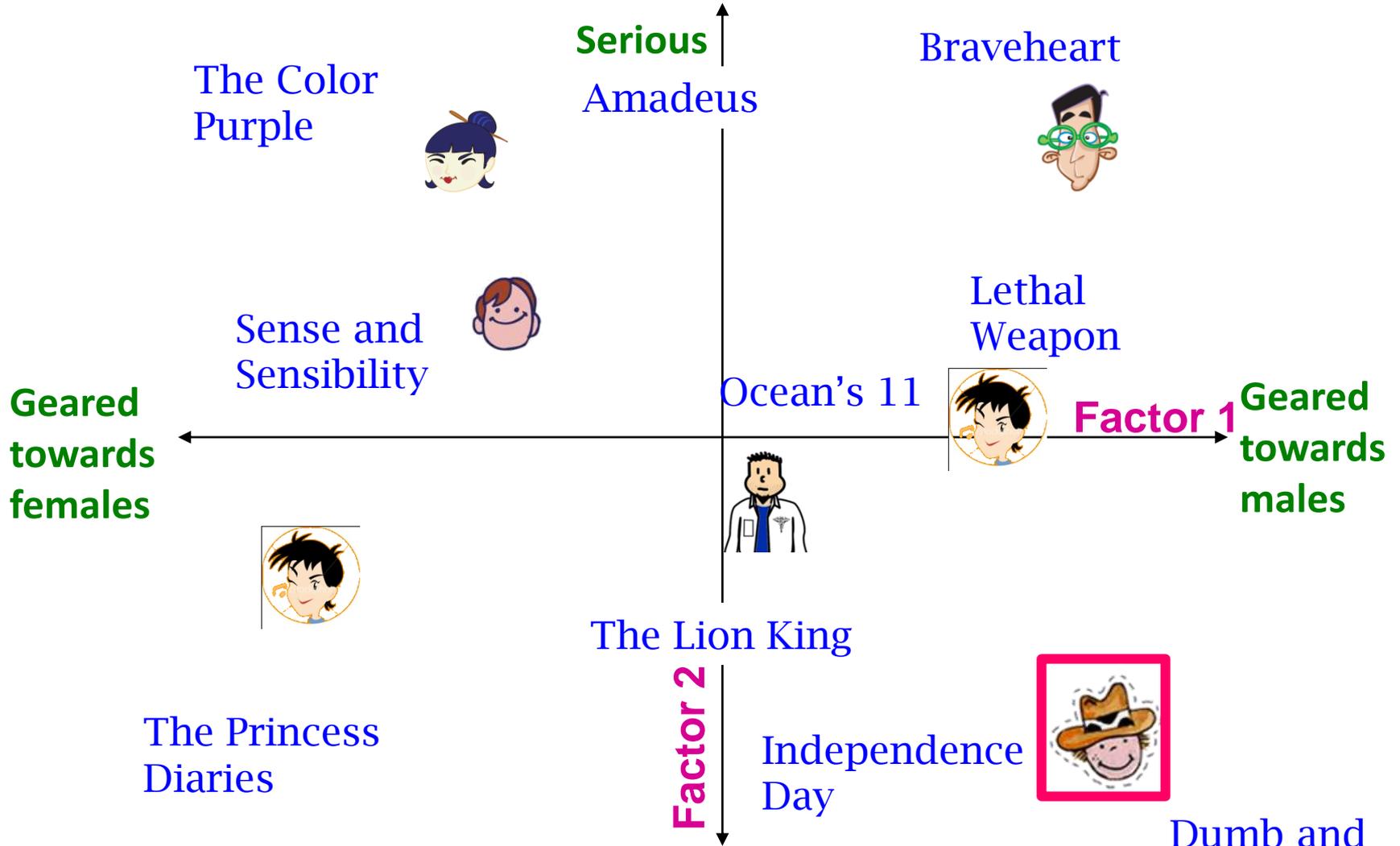
1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

P^T

Latent Factor Models



Latent Factor Models



Example

Missing ratings

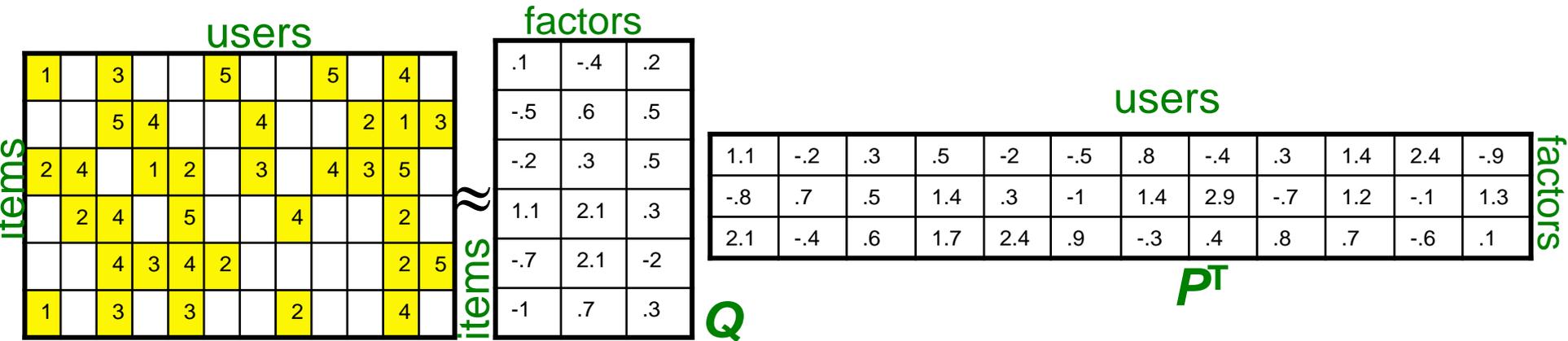
$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 3 & 3 & 0 & 0 \\ 4 & 4 & 0 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.14 & -0.06 & -0.04 \\ 0.30 & -0.11 & -0.61 \\ 0.43 & -0.16 & 0.76 \\ 0.74 & -0.31 & -0.18 \\ 0.15 & 0.53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

