

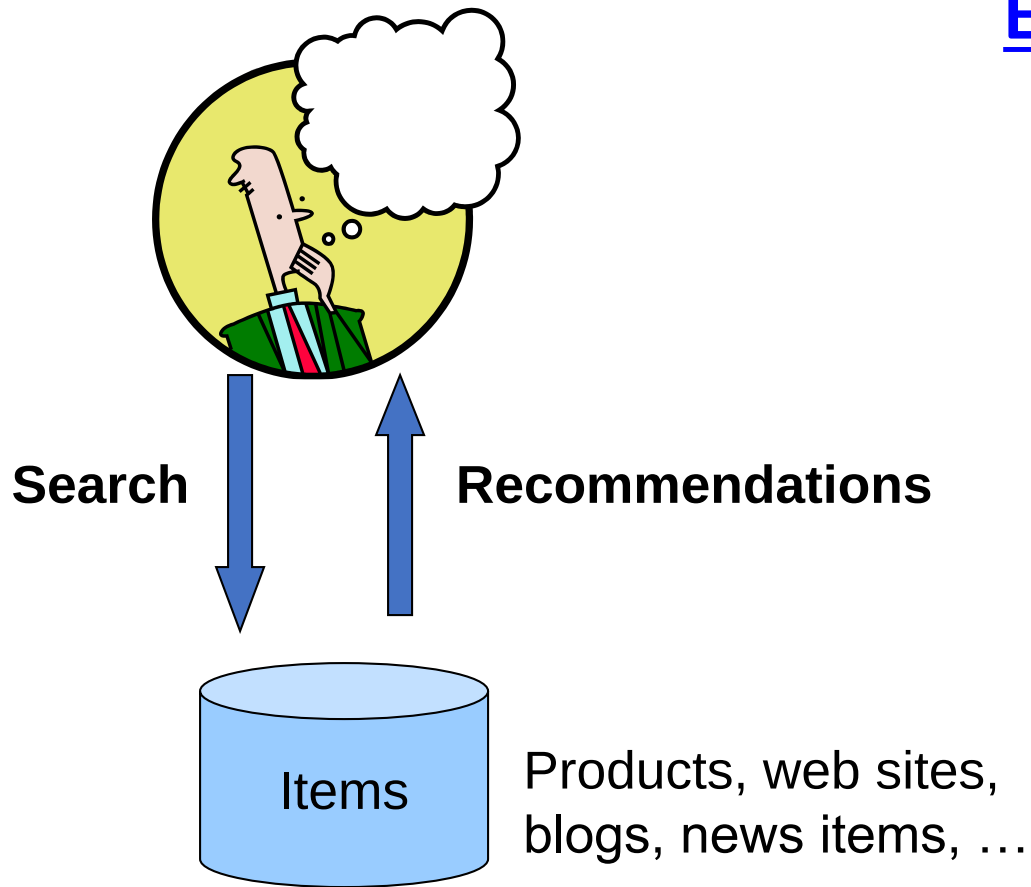


Recommender systems

Content-based, collaborative-based, hybrid methods

Web Search

Recommendations



Examples:

