universidade de aveiro

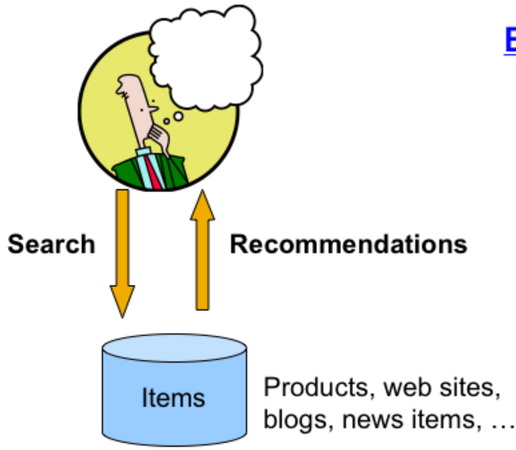
Mining Large Scale Datasets

Recommender Systems

(Adapted from CS246@Starford.edu; http://www.mmds.org)

Sérgio Matos - aleixomatos@ua.pt

Recommendations



Examples:

Scarcity vs Abundance

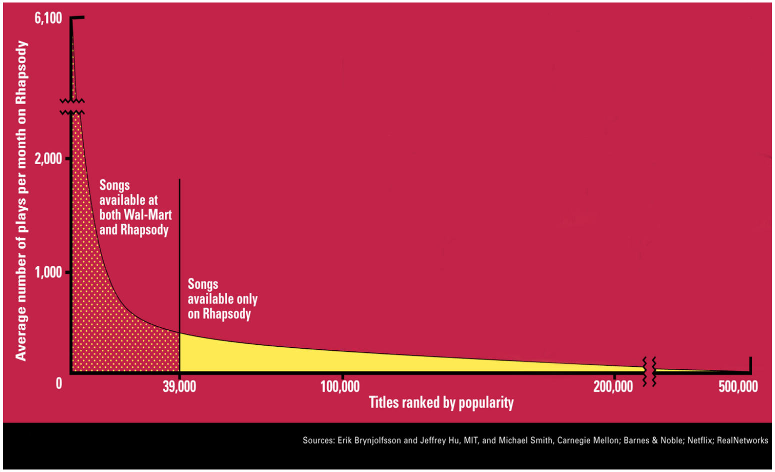
- Shelf space in traditional stores is scarce (and expensive)
  - Also: TV schedule, movie theaters, newspaper pages, ...

- Web enables near-zero-cost dissemination of information about products

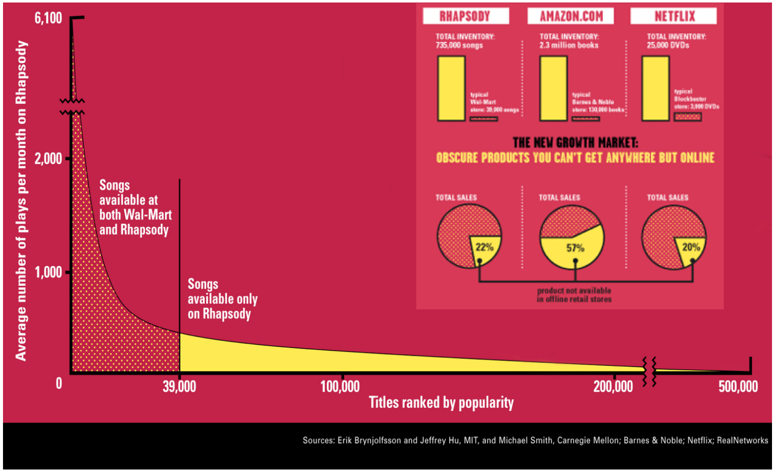
  $\hookrightarrow$ Abundance

$\Rightarrow$ More choice necessitates better filters
  - Recommendation engines
  - Association rules:
    How *Into Thin Air* made *Touching the Void* a bestseller

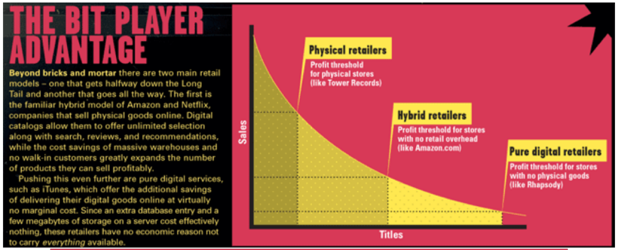
The long tail



The long tail



Physical vs Online



Types of recommendations

- Editorial and hand curated
  - List of favorites
  - Lists of “essential” items
- Simple aggregates
  - Top 10
  - Most Popular
  - Recent Uploads

⇒ **Tailored to individual users**

  - Amazon, Netflix, ...