Example

- Reconstruction of missing ratings

0.96	1.14	0.82	-0.01	-0.01
1.94	2.32	1.66	0.07	0.07
2.77	3.32	2.37	0.08	0.08
4.84	5.74	4.14	-0.08	0.08
0.40	1.42	0.33	4.06	4.06
-0.42	0.63	-0.38	4.92	4.92
0.20	0.71	0.16	2.03	2.03

Latent Factor Models



- SVD also considers entries that are missing!
- Use specialized methods to find P, Q

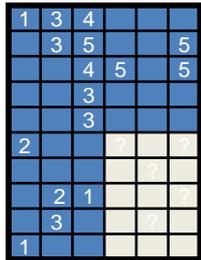
$$- \min_{P, Q} \sum_{(i, x) \in R} (r_{xi} - q_i^T \cdot p_x)^2 \quad \hat{r}_{xi} = q_i^T \cdot p_x$$

– **Note:**

- We don't require cols of P, Q to be orthogonal/unit length
- P, Q map users/movies to a latent space

Back to Our Problem

- **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:**

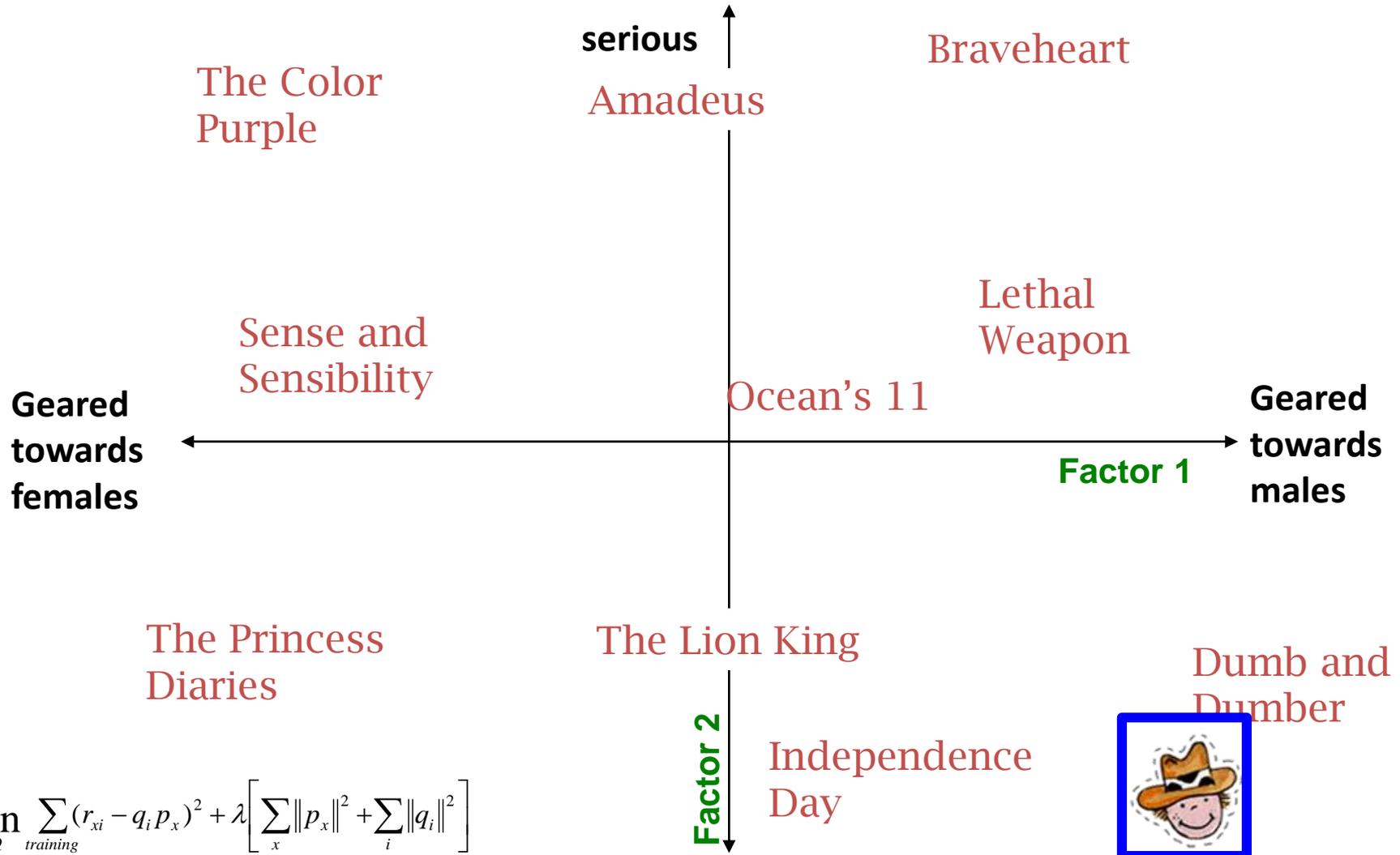
- Allow rich model where there are sufficient data
- Shrink aggressively where data are scarce

