Online Social Networks and Media

Team formation in Social Networks

Recommender Systems and Social Recommendations

Thanks to:
Evimaria Terzi
Jure Leskovec, Anand Rajaraman, Jeff Ullman

Thanks to Evimari Terzi

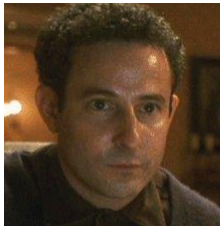
ALGORITHMS FOR TEAM FORMATION

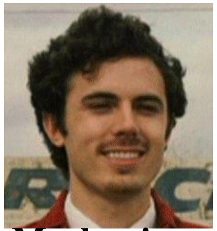
Team-formation problems

- Given a task and a set of experts (organized in a network) find the subset of experts that can effectively perform the task
- Task: set of required skills and potentially a budget
- Expert: has a set of skills and potentially a price
- Network: represents strength of relationships

Insider

Security expert


Electronics expert


Mechanic

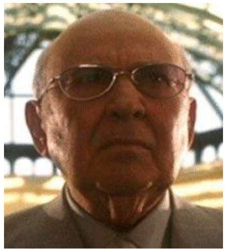
Pick-pocket thief

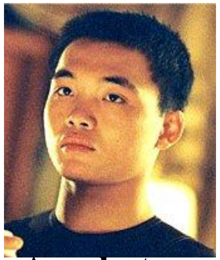
Organizer

Co-organizer

Mechanic

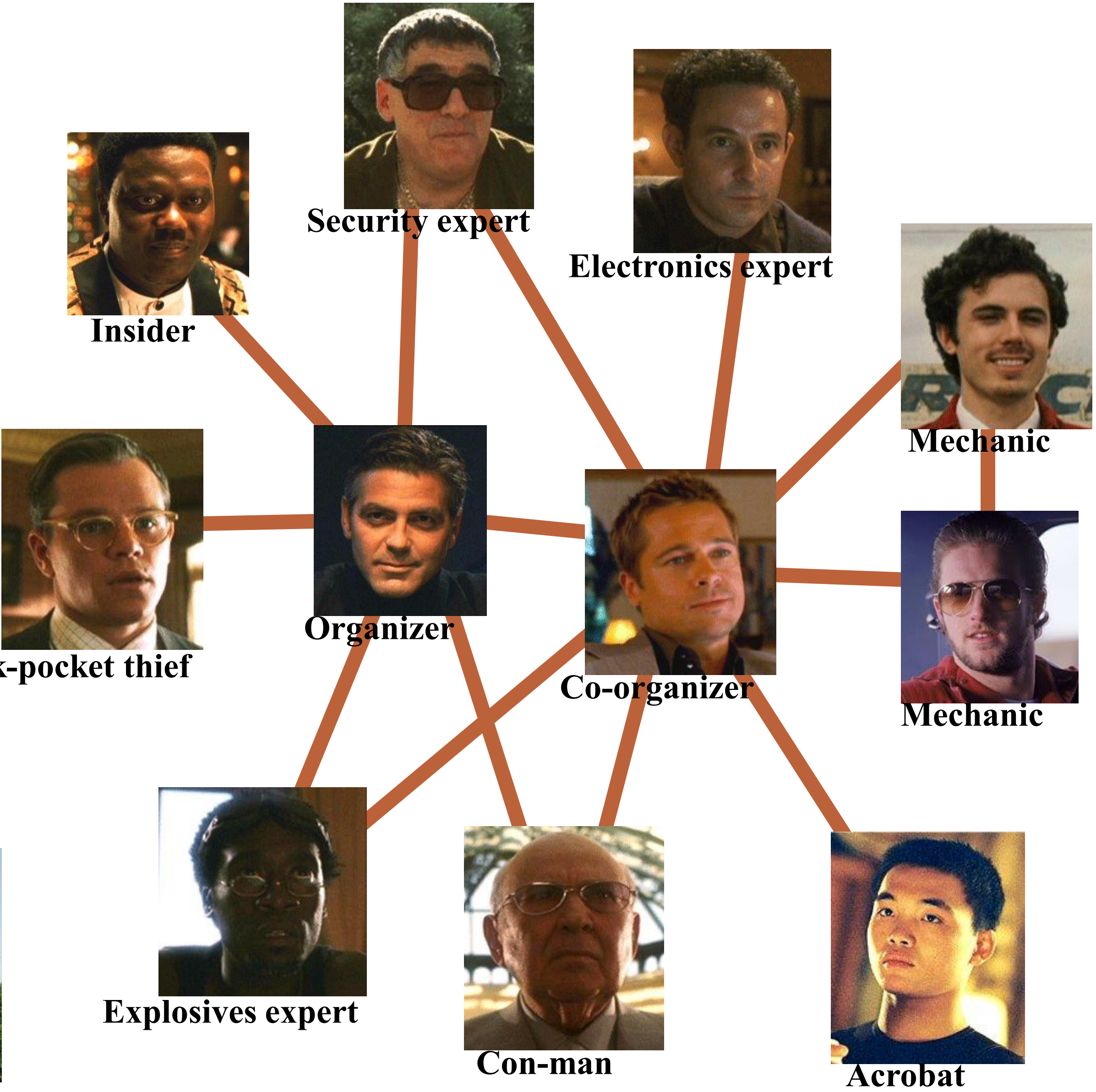
Explosives expert


Con-man


Acrobat

george clooney
matt damon
andy garcia
brad pitt
and julia roberts
OCEAN'S ELEVEN
Place your bets.

MIRAGE
GRAND


Security expert
Electronics expert
Insider
Mechanic
Pick-pocket thief
Organizer
Co-organizer
Mechanic
Explosives expert
Con-man
Acrobat

Applications

- Collaboration networks (e.g., scientists, actors)
- Organizational structure of companies
- LinkedIn, UpWork, FreeLance
- Geographical (map) of experts

Simple Team formation Problem

- Input:
  - A task T, consisting of a set of skills
  - A set of candidate experts each having a subset of skills

T = {algorithms, java, graphics, python}

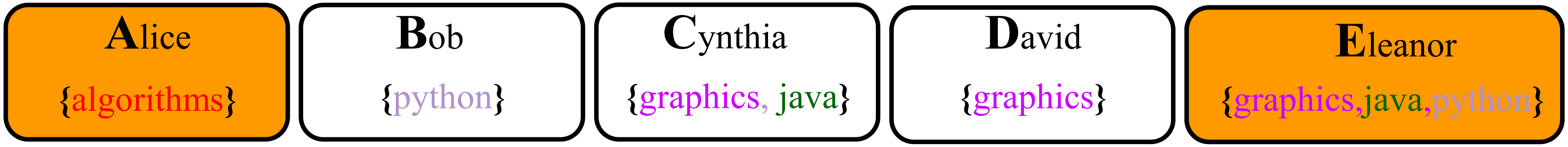
Alice	Bob	Cynthia	David	Eleanor
{algorithms}	{python}	{graphics, java}	{graphics}	{graphics,java,python}

- Problem: Given a task and a set of experts, find the smallest subset (team) of experts that together have all the required skills for the task

Coverage

- The Simple Team Formation Problem is a just an instance of the Set Cover problem
  - Universe U of elements = Set of all skills
  - Collection S of subsets = The set of experts and the subset of skills they possess.

T = {algorithms, java, graphics, python}



Alice
{algorithms}

Bob
{python}

Cynthia
{graphics, java}

David
{graphics}

Eleanor
{graphics,java,python}

Team formation in the presence of a social network

- Given a task and a set of experts organized in a network find the subset of experts that can effectively perform the task

- Task: set of required skills

- Expert: has a set of skills

- Network: represents strength of relationships

- Effectively: There is good communication between the team members

Coverage is NOT enough

T={**algorithms**,**java**,**graphics**,**python**}

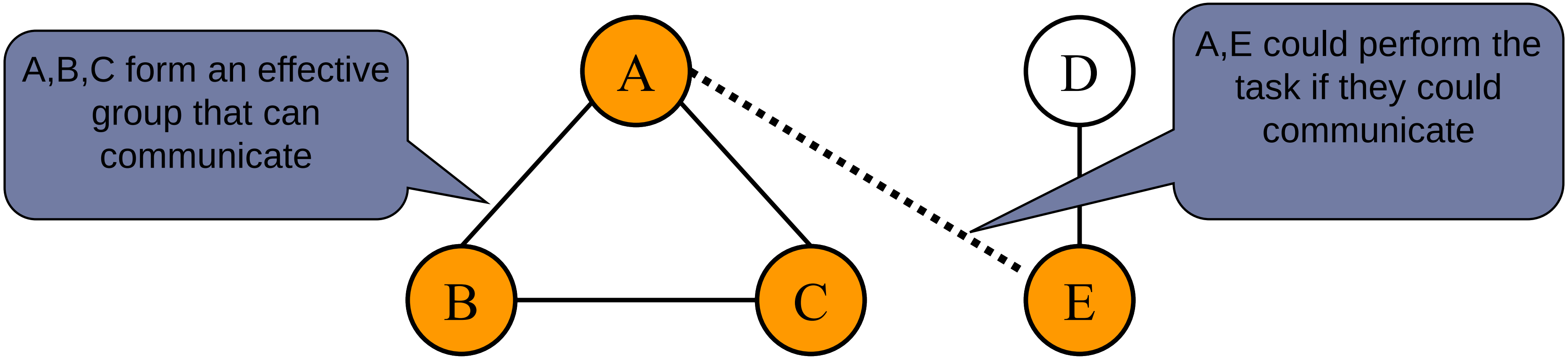
Alice
{algorithms}

Bob
{python}

Cynthia
{graphics, java}

David
{graphics}

Eleanor
{graphics,java,python}



A,B,C form an effective group that can communicate

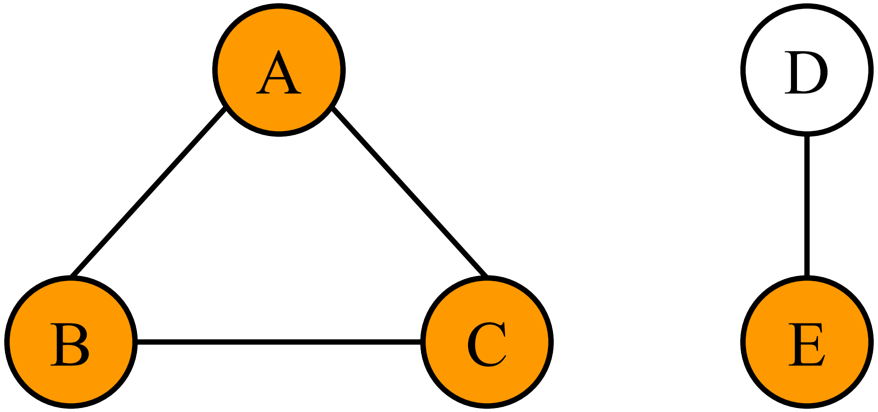
A,E could perform the task if they could communicate

Communication: the members of the team must be able to efficiently communicate and work together

How to measure effective communication?

The longest shortest path between any two nodes in the subgraph

- Diameter of the subgraph defined by the group members

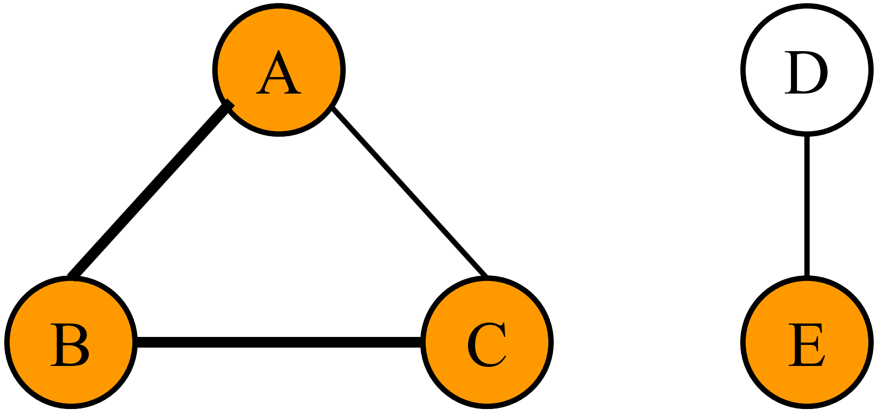


diameter = 1

How to measure effective communication?

The total weight of the edges of a tree that spans all the team nodes

- MST (Minimum spanning tree) of the subgraph defined by the group members



MST = 2

Problem definition (MinDiameter)

- Given a task and a social network G of experts, find the subset (team) of experts that can perform the given task and they define a subgraph G' in G with the minimum diameter.