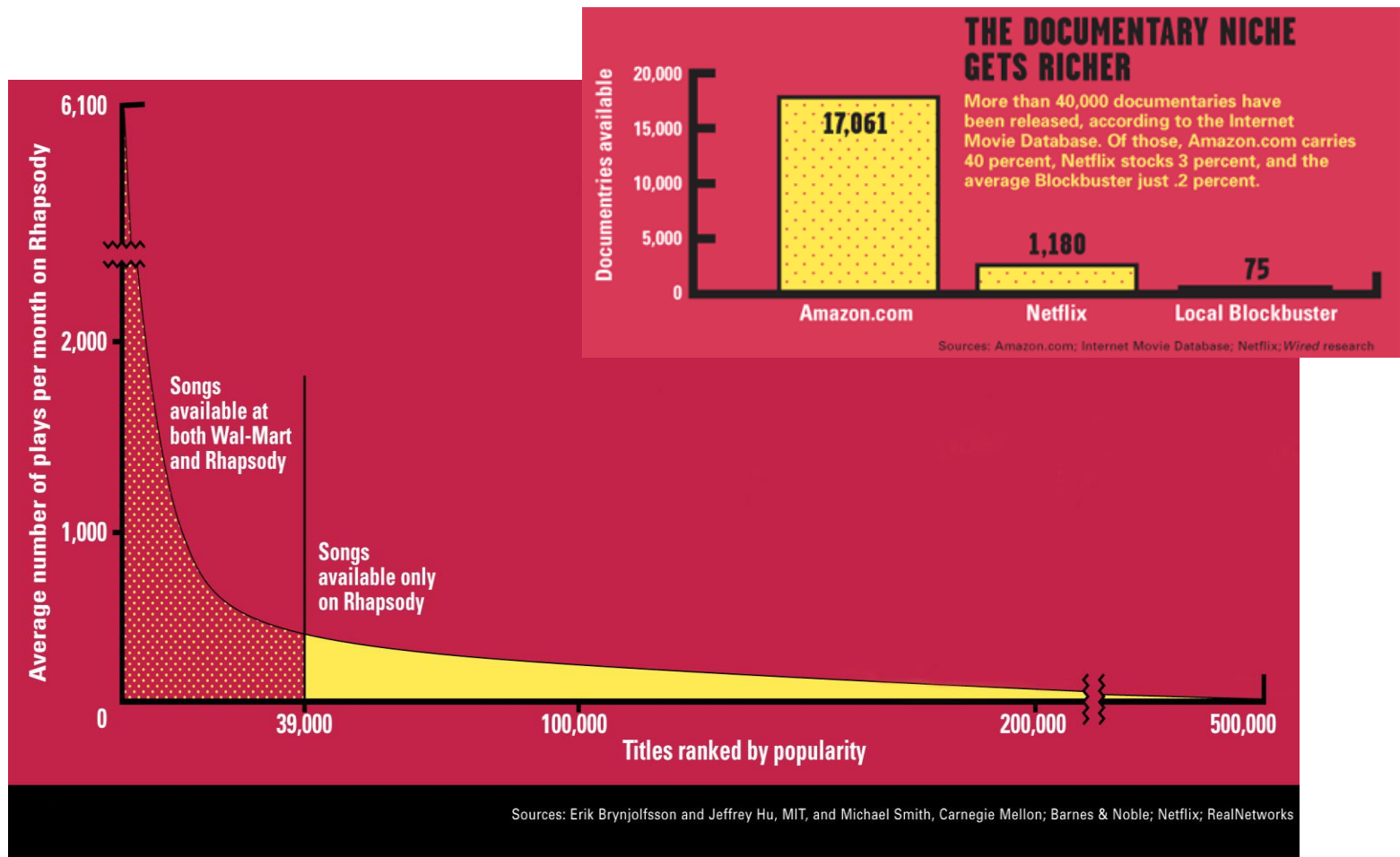
amazon.com.



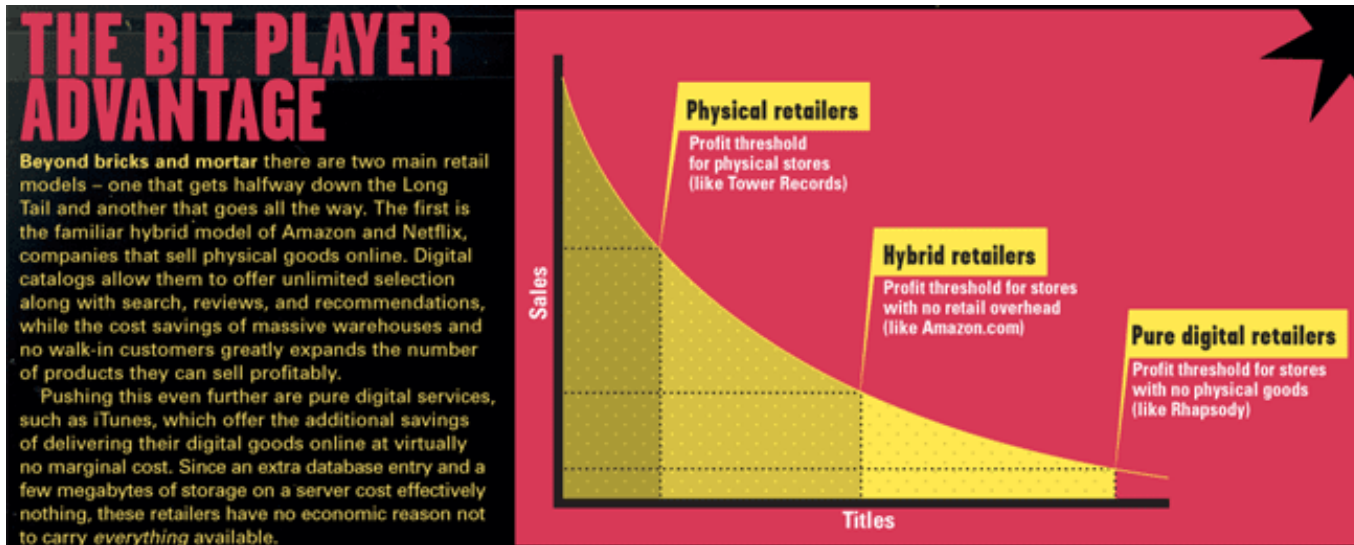
m o v i e l e n s
helping you find the *right* movies