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

$\lambda_1, \lambda_2 \dots$ 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

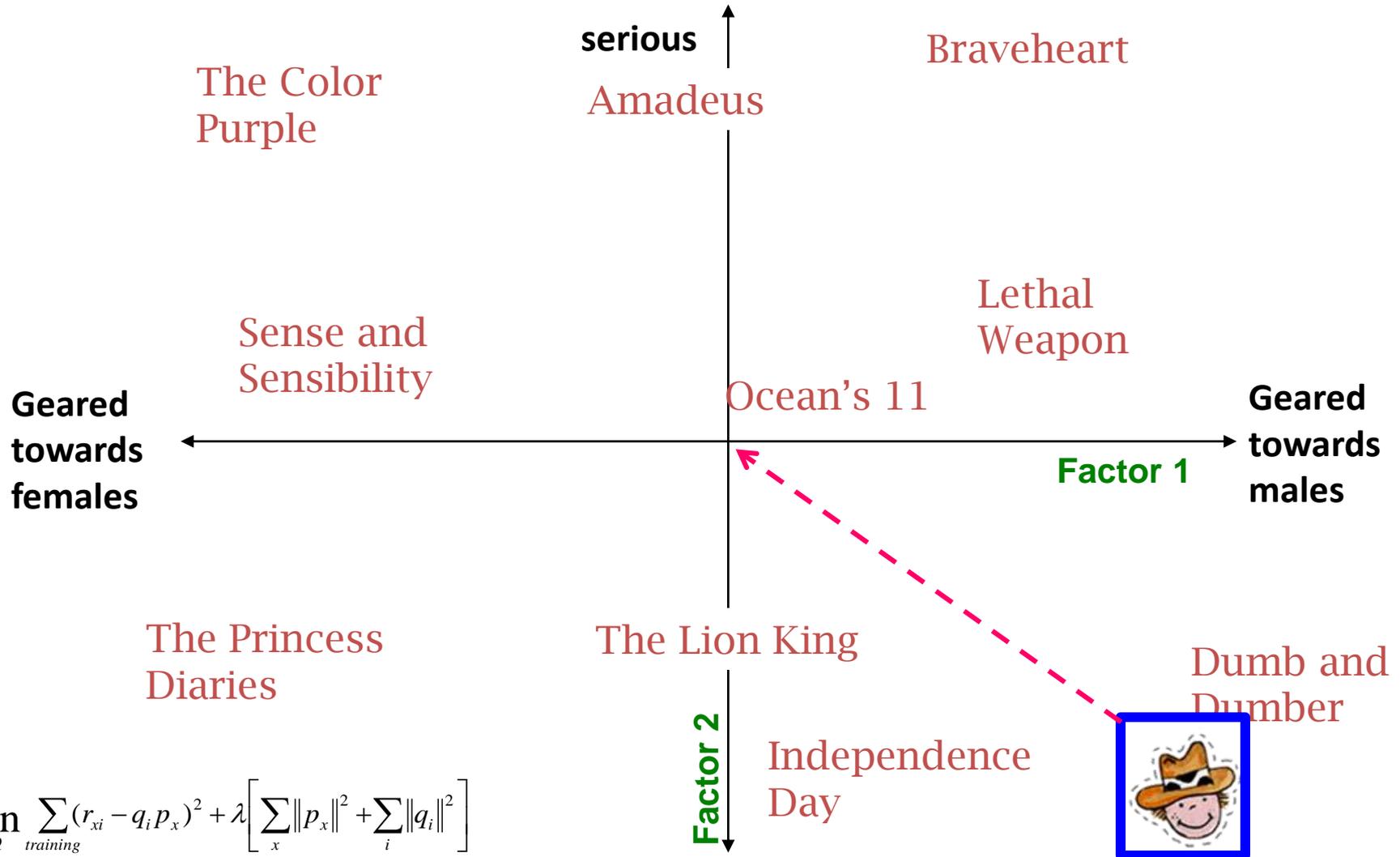
The Effect of Regularization



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

min_{factors} "error" + λ "length"