Formal model

- $\mathbf{X}$ = set of Customers
- $\mathbf{S}$ = set of Items

- Utility function $u : X \times S \to R$
  - R = set of ratings
  - R is a totally ordered set
  - e.g., 0-5 stars, real number in [0,1]

Utility matrix

	Avatar	LotR	Matrix	PotC
Alice	1		0.2	
Bob		0.5		0.3
Carol	0.2		1	
David				0.4

Key problems

(1) Gathering "known" ratings for matrix
- How to collect the data in the utility matrix

(2) Extrapolate unknown ratings from the known ones
- Mainly interested in high unknown ratings
- Interested in knowing what users like, not what they don't like

(3) Evaluating extrapolation methods
- How to measure success/performance of recommendation methods

Gathering ratings

- Explicit
  - Ask people to rate items
  - Doesn't work well in practice – most people won't be bothered; biased to those willing to rate
  - Crowdsourcing: Pay people to label items

- Implicit
  - Learn ratings from user actions
    E.g., purchase / watching implies high rating
  - What about low ratings?

Extrapolating ratings

- Key problem: Utility matrix **U is sparse**
  - Most people have not rated most items
  - Cold start
    - New items have no ratings
    - New users have no history
- Three approaches to recommender systems
  - Content-based
  - Collaborative
  - Latent factor based

Content-based recommendation

- **Main idea**

  Recommend to customer x items similar to previous items rated highly by x

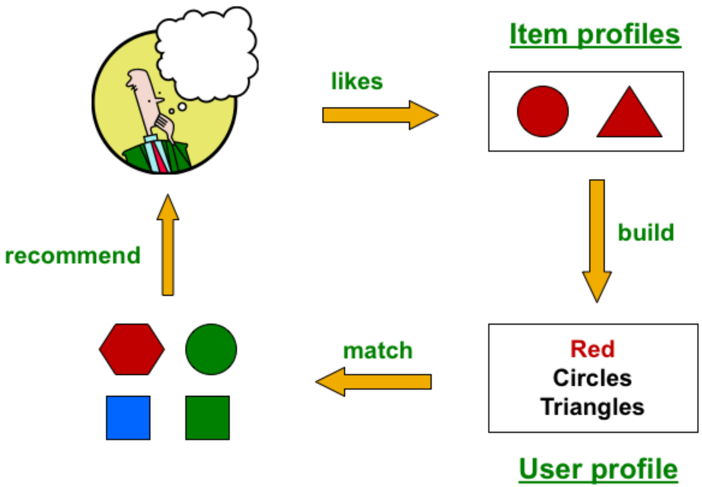
- Movie recommendations

  Recommend movies with same actor(s), director, genre, ...

- Websites, blogs, news

  Recommend other sites with “similar” content

Overview



Item profiles
likes
build
recommend
match
Red
Circles
Triangles
User profile

Item profiles

- Create an **item profile** for each item
  - A set (vector) of features
  - Movies: author, title, actor, director, ...
  - Text: Set of "important" words in document
- How to pick important features?
  - Usual heuristic from text mining is TF-IDF
    (Term frequency * Inverse Doc Frequency)
  - Doc profile = set of words with highest TF-IDF scores

User profiles and prediction

- User profile possibilities
  - Weighted average of rated item profiles
  - Variation: weight by difference from average rating for item

- Prediction heuristic: Cosine similarity of user and item profiles

  Given user profile x and item profile i, estimate

  $$u(\mathbf{x}, \mathbf{i}) = cos(\mathbf{x}, \mathbf{i}) = \frac{\mathbf{x} \cdot \mathbf{i}}{\|\mathbf{x}\| \cdot \|\mathbf{i}\|}$$

- How do you quickly find items closest to $\mathbf{x}$?

User profiles and prediction

- User profile possibilities
  - Weighted average of rated item profiles
  - Variation: weight by difference from average rating for item

- Prediction heuristic: Cosine similarity of user and item profiles

  Given user profile x and item profile i, estimate

  $$u(\mathbf{x}, \mathbf{i}) = cos(\mathbf{x}, \mathbf{i}) = \frac{\mathbf{x} \cdot \mathbf{i}}{\|\mathbf{x}\| \cdot \|\mathbf{i}\|}$$

- How do you quickly find items closest to $\mathbf{x}$?