Sidenote: The Long Tail



Physical vs. Online



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

Recommender systems

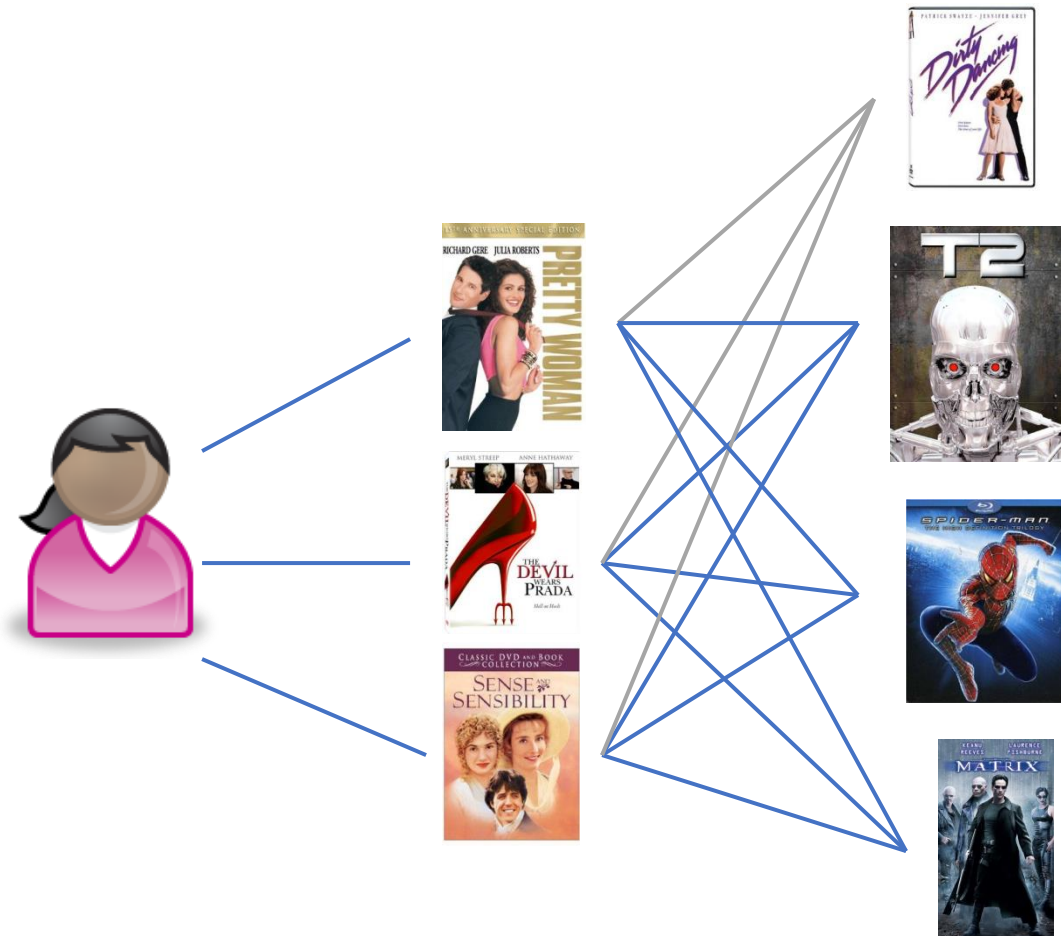
- Recommender systems aim at suggesting new products to users based on their preferences
- Recommendations can be computed from two different type of inputs:
 - Product characteristics
 - Collective user ratings