- Problem is NP-hard
- Equivalent to the Multiple Choice Cover (MCC)
  - We have a set cover instance $(U, \mathbf{S})$, but we also have a distance matrix D with distances between the different sets in $\mathbf{S}$.
  - We want a cover that has the minimum diameter (minimizes the largest pairwise distance in the cover)

The RarestFirst algorithm

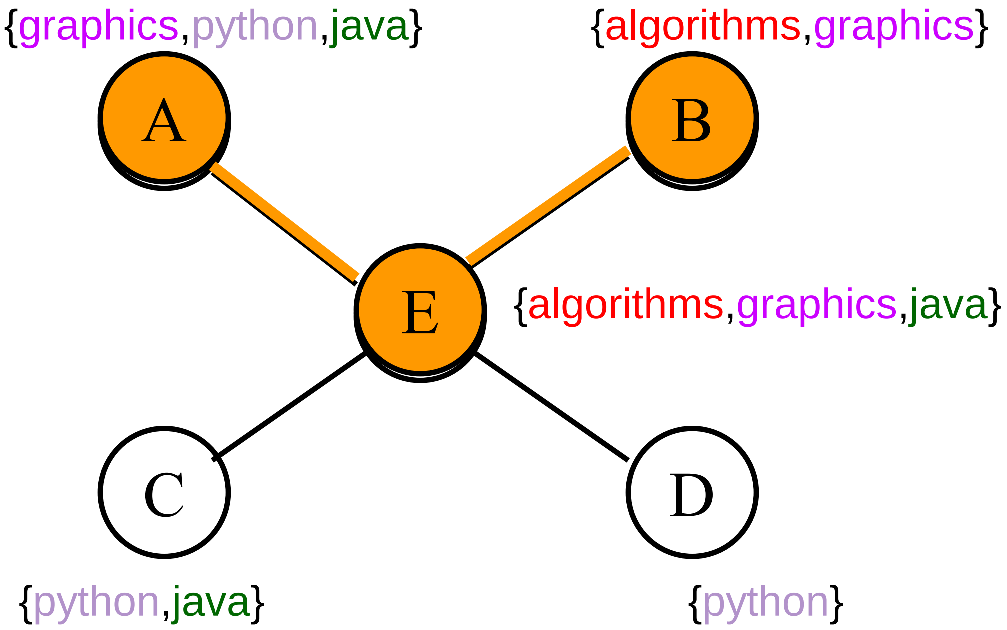
- Compute all tree distances from the input graph G and create a new complete graph G_C
- Find Rarest skill α_{rare} required for a task
- S_{rare} = group of people that have α_{rare}
- Evaluate star graphs in G_C, centered at individuals from S_{rare}
- Report cheapest star

Running time: Quadratic to the number of nodes

Approximation factor: 2xOPT

The RarestFirst algorithm

T={algorithms,java,graphics,python}



Skills:

algorithms

graphics

java

python

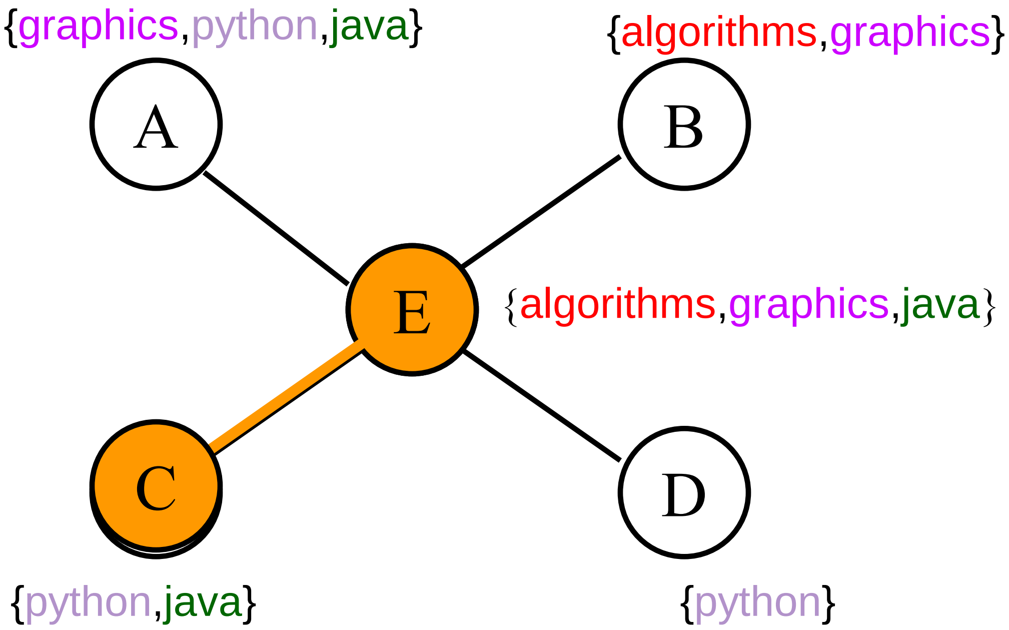
α_{rare} = algorithms

S_{rare} ={Bob, Eleanor}

Diameter = 2

The RarestFirst algorithm

T={algorithms,java,graphics,python}



Skills:

algorithms

graphics

java

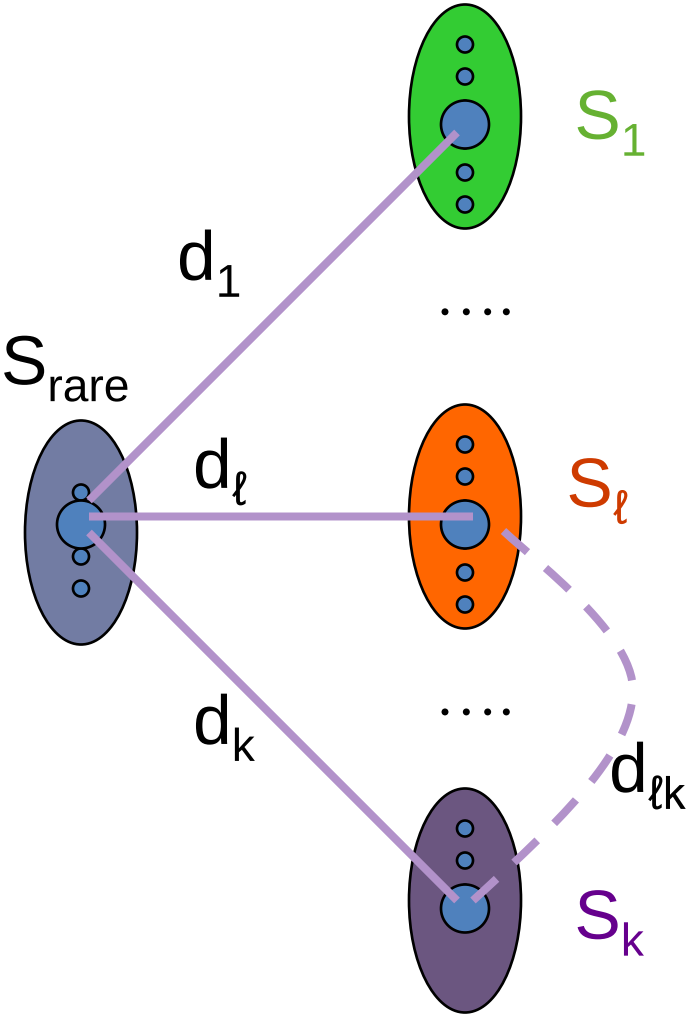
python

α_{rare} = algorithms

S_{rare} ={Bob, Eleanor}

Diameter = 1

Analysis of RarestFirst



- The diameter is
  - either $D = d_k$, for some node k,
  - or $D = d_{\ell k}$ for some pair of nodes ℓ, k
- Fact: $OPT \geq d_k$
- Fact: $OPT \geq d_\ell$
- $D \leq d_{\ell k} \leq d_\ell + d_k \leq 2*OPT$

Problem definition (MinMST)

- Given a task and a social network G of experts, find the subset (team) of experts that can perform the given task and they define a subgraph G' in G with the minimum MST cost.

- Problem is NP-hard
- Follows from a connection with Group Steiner Tree problem

The SteinerTree problem

- Graph G(V,E)

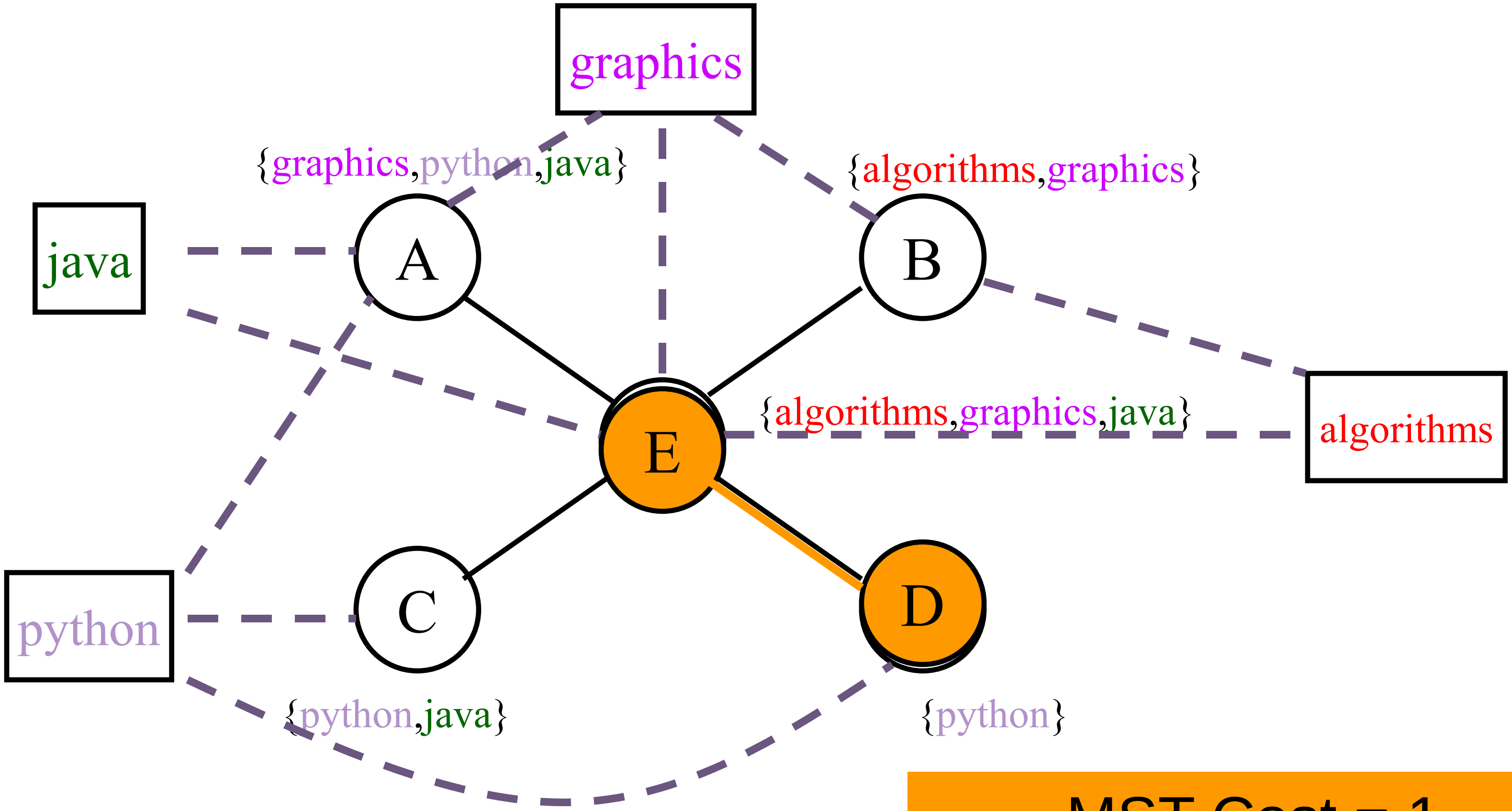
Required vertices

- Partition of V into V = {R,N}

- Find G' subgraph of G such that G' contains all the required vertices (R) and MST(G') is minimized
  - Find the cheapest tree that contains all the required nodes.