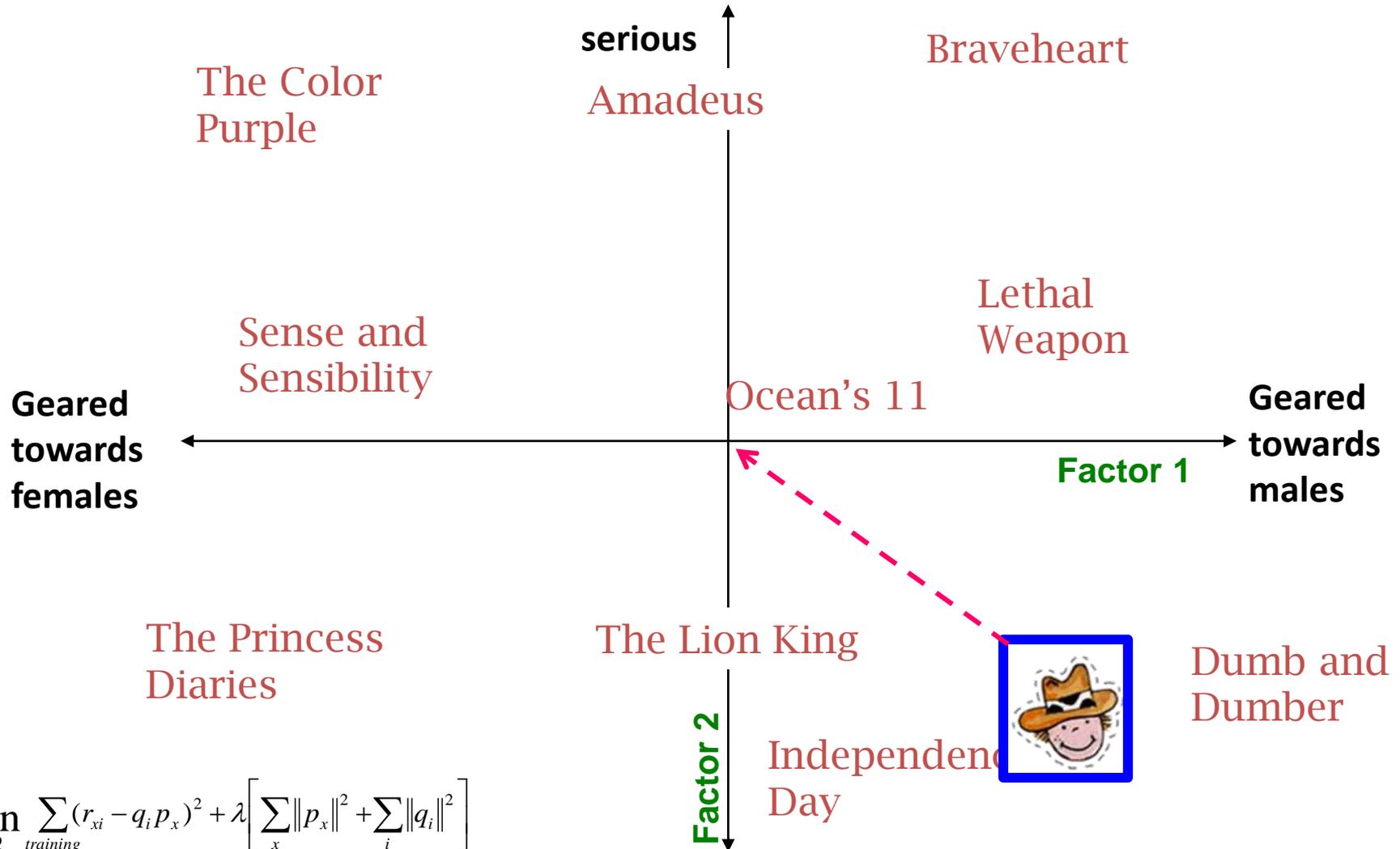
The Effect of Regularization



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

min_{factors} "error" + λ "length"