  $\hookrightarrow$ LSH!

Content-based: Pros

+ No need for data on other users
+ Able to recommend to users with unique tastes
+ Able to recommend new and unpopular items
  - No first-rater problem
+ Able to provide explanations
  - Explain recommended items by listing content-features that caused items to be recommended

Content-based: Cons

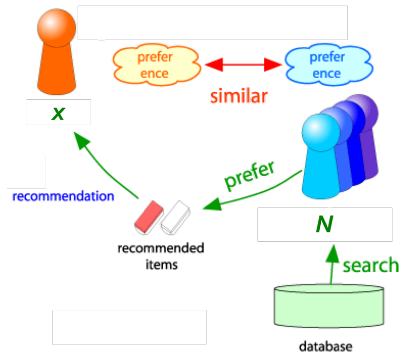
- Finding the appropriate features is hard
  - E.g., images, movies, music

- Recommendations for new users
  - How to build a user profile?

- Overspecialization
  - Never recommends items outside user's content profile
  - People might have multiple interests
  - Unable to exploit quality judgments of other users

Collaborative filtering

- Consider user x
- Find set N of other users whose ratings are "similar" to x's ratings
- Estimate x's ratings based on ratings of the N users



Finding "similar" users: Similarity metric

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- Intuitively we want $sim(A, B) > sim(A, C)$

Finding "similar" users: Similarity metric

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- Jaccard similarity

  $sim(A, B) = 1/5 < 2/4 = sim(A, C)$

- Problem: Ignores the values of ratings

Finding "similar" users: Similarity metric

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- Cosine similarity $\quad sim(\mathbf{x}, \mathbf{y}) = cos(\mathbf{r_x}, \mathbf{r_y}) = \frac{\mathbf{r_x} \cdot \mathbf{r_y}}{\|\mathbf{r_x}\| \cdot \|\mathbf{r_y}\|}$

  $sim(A, B) = 0.380 > 0.322 = sim(A, C)$

- Problem: Treats missing ratings as "negative" (disliked)

  $r_A = 4, 0, 0, 5, 1, 0, 0, \; r_B = 5, 5, 4, 0, 0, 0, 0$

Finding "similar" users: Similarity metric

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	$2/3$			$5/3$	$-7/3$		
B	$1/3$	$1/3$	$-2/3$				
C				$-5/3$	$1/3$	$4/3$	
D		0					0

- Cosine similarity $\quad sim(\mathbf{x}, \mathbf{y}) = cos(\mathbf{r_x}, \mathbf{r_y}) = \frac{\mathbf{r_x} \cdot \mathbf{r_y}}{\|\mathbf{r_x}\| \cdot \|\mathbf{r_y}\|}$

  $sim(A, B) = 0.380 > 0.322 = sim(A, C)$

- Problem: Treats missing ratings as "negative" (disliked)
- **Solution: subtract the (row) mean**
  - $=$ Pearson correlation coefficient

Finding "similar" users: Similarity metric

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- Pearson correlation coefficient
  - S_{xy} = items rated by both users x and y

$$sim(\mathbf{x}, \mathbf{y}) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \bar{\mathbf{r}}_{\mathbf{x}})(r_{ys} - \bar{\mathbf{r}}_{\mathbf{y}})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \bar{\mathbf{r}}_{\mathbf{x}})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \bar{\mathbf{r}}_{\mathbf{y}})^2}}$$

$$sim(A, B) = 0.092 > -0.559 = sim(A, C)$$

Predicting ratings

From similarity metric to recommendations

- Let $\mathbf{r_x}$ be the vector of ratings for user x
- Let N be the set of k users most similar to x who have rated item i
- Prediction for item i of user x:

  $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$

  or even better,

  $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$, $s_{xy} = sim(x, y)$

Item-Item Collaborative Filtering

- Item-item vs User-user

- For item i, find other similar items
- Estimate rating for item i based on ratings for similar items
- Can use same similarity metrics and prediction functions as in user-user model

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

s_{ij}: similarity of items i and j
r_{xj}: rating of user x on item j
$N(i;x)$: set items rated by x that are similar to i

Item-Item CF

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3			5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