The EnhancedSteiner algorithm

T={algorithms,java,graphics,python}



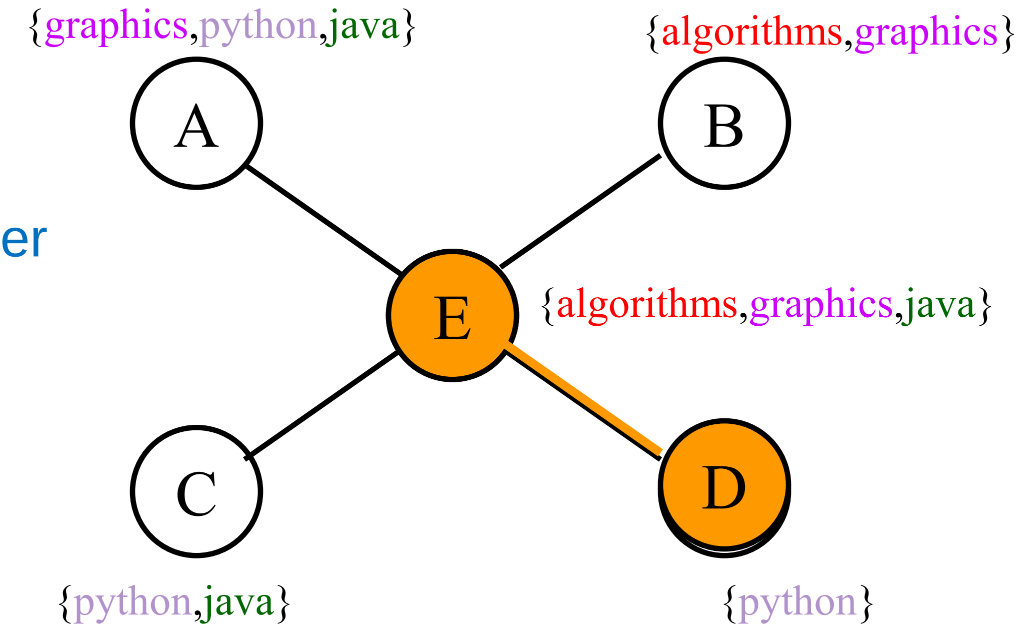
MST Cost = 1

The CoverSteiner algorithm

T={algorithms,java,graphics,python}

1. Solve SetCover

- Solve Steiner

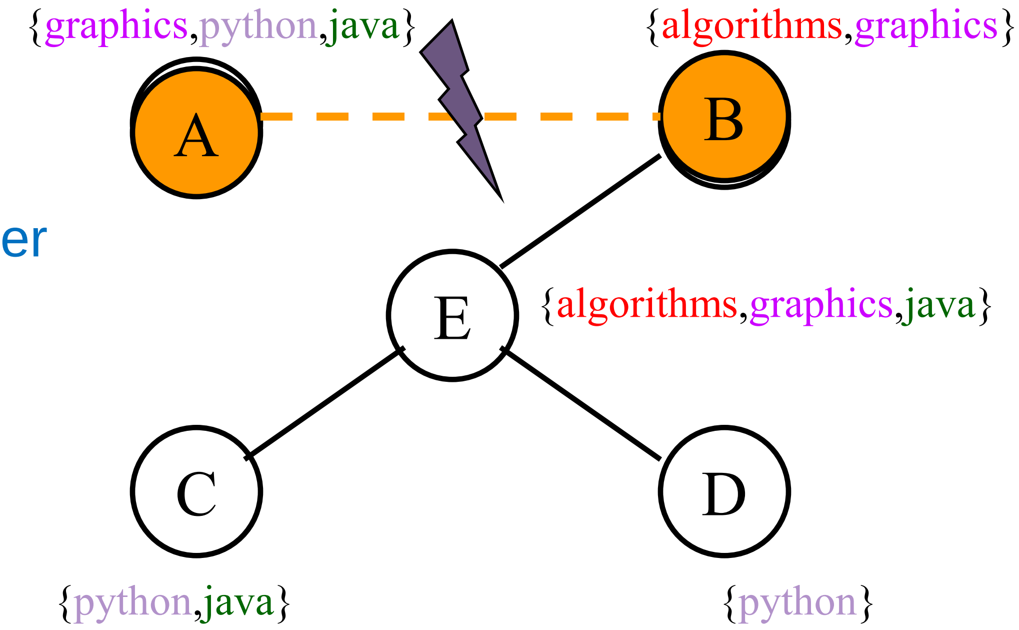


MST Cost = 1

How good is CoverSteiner?

T={algorithms,java,graphics,python}

1. Solve SetCover
- Solve Steiner



MST Cost = Infty

References

Theodoros Lappas, Kun Liu, Evimaria Terzi, Finding a team of experts in social networks. KDD 2009: 467-476

Thanks to: Jure Leskovec, Anand Rajaraman, Jeff Ullman

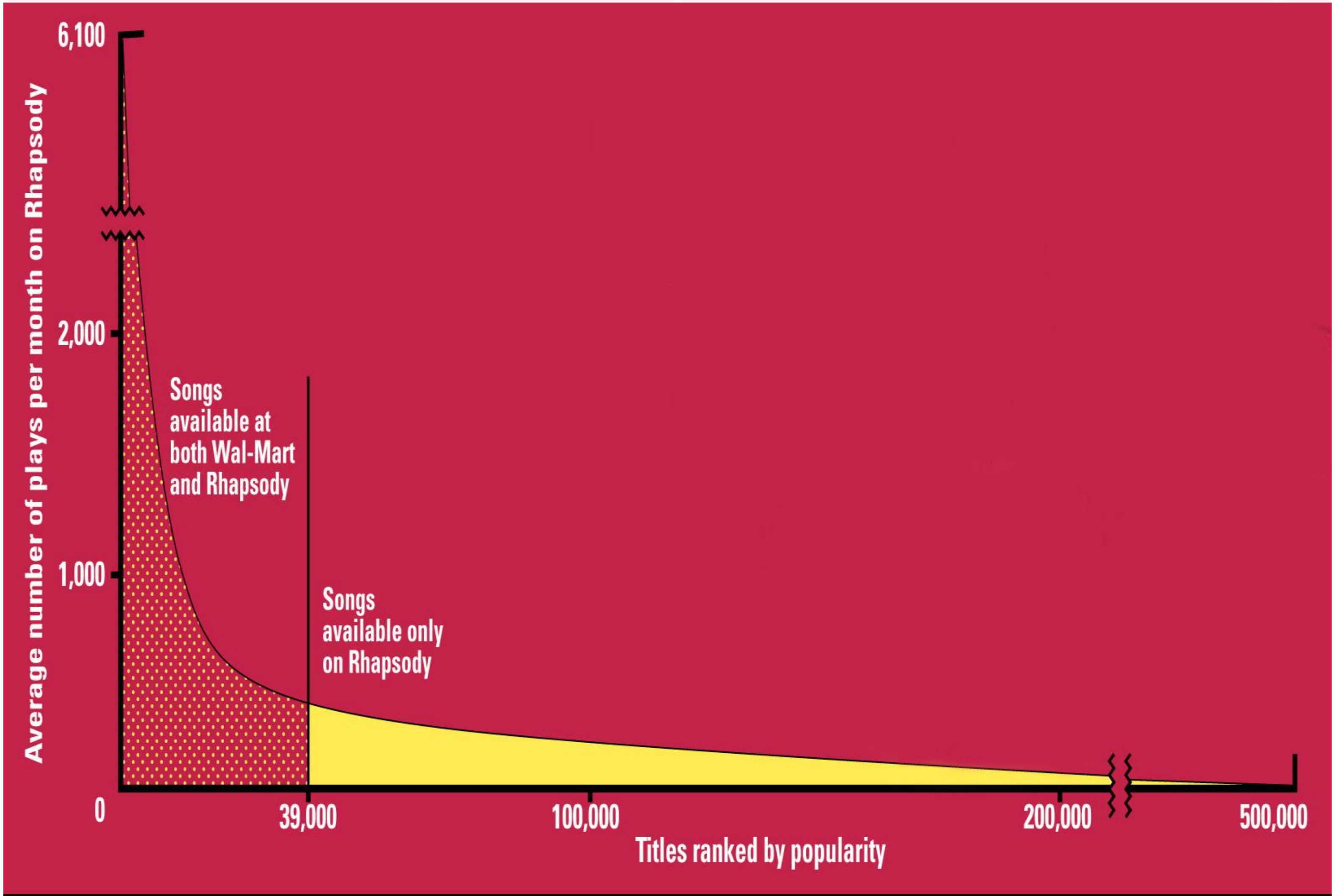
RECOMMENDATION SYSTEMS AND SOCIAL RECOMMENDATIONS

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

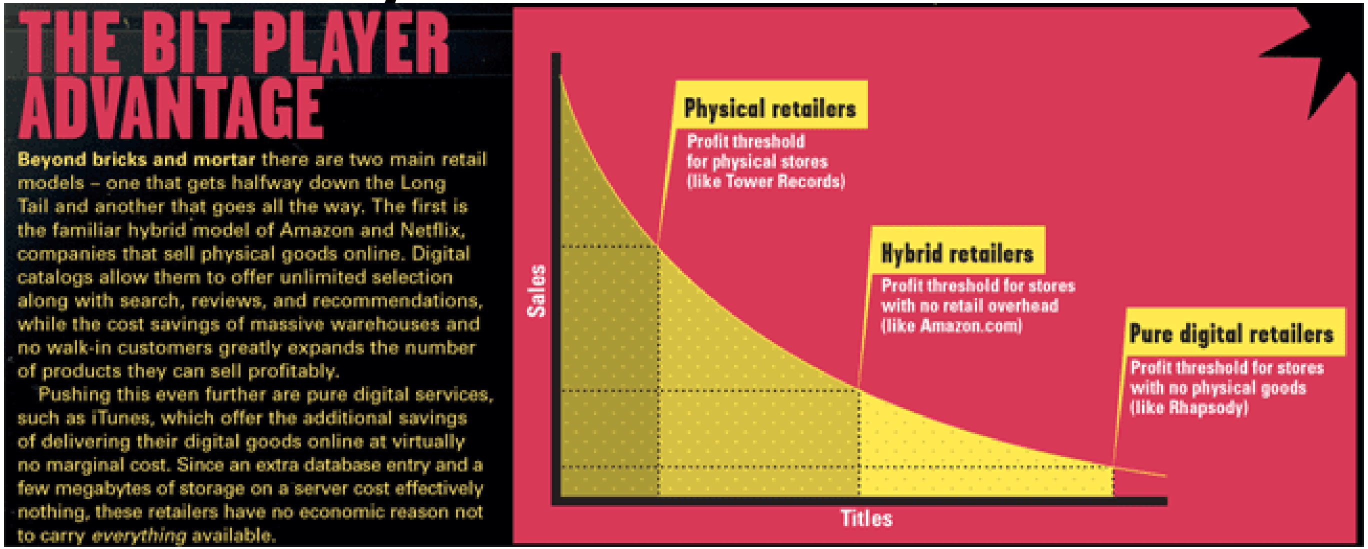
The Long Tail



Source: Chris Anderson (2004)

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

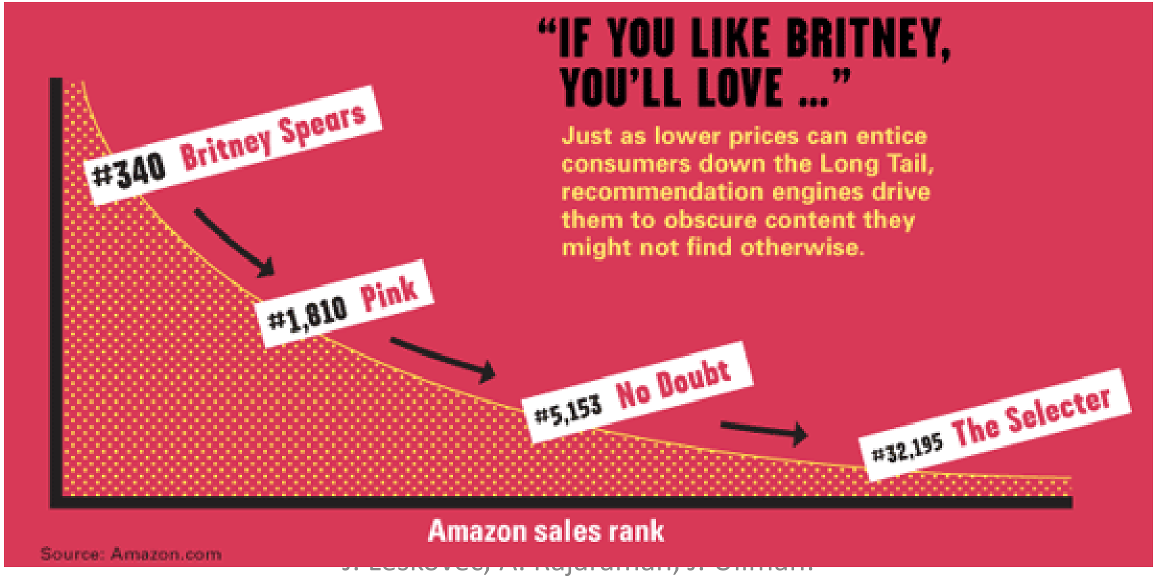
Physical vs. Online



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!