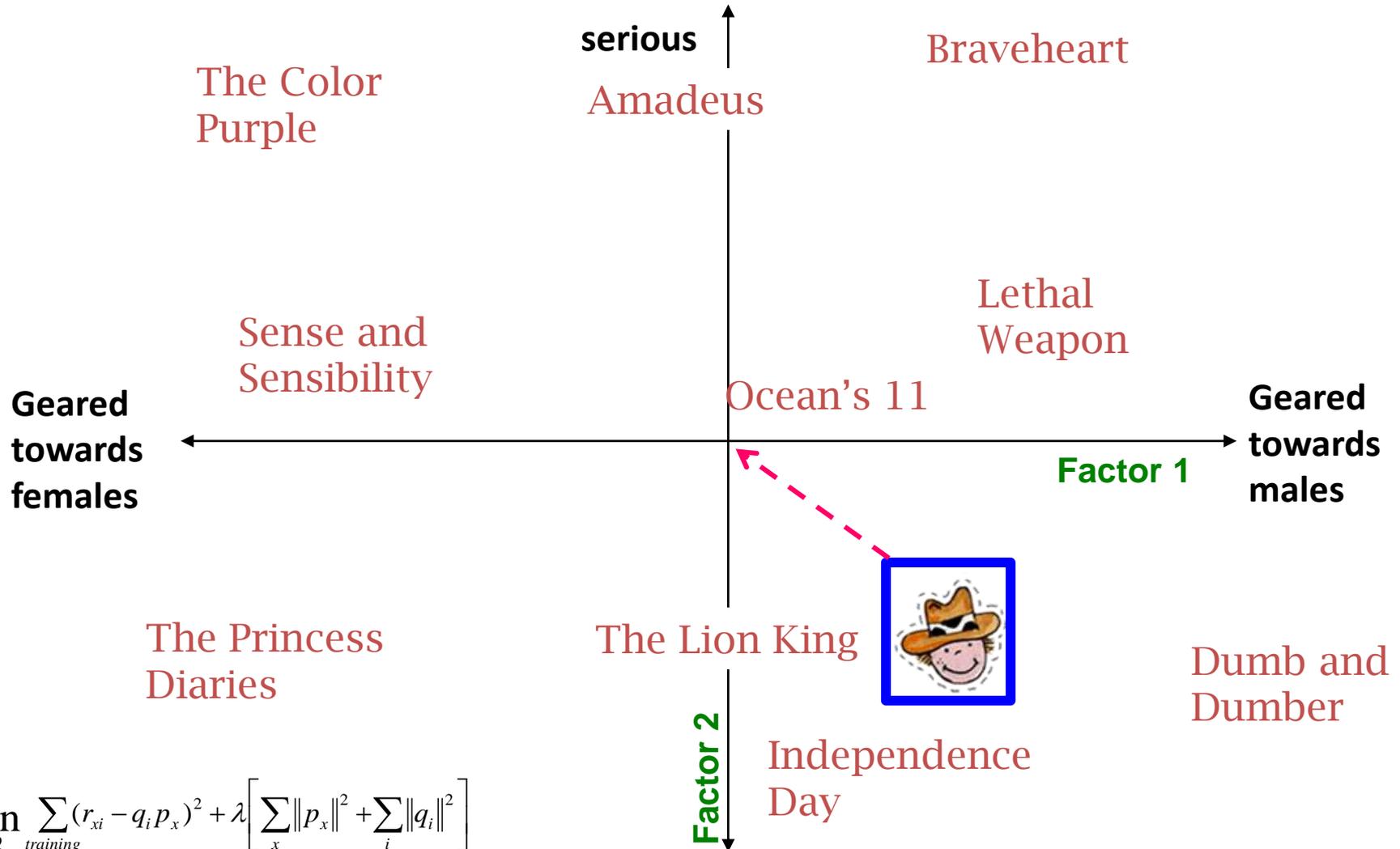
The Effect of Regularization



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

min_{factors} "error" + λ "length"

The Effect of Regularization



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

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

Pros and cons of collaborative filtering

- Works for any kind of item
 - No feature selection needed
- Cold-Start problem:
 - New user problem
 - New item problem
- Sparsity of rating matrix
 - Cluster-based smoothing?