- unknown rating

- rating between 1 to 5

Item-Item CF

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

- estimate rating of movie **1** by user **5**

Item-Item CF

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
<u>3</u>	2	4		1	2		3		4	3	5		<u>0.41</u>
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
<u>6</u>	1		3		3			2			4		<u>0.59</u>

Neighbor selection:
Identify movies similar to movie **1**, **rated by user 5**

Here we use Pearson correlation as similarity:

1) Subtract mean rating m_i from each movie i

$m_1 = (1+3+5+5+4)/5 = 3.6$

row 1: *[-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]*

2) Compute cosine similarities between rows

Item-Item CF

users

movies	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
<u>3</u>	2	4		1	2		3		4	3	5		<u>0.41</u>
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
<u>6</u>	1		3		3			2			4		<u>0.59</u>

Compute similarity weights:
$s_{1,3}$=0.41, $s_{1,6}$=0.59

Item-Item CF

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		2.6	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Predict by taking weighted average:

$r_{1.5}$ = **(0.41*2 + 0.59*3) / (0.41+0.59) = 2.6**

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$

Item-item vs User-user

	Avatar	LotR	Matrix	PotC
Alice	1		0.2	
Bob		0.5		0.3
Carol	0.2		1	
David				0.4

- In theory, these are dual approaches with similar performance
- In practice, it has been observed that item-item often works better than user-user
- Why? Items are simpler, users have multiple tastes

Pros/Cons of Collaborative Filtering

\+ Works for any kind of item
  - No feature selection needed

\- Cold Start
  - Need enough users in the system to find a match

\- Sparsity
  - The user/ratings matrix is sparse
  - Hard to find users that have rated the same items

\- First rater
  - Cannot recommend an item that has not been previously rated
  - New items, esoteric items

\- Popularity bias
  - Cannot recommend items to someone with unique taste
  - Tends to recommend popular items

Hybrid methods

Combine predictions from two or more different recommenders

- e.g. Global baseline + CF
- Perhaps using a linear model

Add content-based methods to collaborative filtering

- Item profiles for new item problem
- Demographics to deal with new user problem

CF: Common practice

- Define similarity s_{ij} of items i and j
- Select k nearest neighbors $N(i;x)$
  - Items most similar to i, that were rated by x
- Estimate rating r_{xi} as the weighted average

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

$b_{xi} = \mu + b_x + b_i$ baseline estimate for r_{xi}

$\mu =$ overall mean movie rating
$b_x =$ rating deviation of user $x =$ (avg rating of user x) - μ
$b_i =$ rating deviation of movie $i =$ (avg rating of movie i) - μ

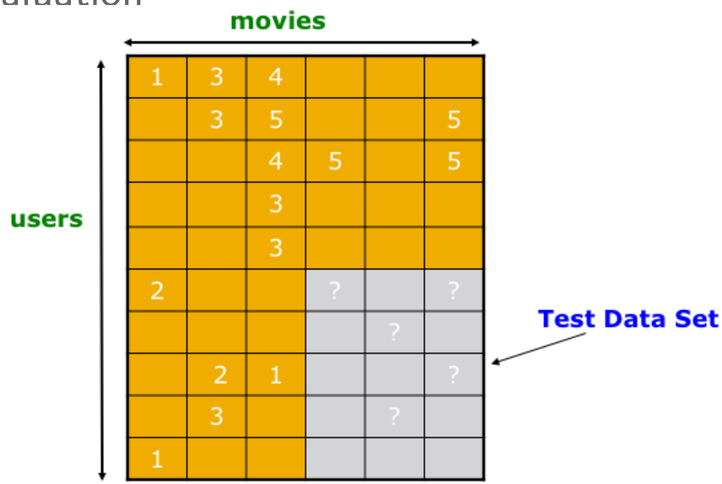
Evaluation

movies

users

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

Evaluation



Evaluating predictions

- Compare predictions with known ratings
  - Root-mean-square error (RMSE)

    $\sqrt{\frac{\sum_{xi}(r_{xi}-r^*_{xi})^2}{\sum_{xi} 1}}$ where r_{xi} is predicted; r^*_{xi} is the true rating
  - Precision at top 10
  - Rank correlation
    Spearman's correlation between system's and user's complete rankings