Recommender systems

- Content-based recommendations
- Collaborative filtering
 - Neighborhood methods
 - Matrix factorization methods
- Hybrid methods

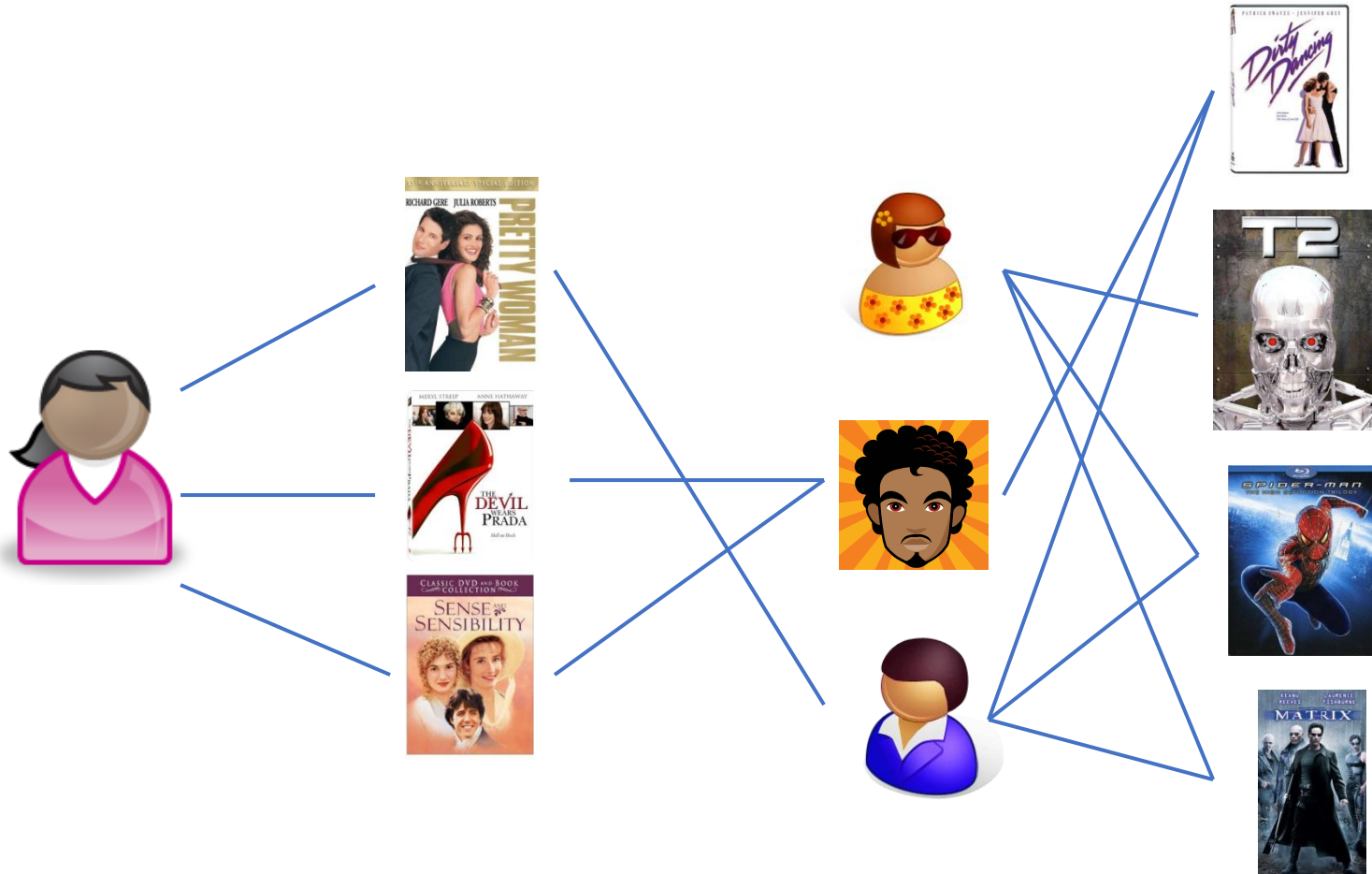
Content-based recommendations



Content-based recommendations

- Users who enjoyed a product because of its characteristics, will most likely appreciate other products with related characteristics
- The recommendation will be the set of products most similar to the consumed products
 - A similarity between a user consumed products and all other products is computed
 - The similarity is computed as a distance in the space of product characteristics
 - This is equivalent to the vector space discussed previously
- This approach requires a knowledge-base of product characteristics