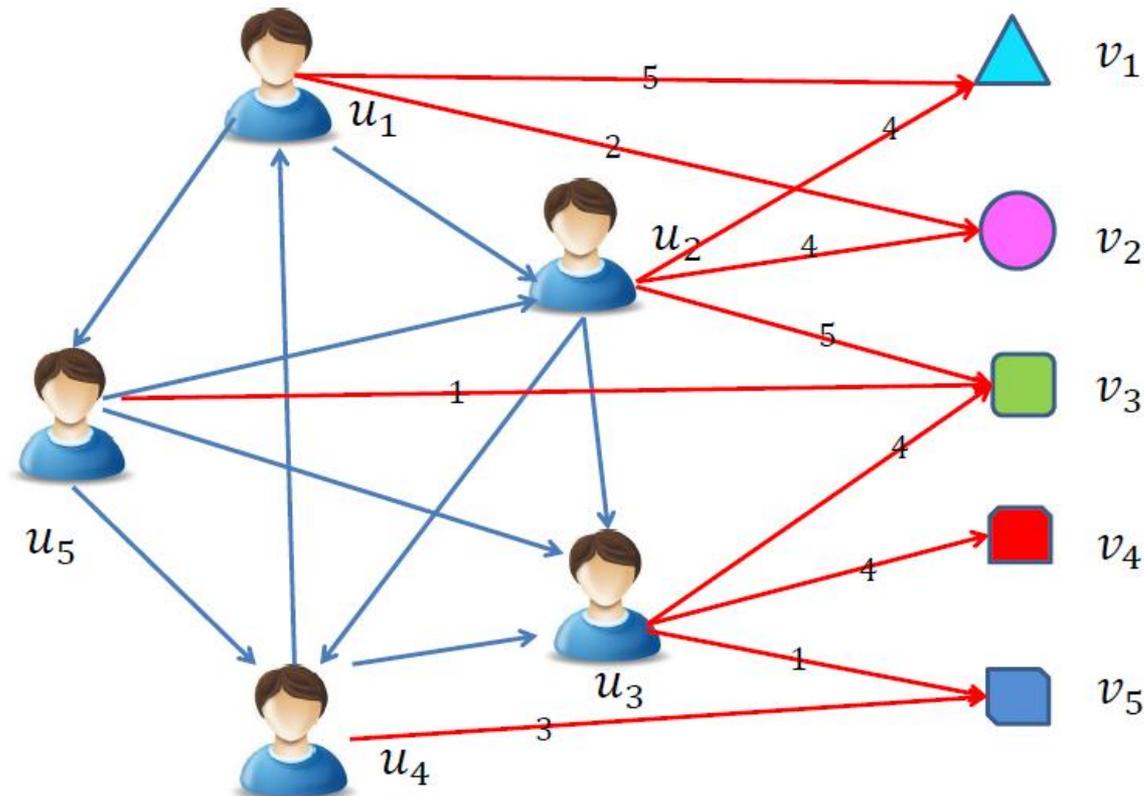
The Netflix Challenge

- 1M prize to improve the prediction accuracy by 10%



Social Recommendations

- Suppose that except for the rating matrix, you are also given a **social network** of all the users
 - Such social networks exist in sites like Yelp, Foursquare, Epinions, etc



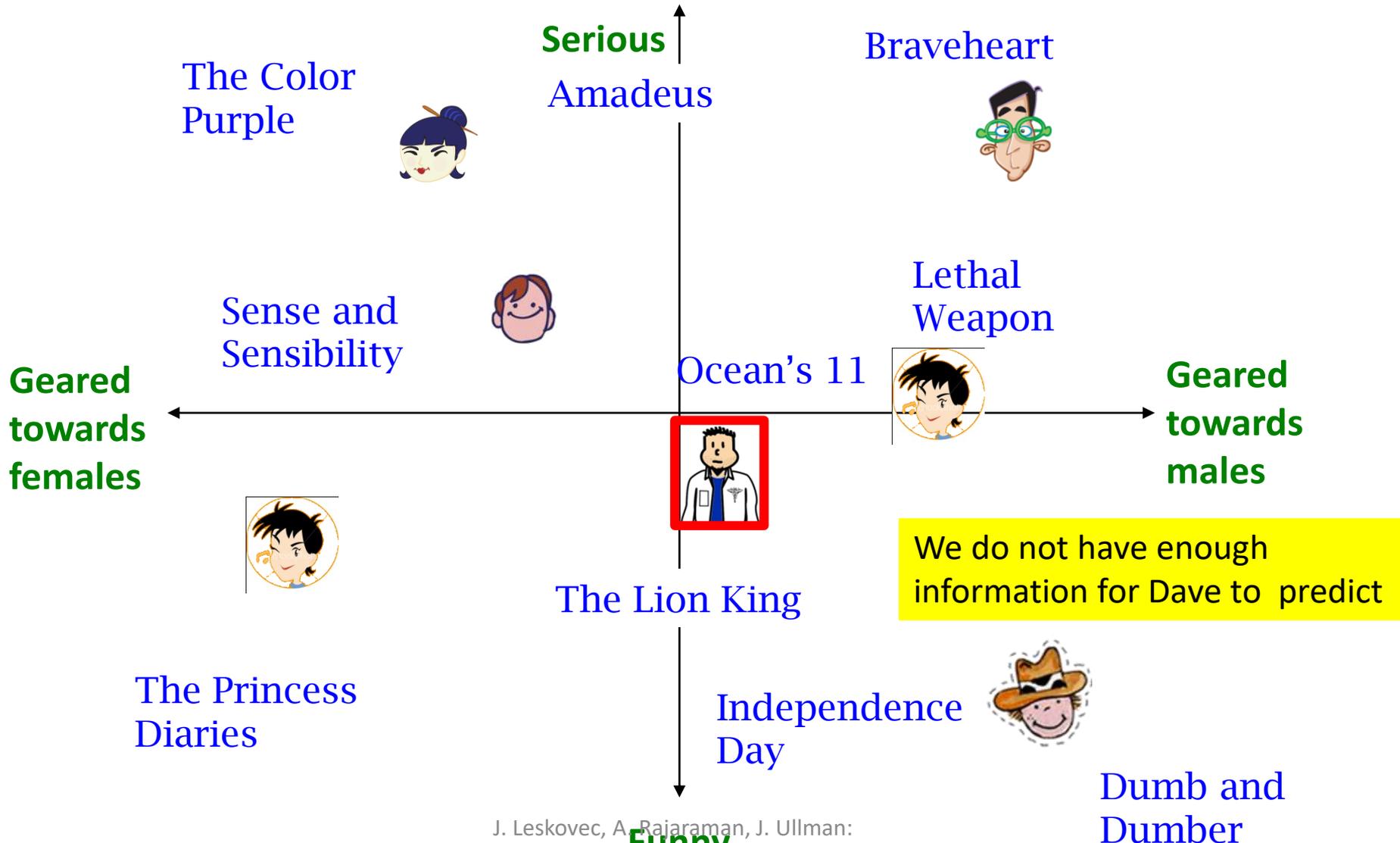
Social Recommendations

- How can we use the social network to improve recommendations?
 - **Homophily**: connected users are likely to have similar ratings.