- Another approach: 0/1 model (dislike/like)
  - Coverage

    # items/users for which the system can make predictions
  - Precision
  - Receiver operating characteristic (ROC)

    Tradeoff curve between false positives and false negatives

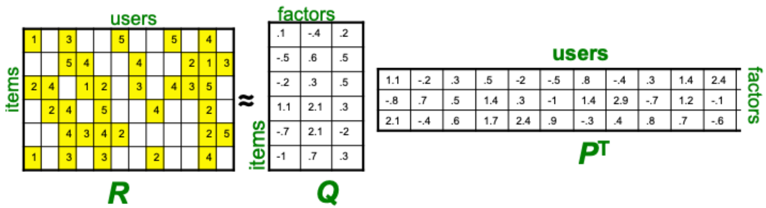
Problems with error measures

- Narrow focus on accuracy sometimes misses the point
    - Prediction diversity
    - Prediction context
    - Order of predictions
- In practice, we care only about predicting high ratings
    - RMSE might penalize a method that does well for high ratings and badly for others

Collaborative Filtering: Complexity

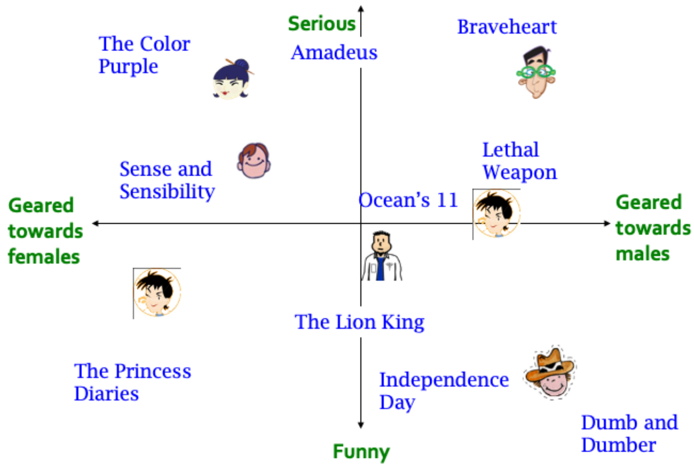
- Expensive step is finding k most similar customers: $O(|X|)$
- Too expensive to do at runtime
  - Could pre-compute
  - Naïve pre-computation takes time $O(k \cdot |X|)$
- We already know how to do this!
  - Near-neighbor search in high dimensions (LSH)
  - Clustering
  - Dimensionality reduction (PCA, SVD)

Latent factor models

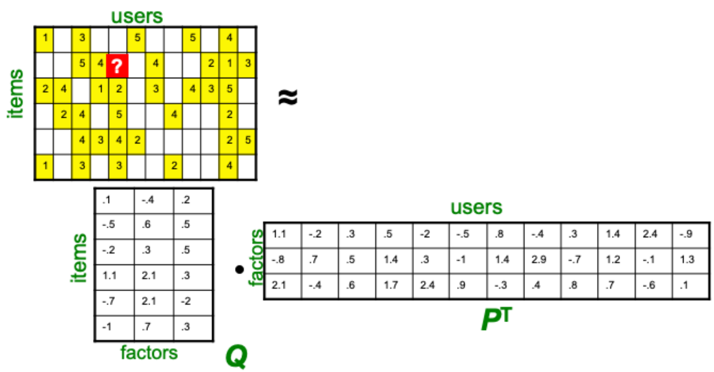


Assume we can approximate the rating matrix R as a product of "thin" $Q \cdot P^T$