Collaborative filtering



Collaborative filtering

- This family of methods explore information provided by a large number of users about a large number of products
 - Usually the so-called product ratings
- Data about co-rated product items allows us to explore co-occurrences
 - Co-occurrences can be explored in a vector space text retrieval
 - Co-occurrences matrices can also be factorized into a simpler model
- Collaborative filtering is based in the notion of product-user ratings matrix

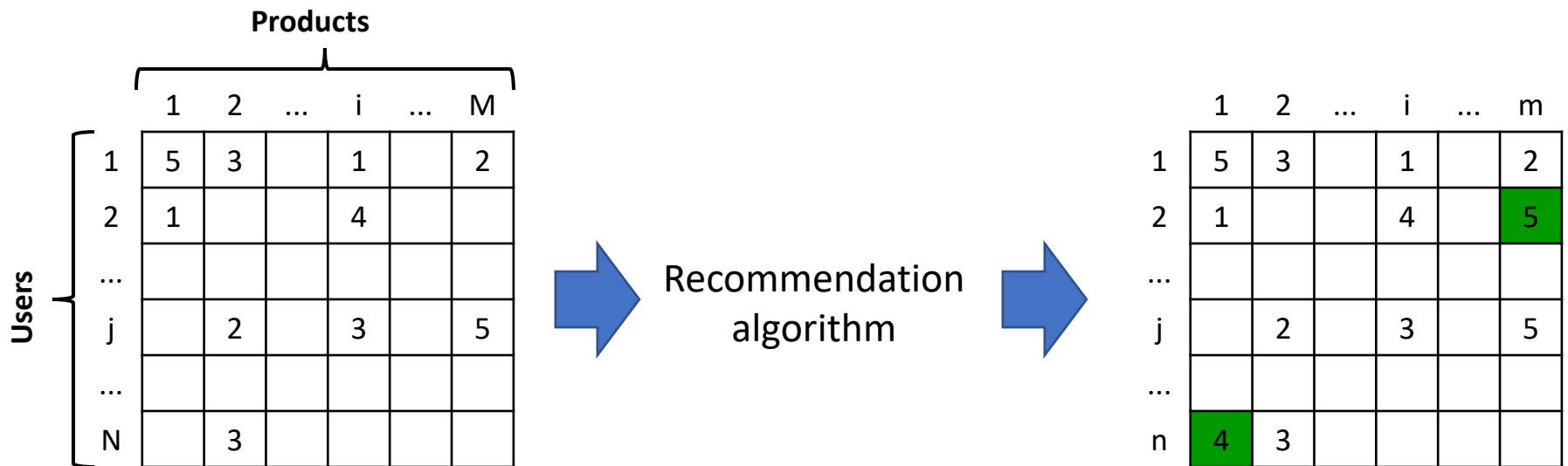
Ratings matrix

- Consider a set of M products and a set of N users
- Users indicate their preference for each product with a rating of 1 (don't like) to 5 (like)
- The matrix R collects the ratings of all users about all products
 - It is highly incomplete (sparse) because most users have only rated a small portion of all products

		Products					
		1	2	...	i	...	M
Users	1	5	3		1		2
	2	1			4		
	...						
	j		2		3		5
	...						
	N		3				

Objective

The goal is to mine the relations between products and users, and predict the most likely preferences of users



Neighborhood methods

- In neighbourhood methods, a subset of users are chosen to compute recommendations for a particular user
- This is based in the k-nearest-neighbour (k-nn) algorithm:
 - Compute the distance between the current user and all other users
 - Select the k users that have the highest similarity to the current user
 - Compute the prediction vector of all products from a weighted combination of selected neighbours' ratings.