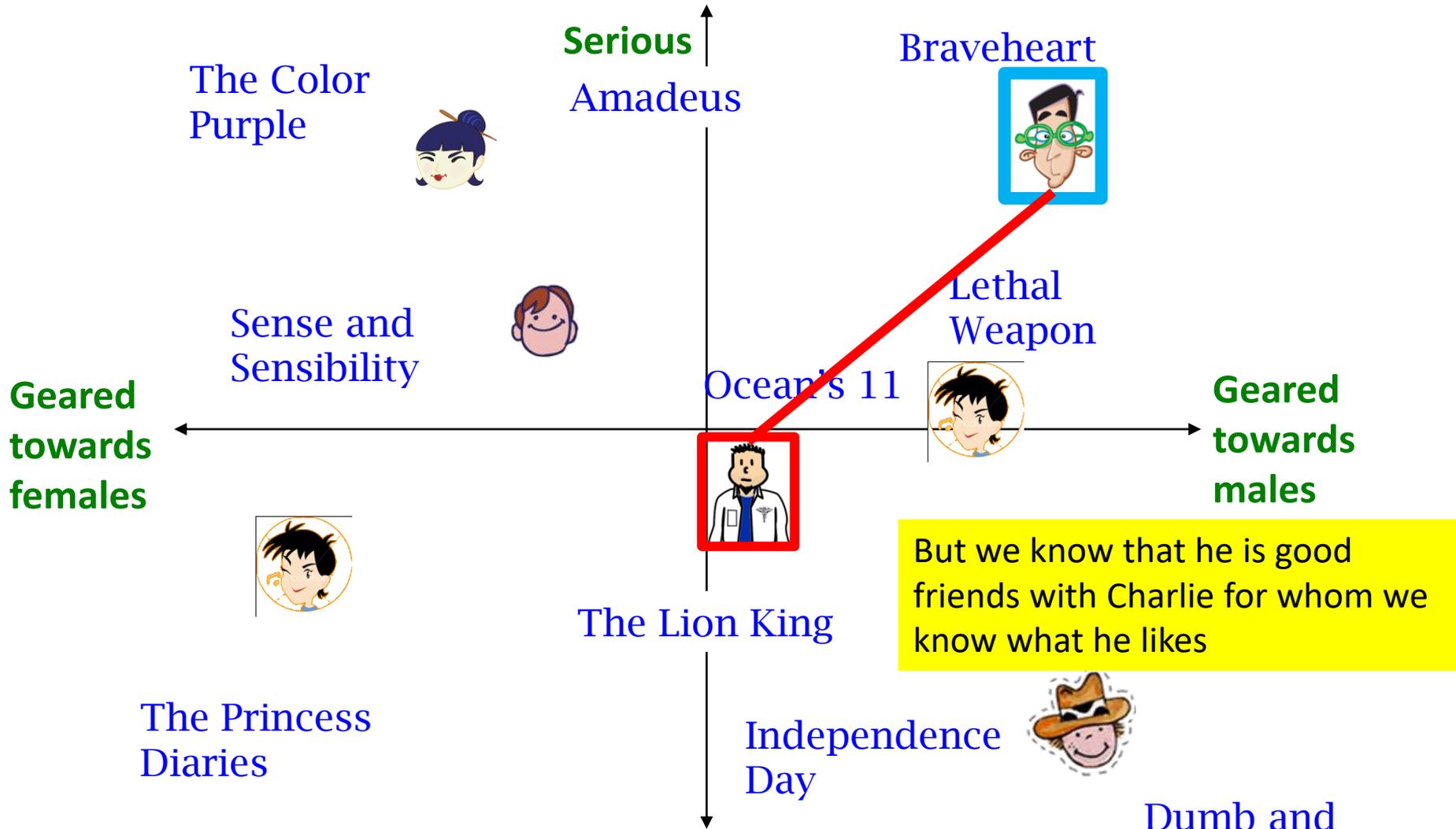
Social Recommendations

- Ideas?
 - For a user who has no ratings, but has friends take the average of the friends to be the predicted rating
 - Use relationship strength when available
 - Combine the rating similarity and neighborhood rating to predict rating.
 - Use social network to determine similarity between users.

Social Recommendations



Social Recommendations



But we know that he is good friends with Charlie for whom we know what he likes

Social Recommendations



Social Regularization

- Mathematically, this means that we add an additional **regularization** term to our optimization function:

$$\min_{P,Q} \left\{ \begin{array}{l} \sum_{r_{ix}} (r_{ix} - q_i^T p_x)^2 \\ + \lambda \left[\sum_i \|q_i\|^2 + \sum_x \|p_x\|^2 \right] \\ + \beta \sum_{x,y} w_{xy} \|p_x - p_y\|^2 \end{array} \right\}$$

- w_{xy} : strength of the connection between x and y
- $\|p_x - p_y\|$: the difference between the latent preferences of the users.