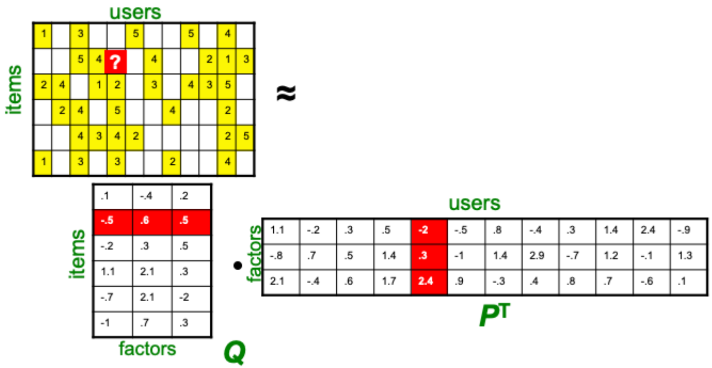
Latent factor models



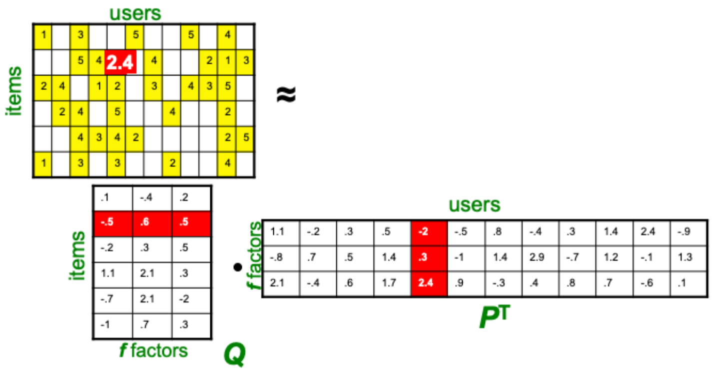
Latent factor models



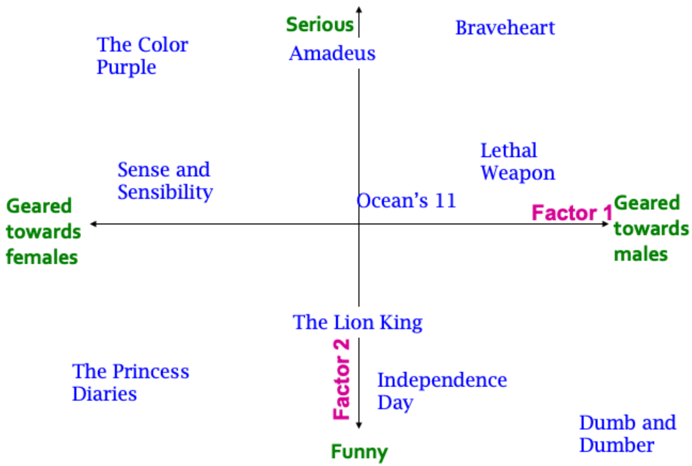
Latent factor models



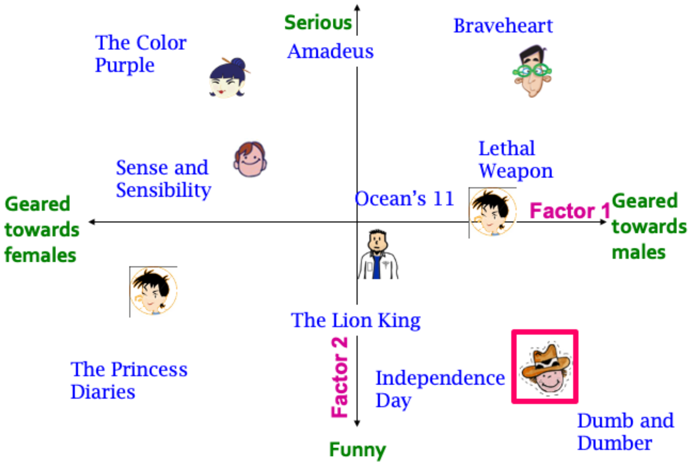
Latent factor models



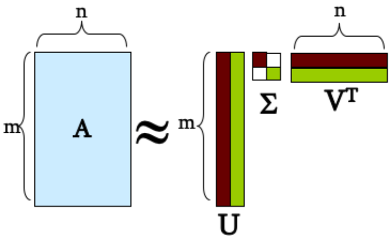
Latent factor models



Latent factor models



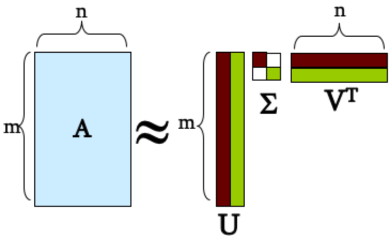
Latent factor models: SVD



- A: Input matrix
- U: Left singular matrix
- V: Right singular matrix
- Σ: Singular values

Latent factors model: $R \approx Q \cdot P^T$
As SVD: $A = R,\ Q = U,\ P^T = \Sigma V^T$