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

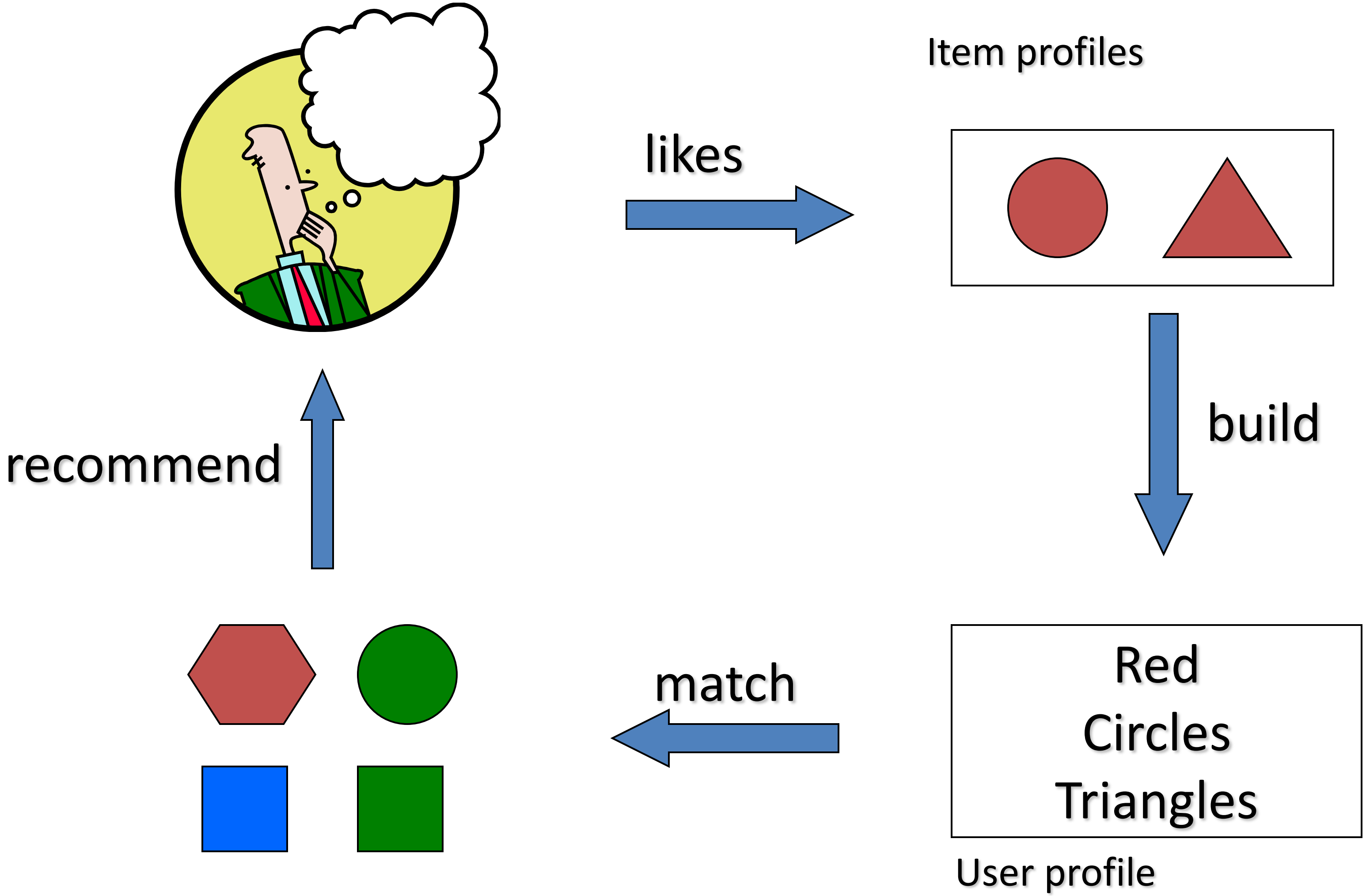
Content-based prediction

	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



Item profiles

likes

build

Red
Circles
Triangles

User profile

match

recommend

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

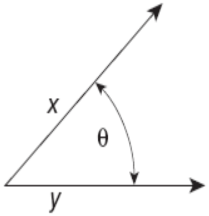
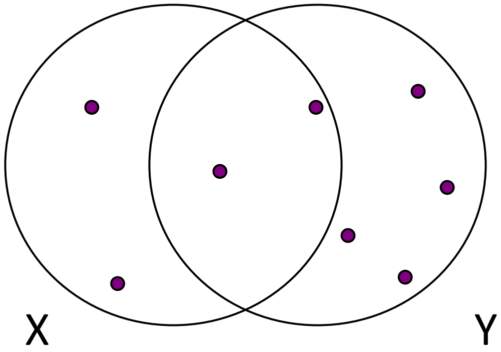


Figure 2.16. Geometric illustration of the cosine measure.

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

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

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



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.
Jsim(A,B) = 1/5
Jsim(A,C) = 1/2
Jsim(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:

Cos(A,B) = 0.38

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)

Corr(A,B) = 0.092
Corr(A,C) = -0.559

User-User Collaborative Filtering

- Consider user c
- Find set D of other users whose ratings are most "similar" to c's ratings
- Estimate user's ratings based on ratings of users in D using some aggregation function

$$r_{ui} = \sum_{v \in TopK(u)} \mathrm{sim}(u,v) r_{vi}$$

  - Modeling deviations:

$$r_{ui} = \bar{r_u} + \sum_{v \in TopK(u)} \mathrm{sim}(u,v)(\bar{r_v} - r_{vi})$$

- 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 c and item i
  - Find the set D of most similar items to item i that have been rated by user c.
  - 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:

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

  - 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

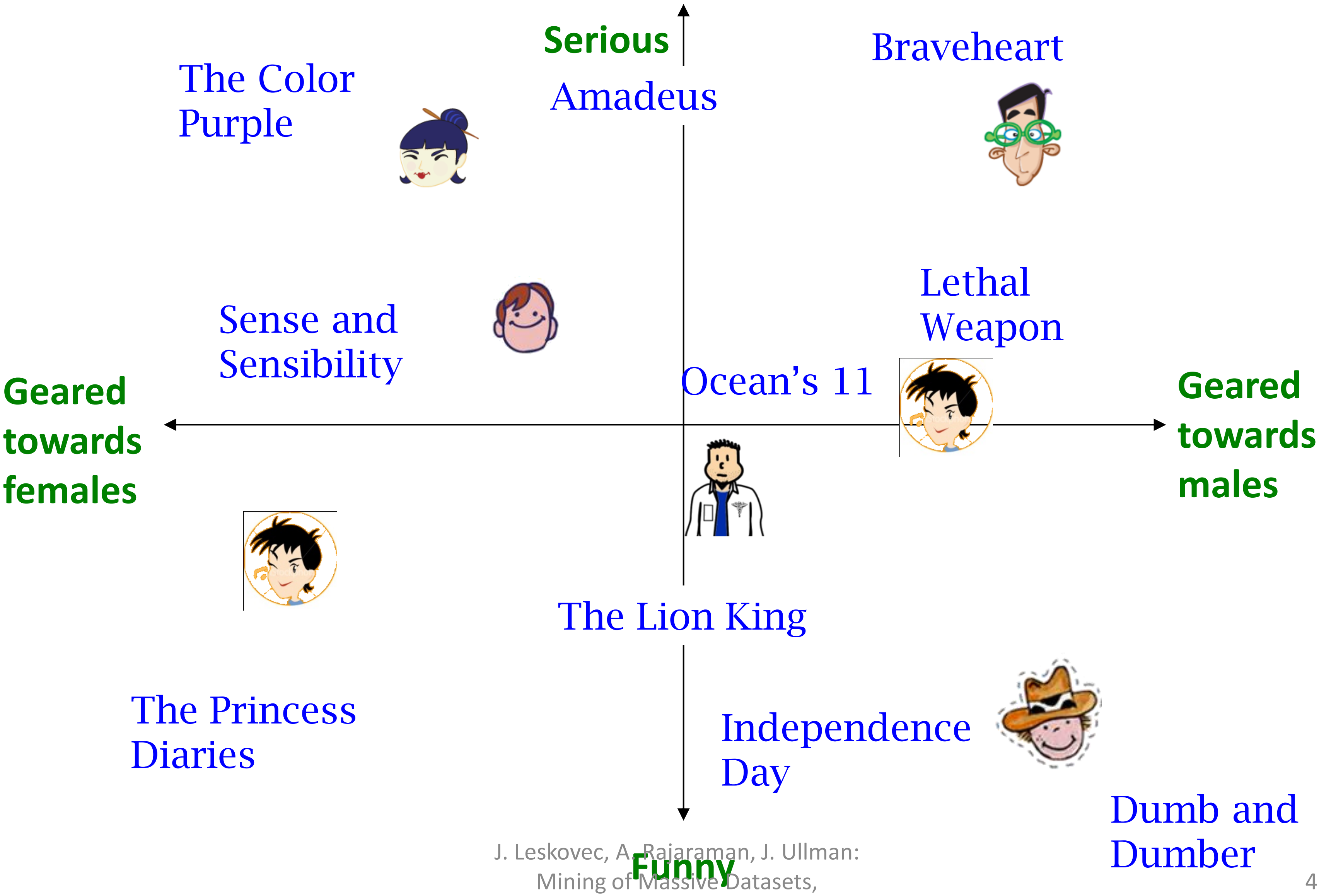
  - Kendal' tau for measuring the quality of predicting the ranking of items:
    - The fraction of pairs of items that are ordered correctly
      – The fraction of pairs that are ordered incorrectly

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



Serious

Braveheart

The Color Purple

Amadeus

Lethal Weapon

Sense and Sensibility

Geared towards females

Ocean's 11

Geared towards males

The Lion King

The Princess Diaries

Independence Day

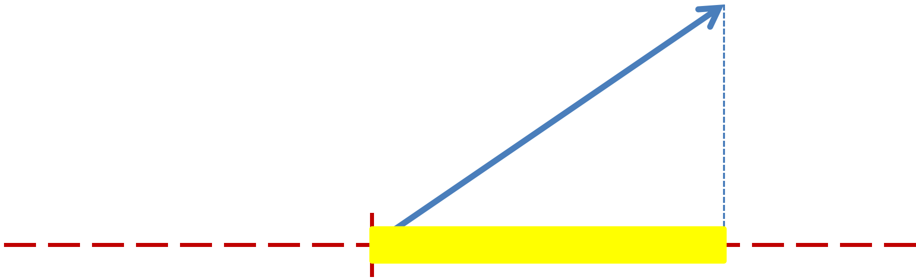
Dumb and Dumber

Funny

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 **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 $c = [1,2,3]^T$
- 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

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

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

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 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 **v**
  - Strength of the i-th linear trend: $||Av_i|| = \boldsymbol{\sigma_i}$

Symmetric matrices

- Special case: A is symmetric positive definite matrix

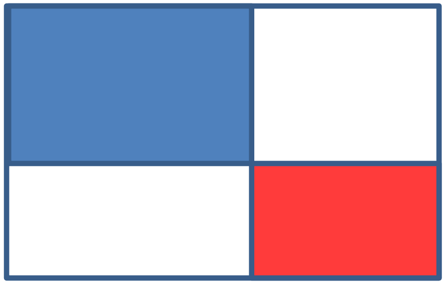
$$A = \lambda_1 u_1 u_1^T + \lambda_2 u_2 u_2^T + \cdots + \lambda_r u_r u_r^T$$

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

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

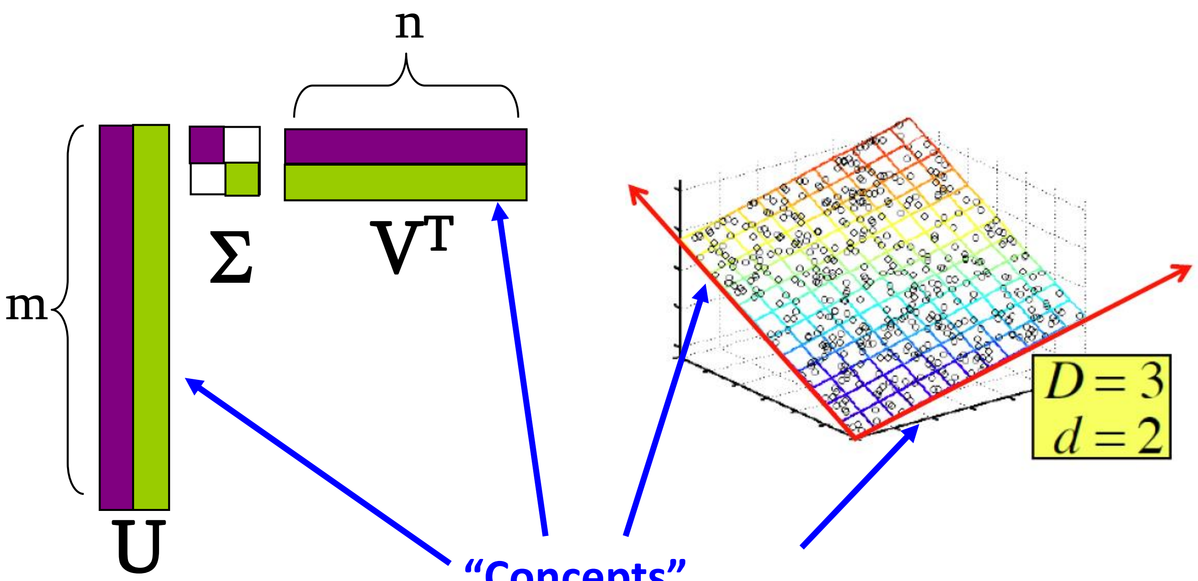
- A is a rank-2 matrix

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

SVD – Example: Users-to-Movies

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

$$
\begin{array}{c|ccccc}
 & \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\
\hline
\text{SciFi} & 1 & 1 & 1 & 0 & 0 \\
 & 3 & 3 & 3 & 0 & 0 \\
 & 4 & 4 & 4 & 0 & 0 \\
 & 5 & 5 & 5 & 0 & 0 \\
\text{Romance} & 0 & 0 & 0 & 4 & 4 \\
 & 0 & 0 & 0 & 5 & 5 \\
 & 0 & 0 & 0 & 2 & 2
\end{array}
=
$$