Similarity among users

- Given a matrix of ratings
 - The similarity between user a and user u can be computed as the Pearson correlation coefficient:

$$w_{a,u} = \frac{\sum_{i \in I} (r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i \in I} (r_{a,i} - \bar{r}_a)^2} \sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2}}$$

		Products					
		1	2	...	i	...	M
Users	1	5	3		1		2
	2	1			4		
	...						
	j		2		3		5
	...						
	N		3				

- The resulting vector is the relation between user a and all other N users:

	1	2	...	i	...	N
a	$w_{a,1}$			$w_{a,i}$		$w_{a,N}$

Users neighborhood weighting matrix

- The neighborhood weighting matrix is computed as the similarity across all users

		Users			
		1	2	...	N
Users	1	1	$w_{1,2}$	...	$w_{1,N}$
	2	$w_{2,1}$	1	...	$w_{2,N}$
	...	...	...	...	...
	N	$w_{N,1}$	$w_{N,2}$	...	1

- For each user a the top k most similar users are selected as the neighborhood of a .

Preference predictions

- To predict the preference of user a for product i we compute:

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u \in K} (r_{u,i} - \bar{r}_u) \cdot w_{a,u}}{\sum_{u \in K} w_{a,u}}$$

- From the full set of product preferences

	1	2	...	i	...	N
a	$p_{a,1}$			$p_{a,i}$		$p_{a,N}$

the top L products can be recommended to the user.

Considerations

- Different weighting schemes account for different aspects of data
- Users or items with too many ratings can bias predictions
 - Inverse user frequency (similar to inverse document frequency)
- Users or items with few ratings have unstable predictions
 - A default weight (bias) should be added in these cases
- The ratings of some users are considered as a good references
 - These users should get more weight

Item-based collaborative filtering

- The described approach computes a user similarity matrix
- The same steps can be applied for a matrix of product similarities
 - The similarity between two products can be computed as the Pearson correlation coefficient:

$$w_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2 \sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}}$$

Item-based collaborative filtering

- Given the matrix of product similarities

		Products			
		1	2	...	M
Products	1	1	$w_{1,2}$	...	$w_{1,M}$
	2	$w_{2,1}$	1	...	$w_{2,M}$
	...	...	...	...	...
	M	$w_{M,1}$	$w_{M,2}$	...	1

- The preference of user $\underline{a}$ for product $\underline{i}$ is given by:
$$p_{a,i} = \frac{\sum_{j \in K} r_{a,j} \cdot w_{i,j}}{\sum_{j \in K} |w_{i,j}|}$$

Matrix factorization methods

- The number of users and the number of products might be in the orders thousands
- Reducing the search space into a lower dimensional space helps computing meaningful recommendations
- The **goal** is to find this low-dimensional space to represent both products and user preferences.

Matrix factorization methods

- In matrix factorization methods, the user-products ratings matrix

$$R = \begin{bmatrix} r_{11} & \dots & r_{1M} \\ \dots & \dots & \dots \\ r_{N1} & \dots & r_{NM} \end{bmatrix}$$

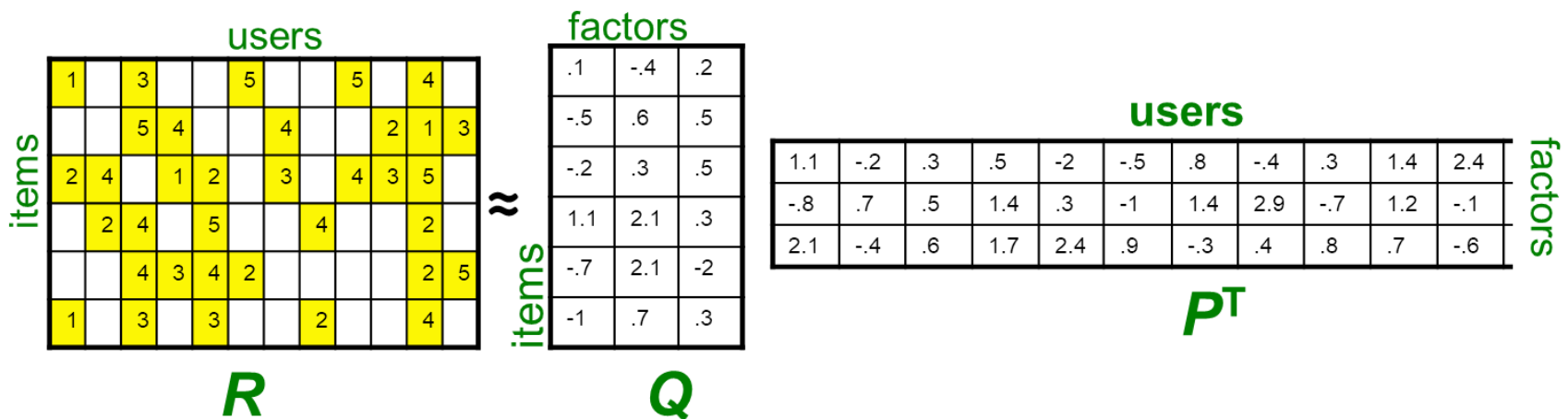
is decomposed into a k dimensional space of latent factors (each one corresponding to a dimension)