Latent factor models: SVD

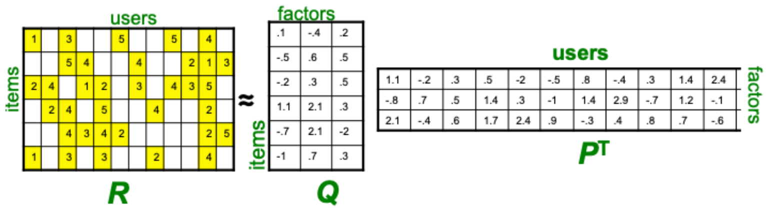


- A: Input matrix
- U: Left singular matrix
- V: Right singular matrix
- Σ: Singular values

Latent factors model: $R \approx Q \cdot P^T$
As SVD: $A = R,\ Q = U,\ P^T = \Sigma V^T$

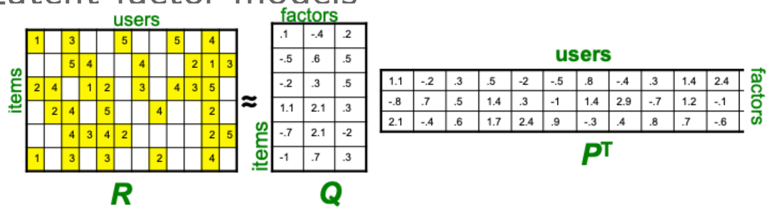
$\hookrightarrow$ SVD minimizes SSE, and thus RMSE!

Latent factor models: SVD



- ✗ SVD is not defined when entries are missing!
- Need a special method to find P, Q

Latent factor models

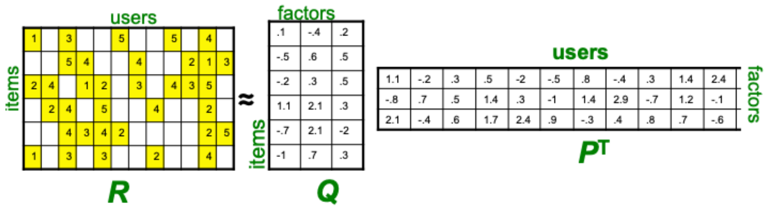


- ✗ SVD is not defined when entries are missing!
- Need a special method to find P, Q

  $\min_{P,Q} \sum_{(i,x)\in R} (r_{xi} - q_i \cdot p_x)^2$

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

Latent factor models



Goal is to find P, Q such that $\min_{P,Q} \sum_{(i,x)\in R} (r_{xi} - q_i \cdot p_x)^2$

Want to minimize SSE for unseen test data

- Approach: Minimize SSE on training data
- Want large k (number of factors) to capture all the signals
- But SSE on test data begins to rise for $k > 2$
  $\hookrightarrow$ **Overfitting**

Latent factor models

Solution: **Regularization**

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

$$\min_{P,Q} \sum_{(i,x)\in R} (r_{xi} - q_i \cdot p_x)^2 + \left[\lambda_1 \sum_x \|p_x\|^2 + \lambda_2 \sum_i \|q_i\|^2\right]$$

Latent factor models

Combining with baseline predictor

$r_{xi} = \mu + b_x + b_i + q_i \cdot p_x$

- μ: overall mean rating
- b_x: bias of user x
- b_i: bias of movie i

$$\min_{P,Q} \sum_{(i,x)\in R} \left(r_{xi} - (\mu + b_x + b_i + q_i \cdot p_x)\right)^2 +$$
$$\left[\lambda_1 \sum_x \|p_x\|^2 + \lambda_2 \sum_i \|q_i\|^2 + \lambda_3 \sum_x \|b_x\|^2 + \lambda_4 \sum_i \|b_i\|^2\right]$$

Latent factor models

Combining with baseline predictor

$r_{xi} = \mu + b_x + b_i + q_i \cdot p_x$

- μ: overall mean rating
- b_x: bias of user x
- b_i: bias of movie i

$$\min_{P,Q} \sum_{(i,x)\in R} \left(r_{xi} - (\mu + b_x + b_i + q_i \cdot p_x)\right)^2 + \left[\lambda_1 \sum_x \|p_x\|^2 + \lambda_2 \sum_i \|q_i\|^2 + \lambda_3 \sum_x \|b_x\|^2 + \lambda_4 \sum_i \|b_i\|^2\right]$$

$\hookrightarrow$ **Stochastic Gradient Descent**

Final tip: Add data

- Leverage all the data
  - Don't try to reduce data size in an effort to make fancy algorithms work
  - Simple methods on large data do best
- Add more data
  - e.g., add IMDB data on genres
- **~~More~~ Richer data beats better algorithms**