"Concepts"
AKA Latent dimensions
AKA Latent factors

SVD – Example: Users-to-Movies

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

$$
\begin{array}{c}
 \\
\uparrow \\
 \\
\text{SciFi} \\
\downarrow \\
\uparrow \\
\text{Romance} \\
\downarrow
\end{array}
\begin{array}{c}
\begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\
\left[\begin{array}{ccccc}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & 0 & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & 0 & 0 & \mathbf{2} & \mathbf{2}
\end{array}\right]
\end{array}
=
\begin{array}{c}
\begin{array}{cc} \text{SciFi-concept} & \text{Romance-concept} \end{array} \\
\left[\begin{array}{cc}
\mathbf{0.14} & 0.00 \\
\mathbf{0.42} & 0.00 \\
\mathbf{0.56} & 0.00 \\
\mathbf{0.70} & 0.00 \\
0.00 & \mathbf{0.60} \\
0.00 & \mathbf{0.75} \\
0.00 & \mathbf{0.30}
\end{array}\right]
\end{array}
\times
\left[\begin{array}{cc}
\mathbf{12.4} & 0 \\
0 & \mathbf{9.5}
\end{array}\right]
\times
\left[\begin{array}{ccccc}
\mathbf{0.58} & \mathbf{0.58} & \mathbf{0.58} & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & \mathbf{0.71} & \mathbf{0.71}
\end{array}\right]
$$

SVD – Example: Users-to-Movies

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

SciFi-concept

Romance-concept

$$
\begin{array}{c|ccccc}
 & \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\
\hline
 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
 & \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\text{SciFi} & \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
 & \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
 & 0 & 0 & 0 & \mathbf{4} & \mathbf{4} \\
\text{Romance} & 0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
 & 0 & 0 & 0 & \mathbf{2} & \mathbf{2}
\end{array}
=
\begin{bmatrix}
\mathbf{0.14} & 0.00 \\
\mathbf{0.42} & 0.00 \\
\mathbf{0.56} & 0.00 \\
\mathbf{0.70} & 0.00 \\
0.00 & \mathbf{0.60} \\
0.00 & \mathbf{0.75} \\
0.00 & \mathbf{0.30}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 \\
0 & \mathbf{9.5}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.58} & \mathbf{0.58} & \mathbf{0.58} & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & \mathbf{0.71} & \mathbf{0.71}
\end{bmatrix}
$$

U is "user-to-concept" similarity matrix

SVD – Example: Users-to-Movies

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

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

V is “movie to concept” similarity matrix

SVD – Example: Users-to-Movies

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

SciFi-concept

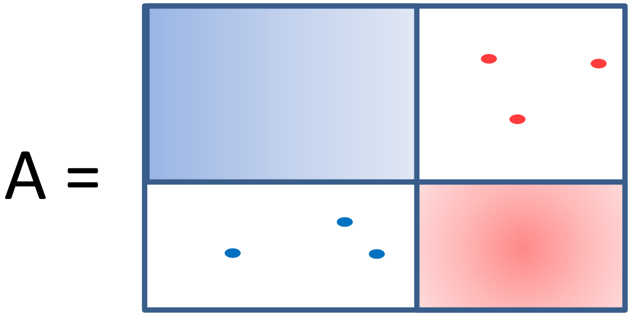
Romance-concept

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

Σ is the "concept strength" matrix

An (more realistic) example

- User-Movie matrix



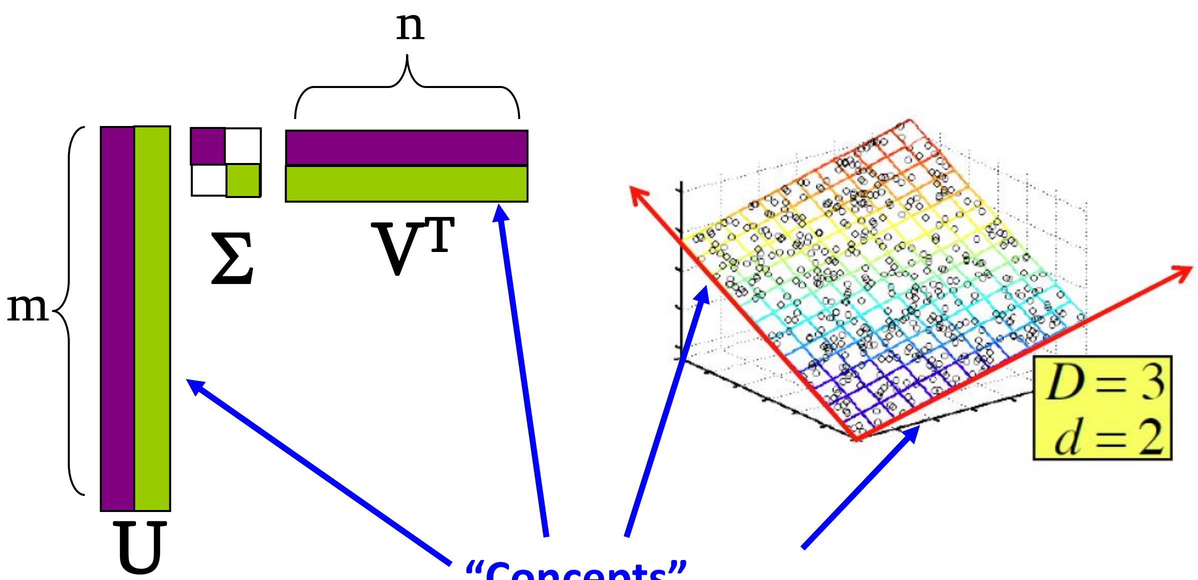
- There are two prototype users and movies but they are noisy

SVD – Example: Users-to-Movies

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

A more realistic scenario

$$\begin{array}{c|ccccc}
 & \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\
\hline
\uparrow\ \text{SciFi} & 1 & 1 & 1 & 0 & 0 \\
 & 3 & 3 & 3 & 0 & 0 \\
 & 4 & 4 & 4 & 0 & 0 \\
\downarrow & 5 & 5 & 5 & 0 & 0 \\
\uparrow\ \text{Romnce} & 0 & 2 & 0 & 4 & 4 \\
 & 0 & 0 & 0 & 5 & 5 \\
\downarrow & 0 & 1 & 0 & 2 & 2
\end{array} =$$



"Concepts"
AKA Latent dimensions
AKA Latent factors

SVD – Example: Users-to-Movies

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

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

SVD – Example: Users-to-Movies

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

SciFi-concept

Romance-concept

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

The first two vectors are more or less unchanged

SVD – Example: Users-to-Movies

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

The third vector has a very low singular value

$$
\begin{array}{c|ccccc}
 & \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\
\hline
\uparrow & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\text{SciFi} & \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\downarrow & \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
 & \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
\uparrow & 0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
\text{Romance} & 0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
\downarrow & 0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{array}
=
\begin{bmatrix}
\mathbf{0.13} & -0.02 & -0.01 \\
\mathbf{0.41} & -0.07 & -0.03 \\
\mathbf{0.55} & -0.09 & -0.04 \\
\mathbf{0.68} & -0.11 & -0.05 \\
0.15 & \mathbf{0.59} & \mathbf{0.65} \\
0.07 & \mathbf{0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \boxed{\mathbf{1.3}}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
-0.12 & 0.02 & -0.12 & \mathbf{0.69} & \mathbf{0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

SVD - Interpretation

‘**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 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

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

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

Example

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

Example

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

Example

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

Frobenius norm:

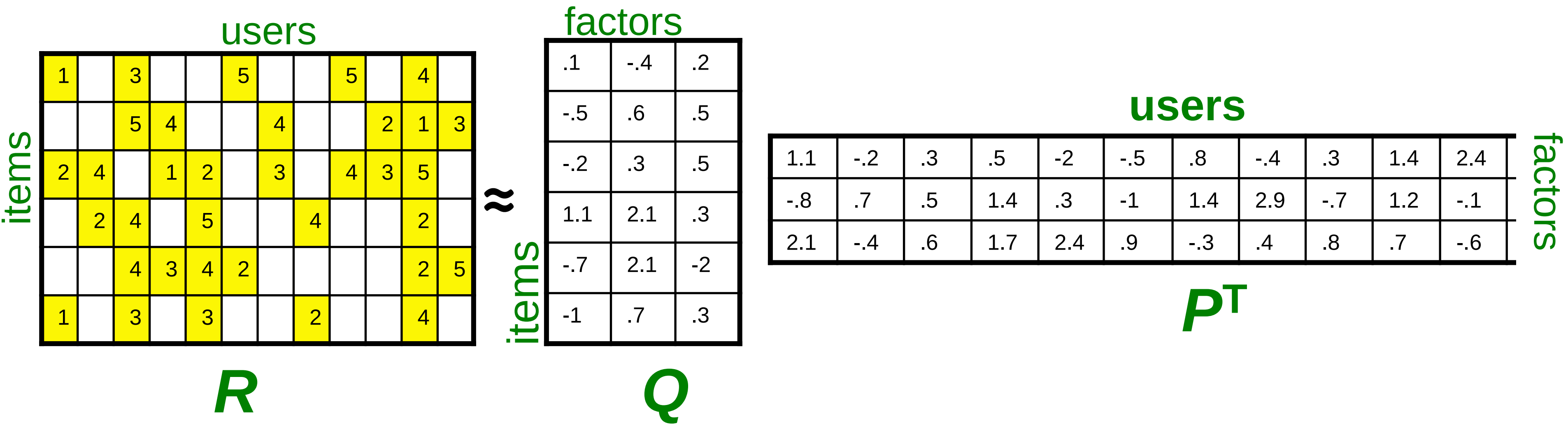
$\|M\|_F = \sqrt{\Sigma_{ij} M_{ij}^2}$

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

Latent Factor Models

- **“SVD” on Netflix data: $\mathbf{R} \approx \boldsymbol{Q} \cdot \boldsymbol{P}^T$**

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



R (items × users):

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	

Q (items × factors):

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

P^T (factors × users):

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

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

≈

$$\hat{r}_{xi} = q_i \cdot p_x$$
$$= \sum_f q_{if} \cdot p_{xf}$$

q_i = row ***i*** of ***Q***
p_x = column ***x*** of ***P***$^{\text{T}}$

items

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

factors

Q

•

users

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$^{\text{T}}$

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	

≈

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

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

q

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

factors

Q

•

users

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

$\boldsymbol{P}^T$

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	**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	