Social Regularization

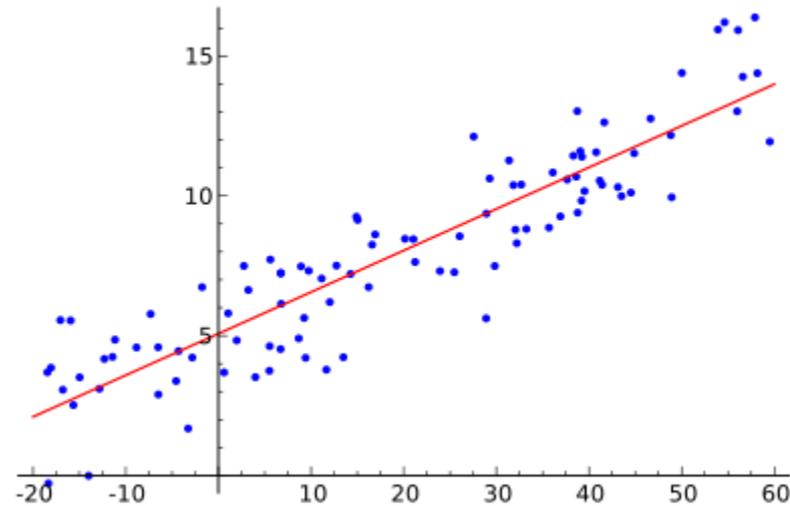
- Helps in giving **additional information** about the preferences of the users
- Helps with **sparse data** since it allows us to make inferences for users for whom we have little data.
- The same idea can be applied in different settings

Example: Linear regression

- Regression: Create a model to predict a **continuous value**.
- Given a dataset of the form $\{(x_1, y_1), \dots, (x_n, y_n)\}$ find a linear function that given the vector x_i predicts the y_i value as $y'_i = w^T x_i$
 - Find a vector of weights w that **minimizes the sum of square errors**

$$\sum_i (w^T x_i - y_i)^2$$

- Several techniques for solving the problem. Closed form solution.



Linear regression task

- Example application: we want to predict the **popularity** of a blogger.
 - We can create **features** about the text of the posts, length, frequency, topics, etc.
 - Using training data we can find the linear function that best predicts the popularity

Linear regression with social regularization

- Suppose now that we have a **social network** between bloggers: there is a link between two bloggers if they follow each other.
 - Assumption: bloggers that are **linked** are likely to be of **similar quality**.
- Minimize:

$$\underbrace{\sum_i (w^T x_i - y_i)^2}_{\text{Regression Cost}} + \alpha \underbrace{\sum_{(i,j) \in E} (w^T x_i - w^T x_j)^2}_{\text{Regularization Cost}}$$

This is sometimes also called **network smoothing**

- This can be written as:

$$\sum_i (w^T x_i - y_i)^2 + \underbrace{\alpha w^T X L_A X^T w}$$

L_A : The Laplacian of the adjacency matrix

- There is still a closed form solution.

Collective Classification

- The same idea can be applied to **classification** problems:
 - Classification: create a model that given a set of features predicts a **discrete class label**
 - E.g. predict what a facebook user will vote in the next election.
 - We can use the postings of the user to train a model that predicts among the existing parties (independent classification)
- We also have the Facebook **social network** information:
 - It is reasonable to assume that people that are connected are more likely to vote the same or similar way
 - We want to **collectively** assign labels so that connected nodes are likely to get similar labels

Collective Classification

- A general formulation:
 - Given a graph $G = (V, E)$ find a **labeling** $f: V \rightarrow L$ of the nodes of a graph such that the following cost is minimized:

$$\underbrace{\sum_{v \in V} \text{cost}(v, f(v))}_{\text{Classification Cost}} + \underbrace{\sum_{(v,u) \in E} \text{dist}(f(v), f(u))}_{\text{Separation Cost}}$$

- This idea has been studied in many different settings and has many different names
 - Ising model
 - Markov Random Fields
 - Metric Labeling
 - Graph Regularization
 - Graph Smoothing

Social network factorization

- Main idea: the **social network adjacency matrix** can also be factorized
 - Usually we assume a **weighted directed** graph (either explicitly (trust), or implicitly (normalized degree))
 - Intuition: user connections are due to underlying latent factors that cause them to connect.

$$A = YZ^T$$

where $Y, Z \in R^{N \times K}$

Y : the latent factors that describe how a user trusts other users

Z : the latent factors that describe how a user is trusted by other users

Co-factorization

- Force the latent factors that describe user purchase behavior to be the same as those that describe trust behavior

$$- Y = P$$

$$\min_{P,Q} \left\{ \begin{array}{l} \sum_{rix} (r_{ix} - q_i^T p_x)^2 + \sum_{x,y} (A_{x,y} - p_x z_y^T) \\ + \lambda \left[\sum_i \|q_i\|^2 + \sum_x \|p_x\|^2 + \sum_x \|z_x\|^2 \right] \end{array} \right\}$$