≈

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

$$= \sum_{f} q_{if} \cdot p_{xf}$$

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

Q

•

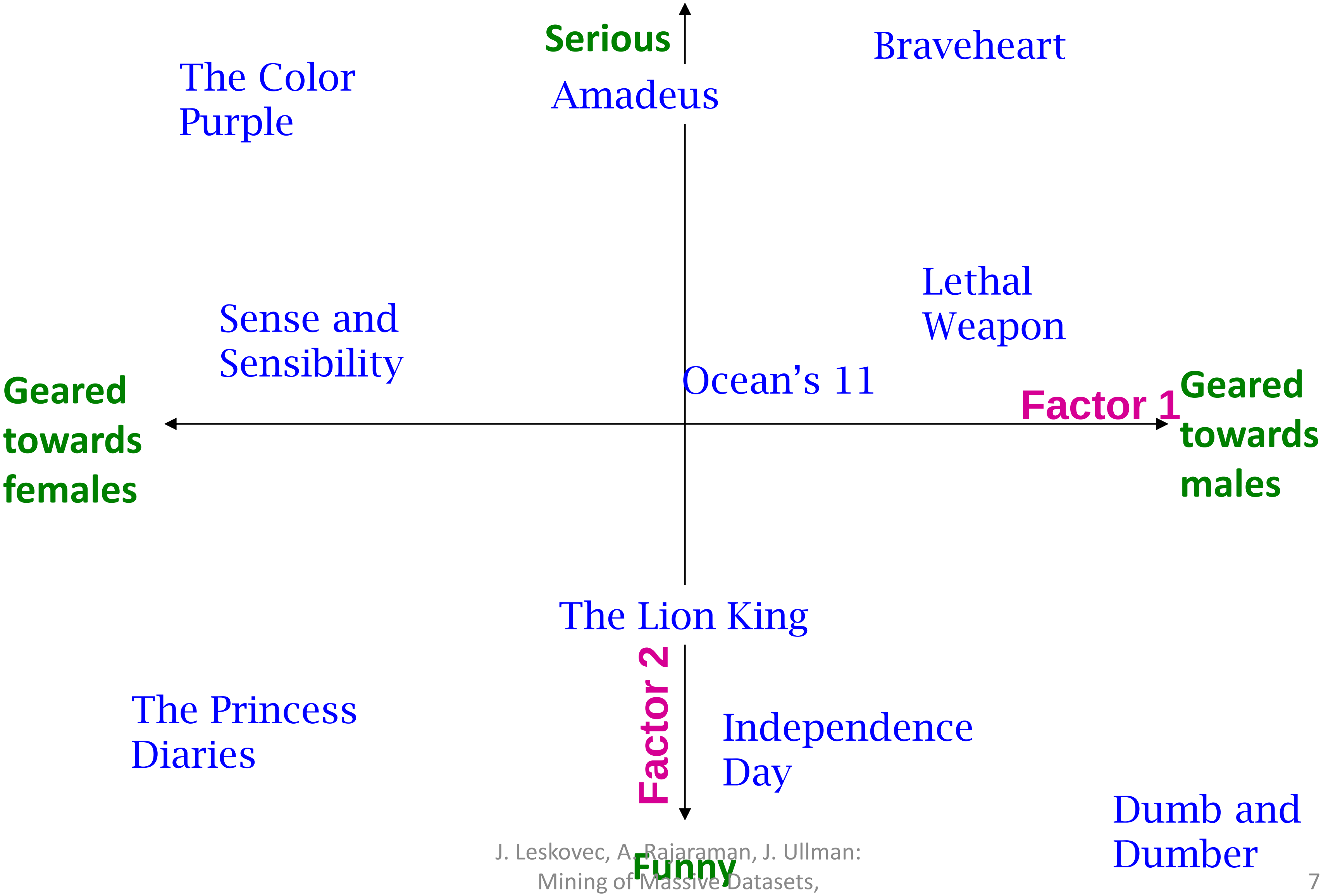
users

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

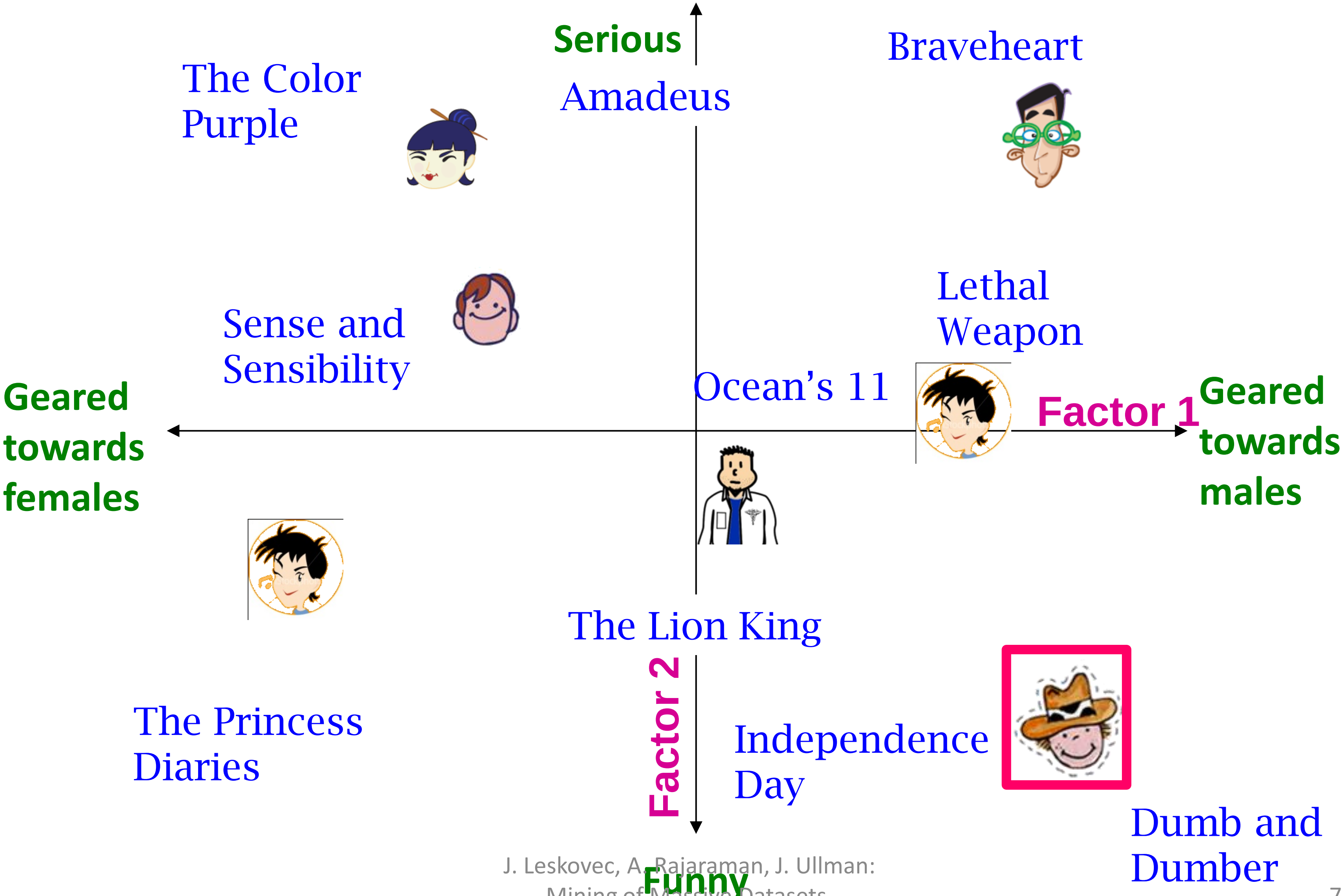
P^T

Latent Factor Models



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

Latent Factor Models



Serious

The Color Purple

Amadeus

Braveheart

Sense and Sensibility

Lethal Weapon

Geared towards females

Ocean’s 11

Factor 1

Geared towards males

The Lion King

Factor 2

The Princess Diaries

Independence Day

Dumb and Dumber

Funny

Example

Missing ratings

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

Example

- Reconstruction of missing ratings

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

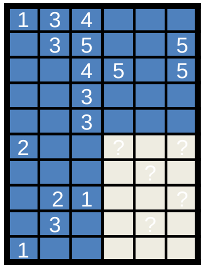
Latent Factor Models

$$\text{items}\;\overset{\text{users}}{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 1 & & 3 & & & 5 & & & 5 & & 4 & \\ \hline & & 5 & 4 & & & 4 & & & 2 & 1 & 3 \\ \hline 2 & 4 & & 1 & 2 & & 3 & & 4 & 3 & 5 & \\ \hline & 2 & 4 & & 5 & & & 4 & & & 2 & \\ \hline & & 4 & 3 & 4 & 2 & & & & & 2 & 5 \\ \hline 1 & & 3 & & 3 & & & 2 & & & 4 & \\ \hline\end{array}} \approx \text{items}\;\underset{Q}{\overset{\text{factors}}{\begin{array}{|c|c|c|}\hline .1 & -.4 & .2 \\ \hline -.5 & .6 & .5 \\ \hline -.2 & .3 & .5 \\ \hline 1.1 & 2.1 & .3 \\ \hline -.7 & 2.1 & -2 \\ \hline -1 & .7 & .3 \\ \hline\end{array}}}\;\underset{P^T}{\overset{\text{users}}{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 1.1 & -.2 & .3 & .5 & -2 & -.5 & .8 & -.4 & .3 & 1.4 & 2.4 & -.9 \\ \hline -.8 & .7 & .5 & 1.4 & .3 & -1 & 1.4 & 2.9 & -.7 & 1.2 & -.1 & 1.3 \\ \hline 2.1 & -.4 & .6 & 1.7 & 2.4 & .9 & -.3 & .4 & .8 & .7 & -.6 & .1 \\ \hline\end{array}}}\;\text{factors}$$

- **SVD also considers entries that are missing!**
- **Use specialized methods to find *P, Q***
  - $\min_{P,Q} \sum_{(i,x)\in R} (r_{xi} - q_i \cdot p_x)^2$ $\qquad \hat{r}_{xi} = q_i \cdot p_x$
  - **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

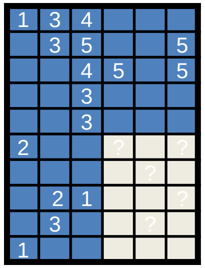
- **Want to minimize SSE for unseen test data**
- **Idea: Minimize SSE on training data**
  - Want large ***k*** (# of factors) to capture all the signals
  - But, **SSE** on **test data** begins to rise for ***k*** > 2



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

- 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



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

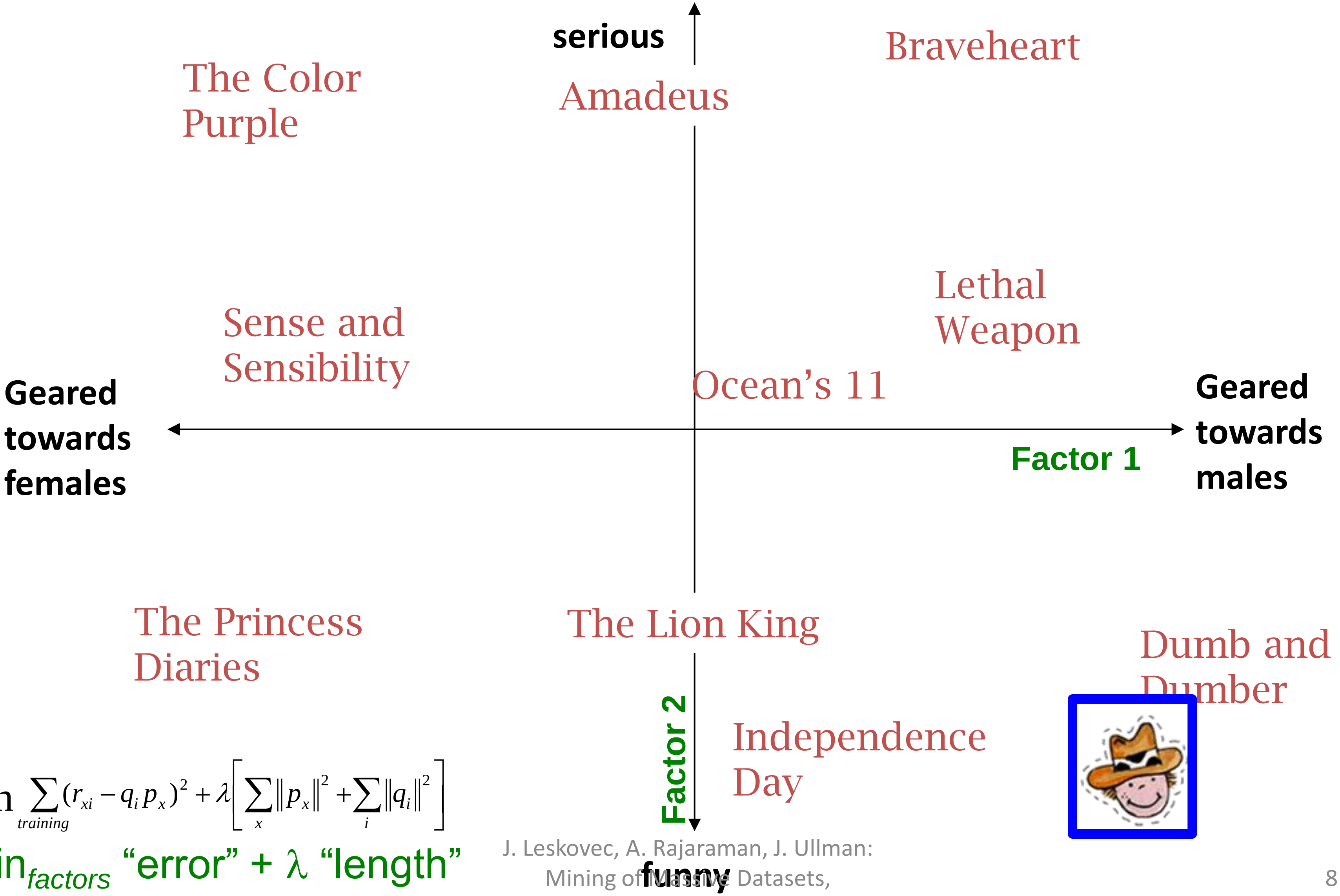
- **To solve overfitting we introduce regularization:**
  - Allow rich model where there are sufficient data
  - Shrink aggressively where data are scarce

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

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

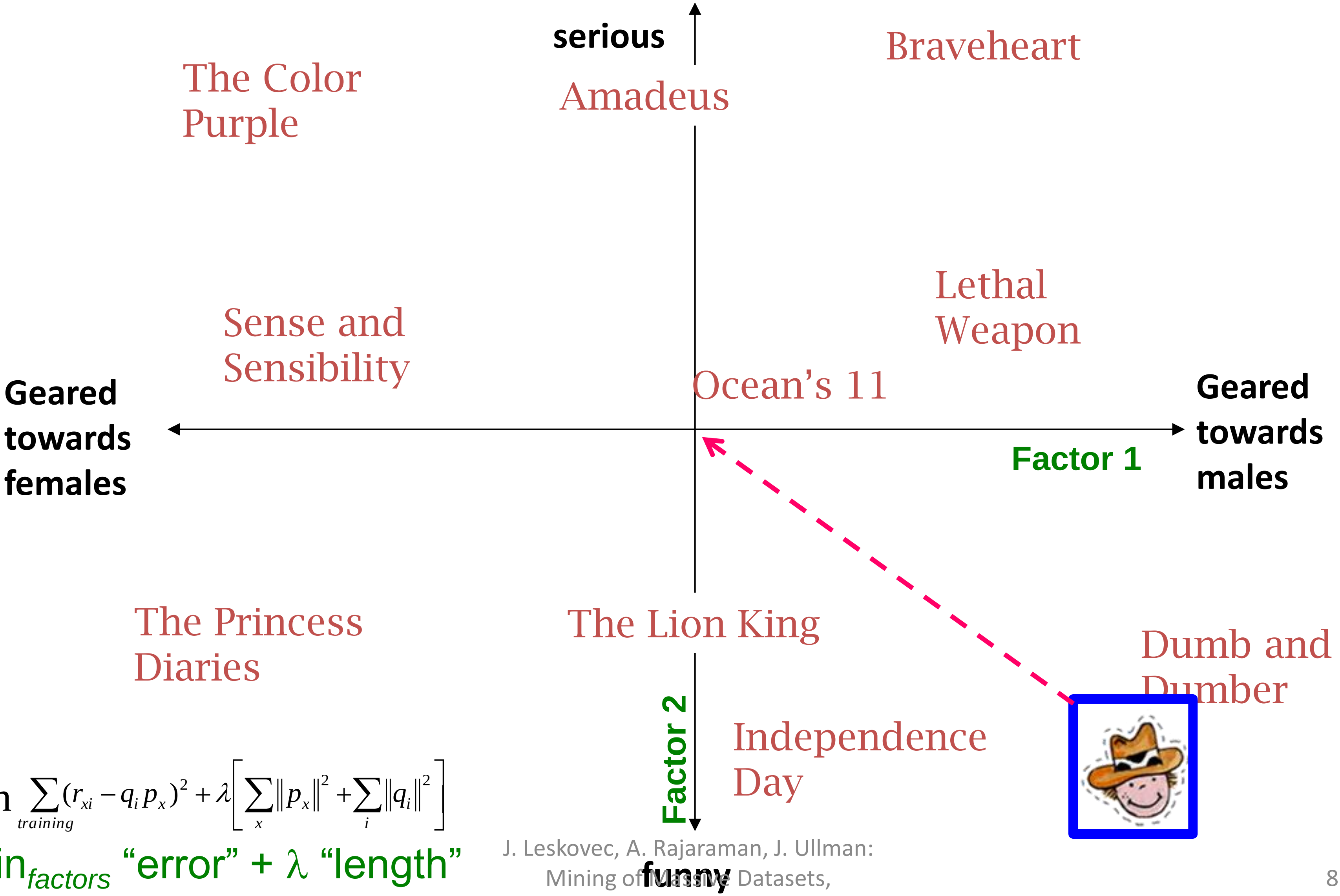
The Effect of Regularization



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

$\min_{factors}$ "error" + λ "length"

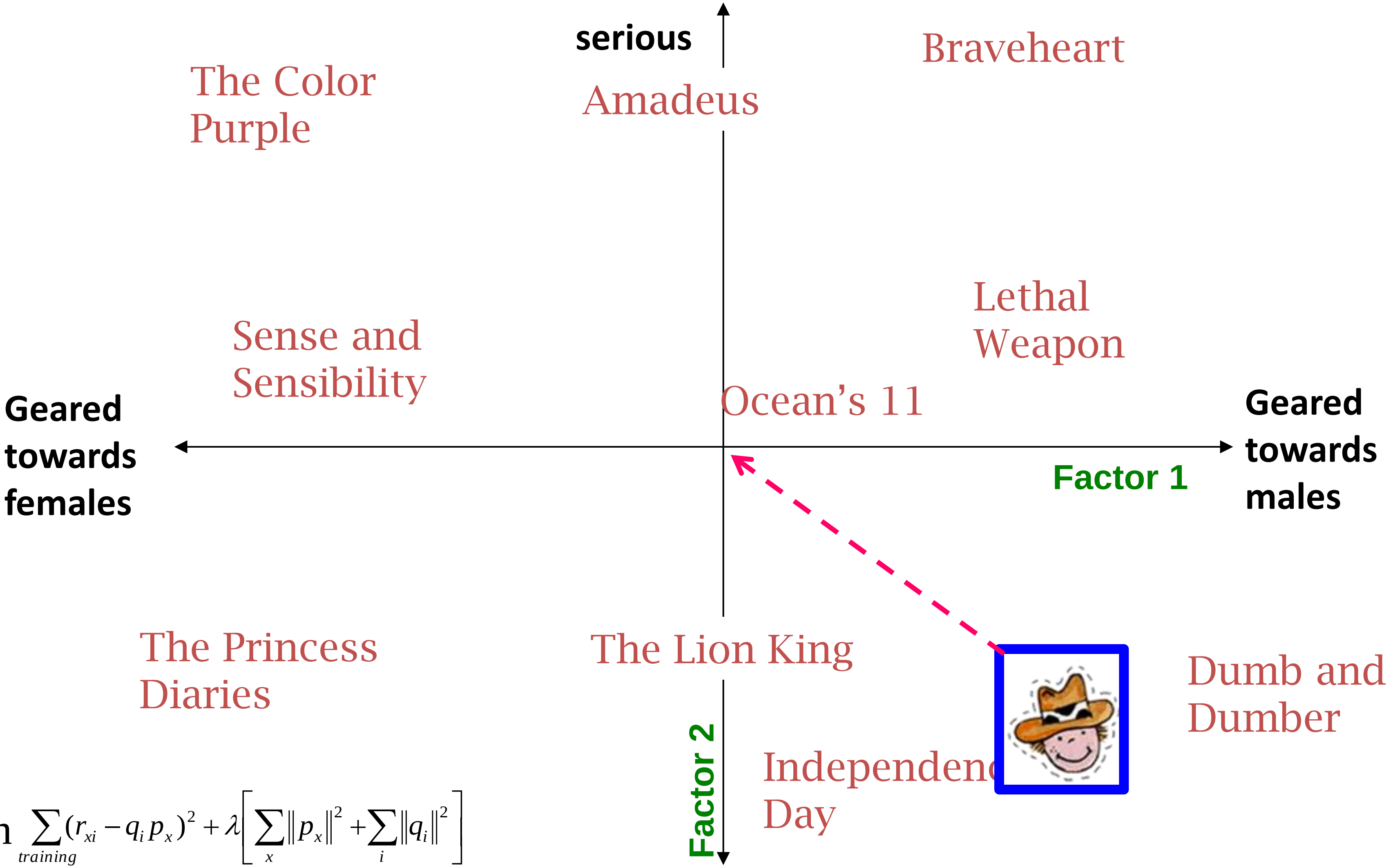
The Effect of Regularization



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

$\min_{factors}$ "error" + λ "length"

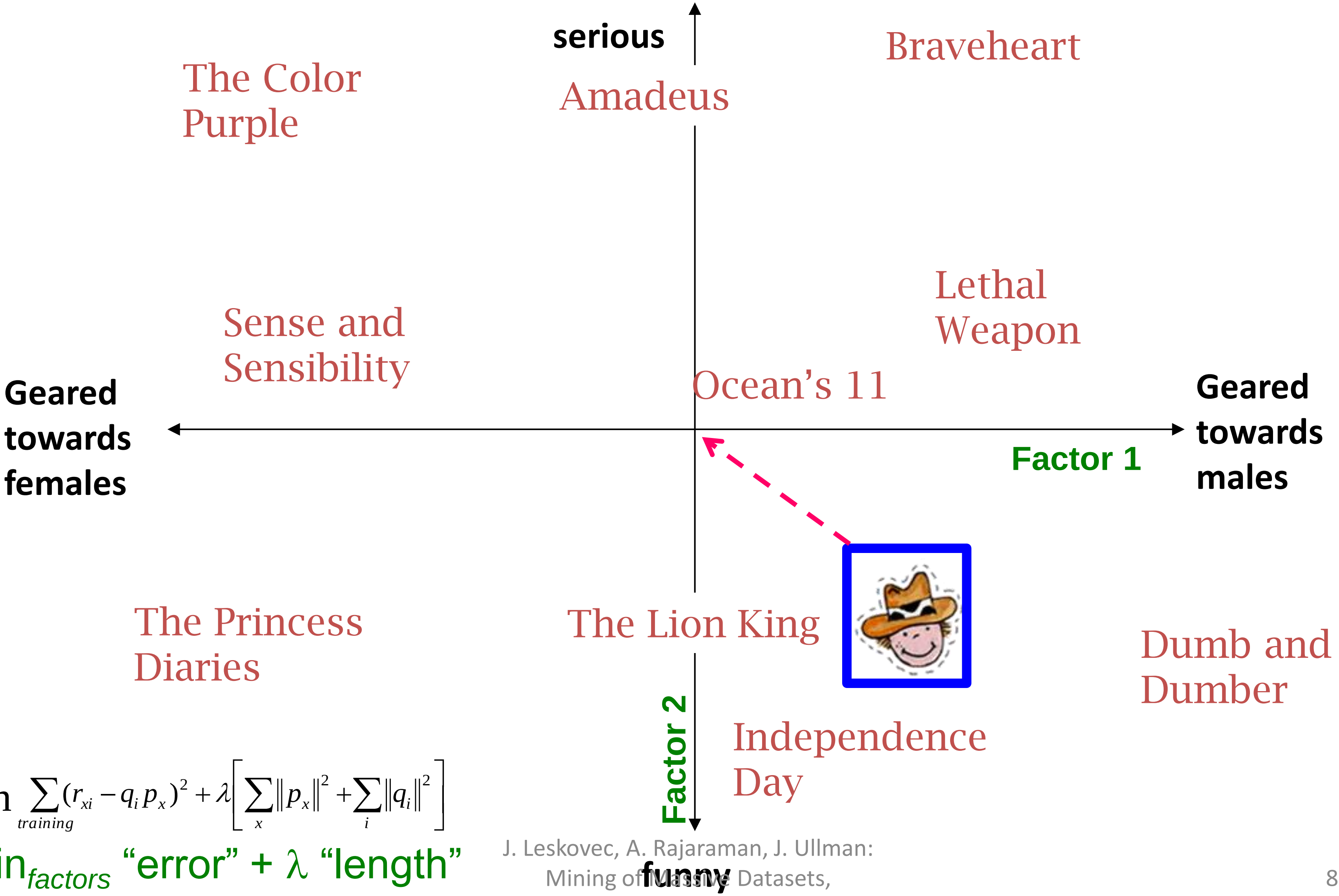
The Effect of Regularization



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

$\min_{factors}$ "error" + λ "length"

The Effect of Regularization



serious

The Color Purple

Amadeus

Braveheart

Sense and Sensibility

Lethal Weapon

Ocean's 11

Geared towards females

Geared towards males

Factor 1

The Princess Diaries

The Lion King

Dumb and Dumber

Independence Day

Factor 2

funny

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

$\min_{factors}$ "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

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%

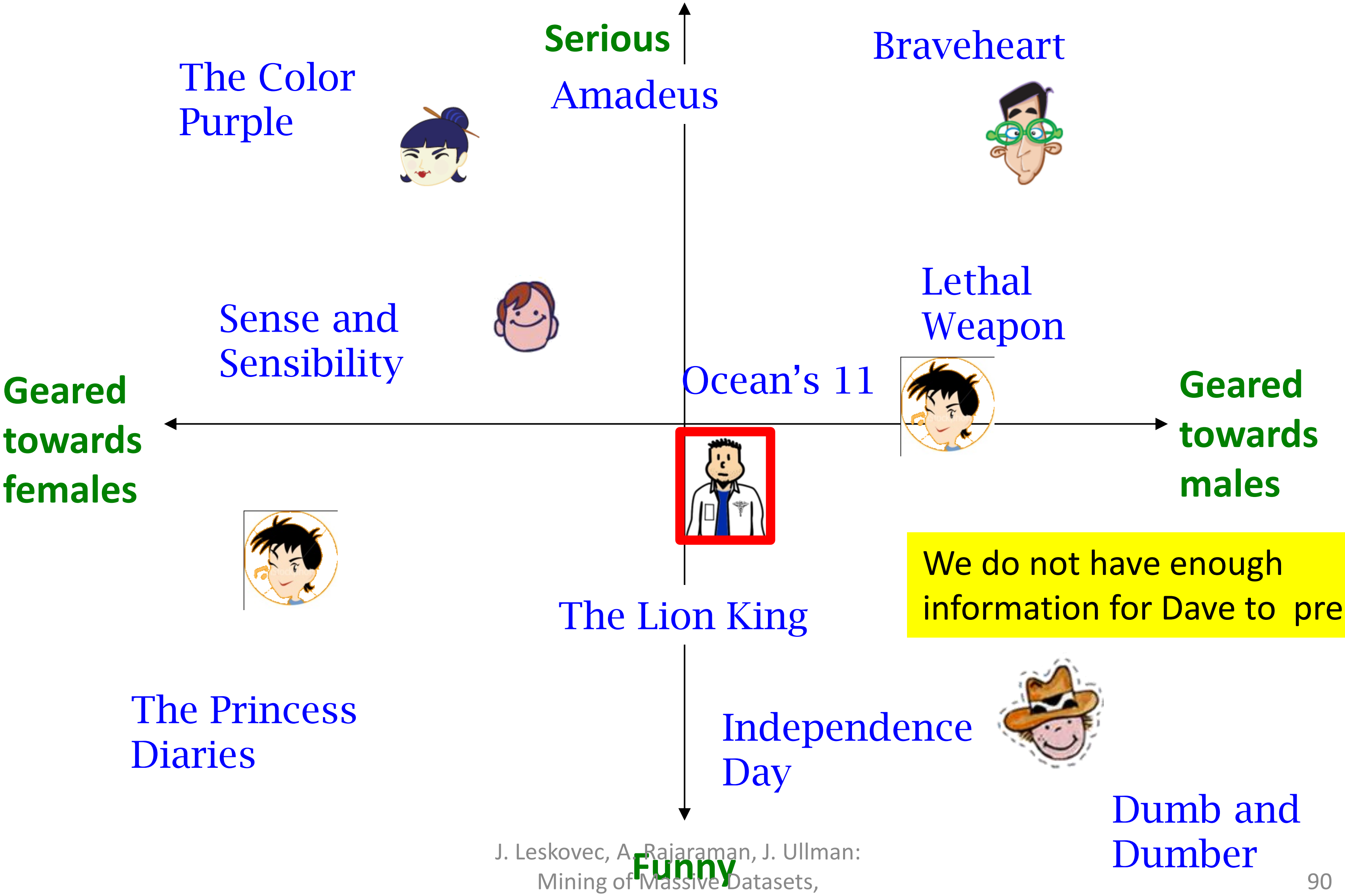
Extensions

- Social Recommendations

Social Recommendations

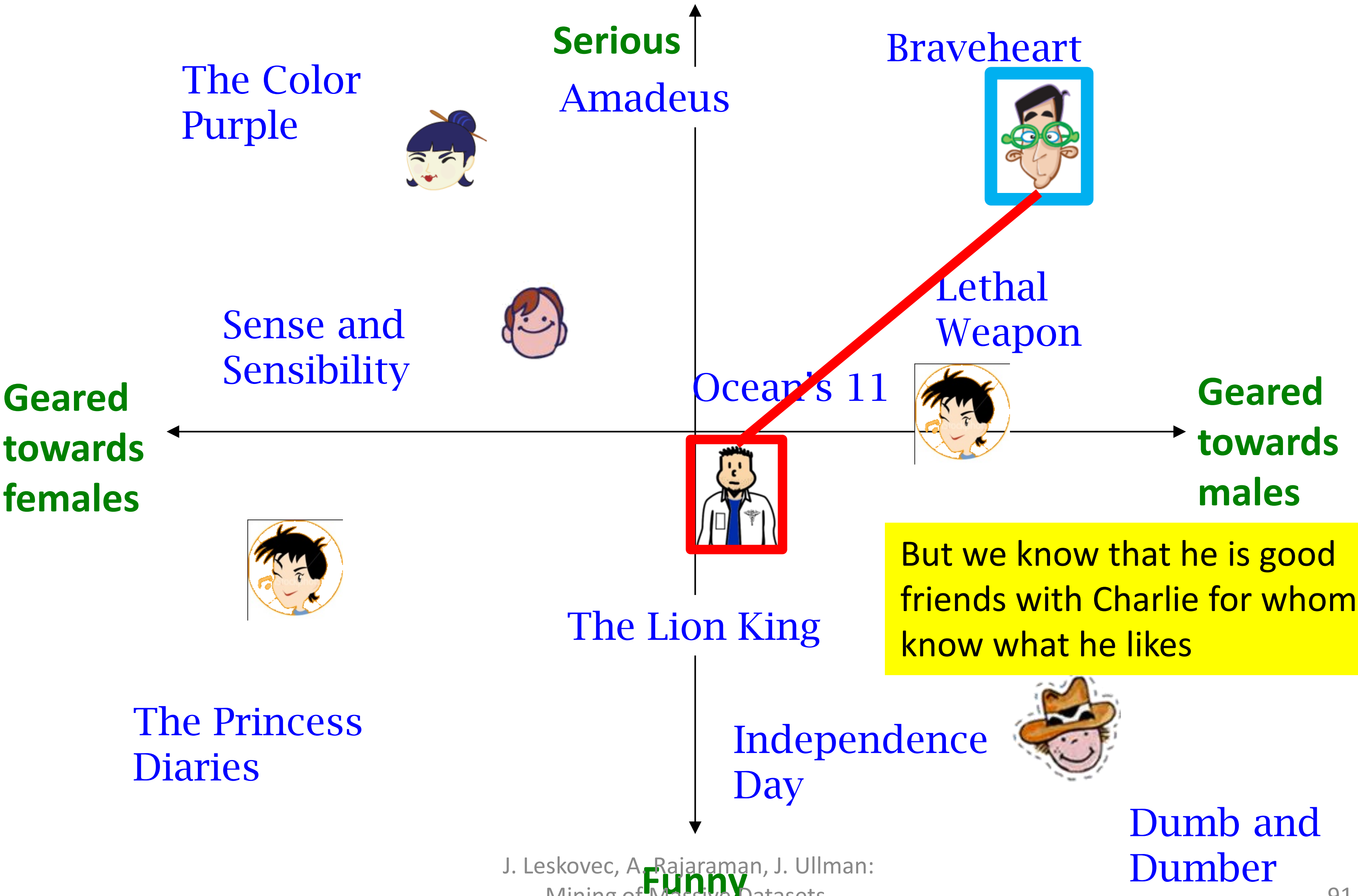
- Suppose that except for the rating matrix, you are also given a social network of all the users
  - Such social networks exist in sites like Yelp, Foursquare, Epinions, etc
- How can we use the social network to improve recommendations?
  - Homophily: connected users are likely to have similar ratings.

Social Recommendations



We do not have enough information for Dave to predict

Social Recommendations



Serious

Funny

Geared towards females

Geared towards males

The Color Purple

Amadeus

Braveheart

Sense and Sensibility

Lethal Weapon

Ocean's 11

The Lion King

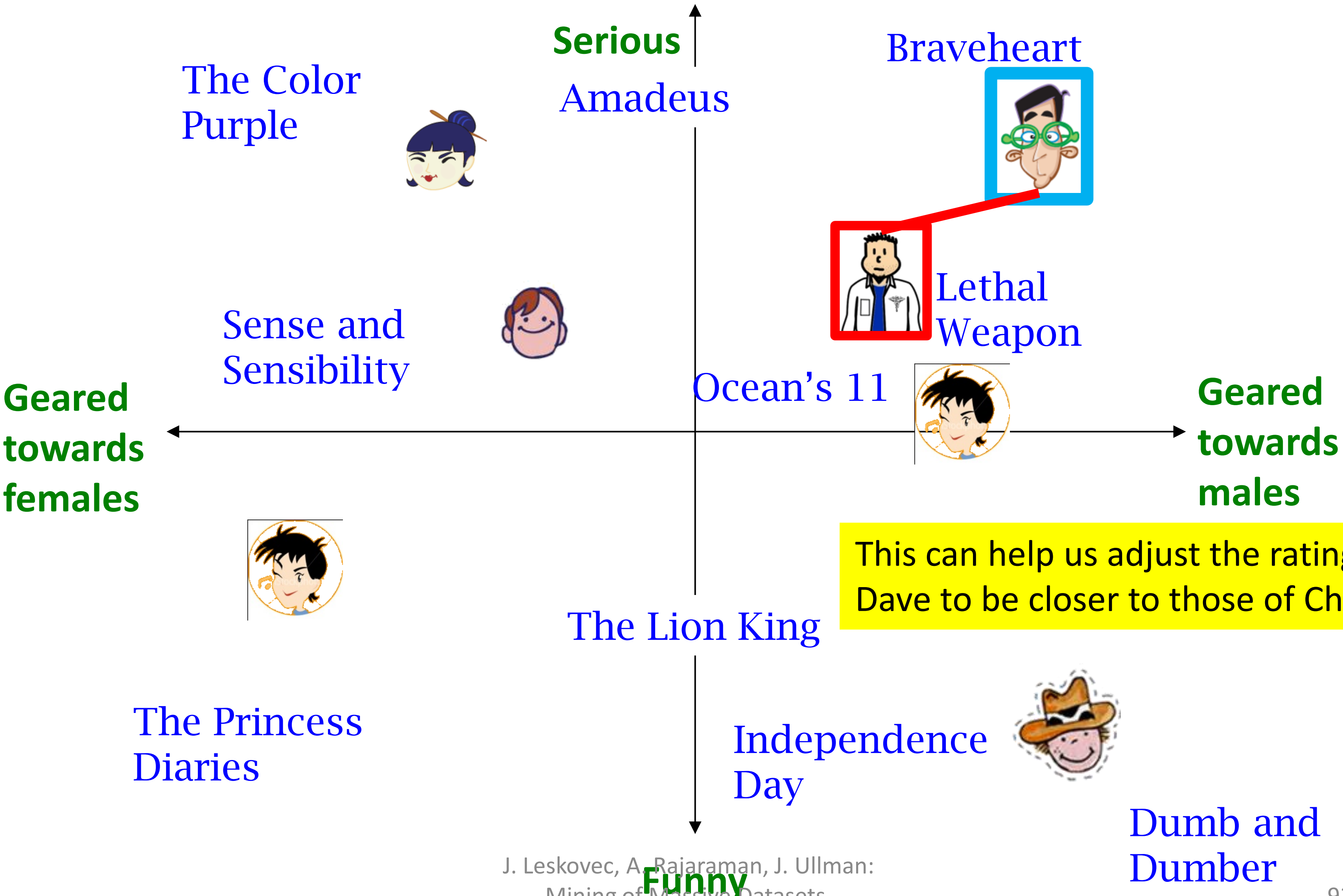
The Princess Diaries

Independence Day

Dumb and Dumber

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

Social Recommendations



Serious

Braveheart

The Color Purple

Amadeus

Lethal Weapon

Sense and Sensibility

Ocean's 11

Geared towards females

Geared towards males

The Lion King

The Princess Diaries

Independence Day

Dumb and Dumber

Funny

This can help us adjust the ratings for Dave to be closer to those of Charlie

Social Regularization

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

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

  - w_{xy}: strength of the connection between x and y
  - $\left\|p_x-p_y\right\|$: 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.

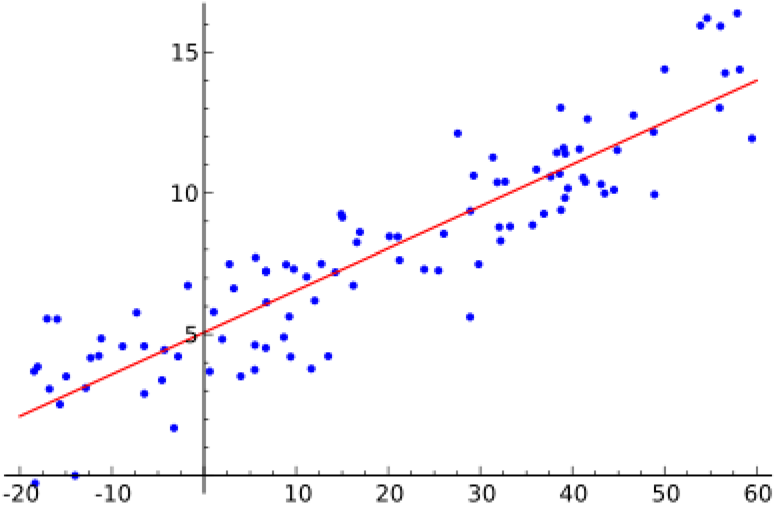
- The same idea can be applied in different settings

Example: Linear regression

- Regression: Create a model to predict a continuous value.
- Given a dataset of the form $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ 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:

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

This is sometimes also called network smoothing

- This can be written as:

$$\sum_i (w^T x_i - y_i)^2 + \underbrace{\alpha w^T X L_A X^T w}_{L_A\text{: The Laplacian of the adjacency matrix}}$$

- 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:

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

- 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