- Users and products are represented by a k dim. vector:

$$q_i = (q_{i1}, \dots, q_{ik})^T \quad p_u = (p_{u1}, \dots, p_{uk})^T$$

- Rating predictions are the inner product $r_{ui} = q_i^T p_u$

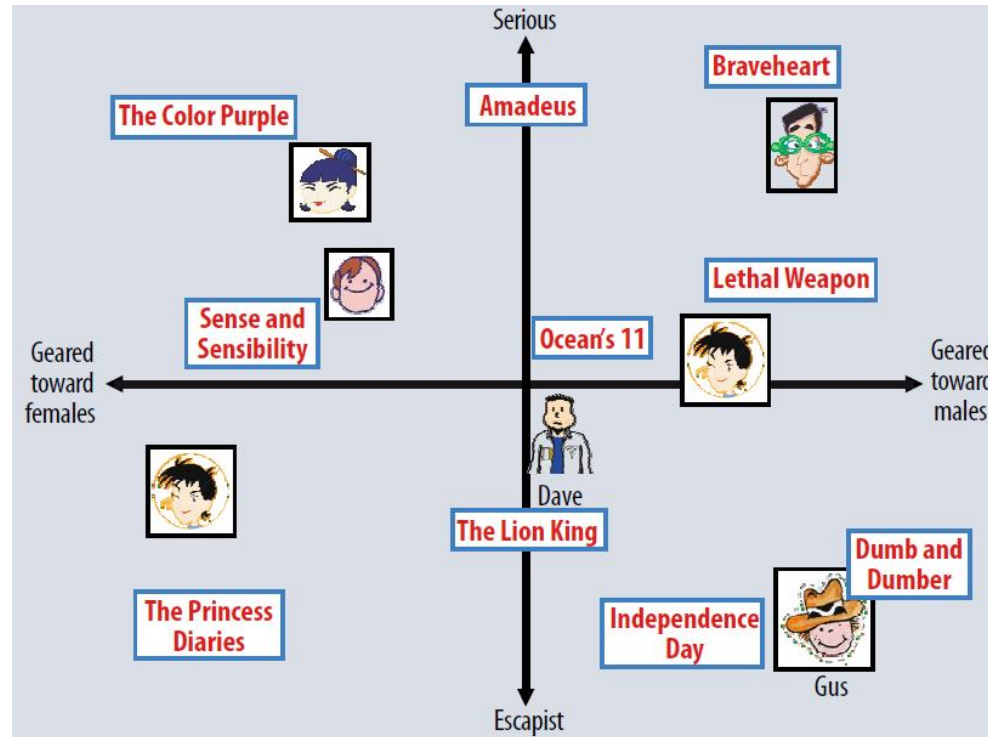
Latent factor models



- 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

Example of latent factors

- The two most important latent factors of the winning solution of the Netflix competition was:



Ratings as products of factors

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

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	

items

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

k factors

Q

k 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

PT

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

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

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

q_i = row i of Q

$\mathbf{p}_x$ = column x of $\mathbf{P}^\top$

Ratings as products of factors

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

[illegible]

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

k factors

Q



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

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

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

q_i = row i of Q

$\mathbf{p}_x$ = column x of $\mathbf{P}^\top$

users

k 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

PT

Ratings as products of factors

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

users

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	

items

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

k factors

Q



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

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

Diagram illustrating a matrix P^T (size $k \times \text{users}$) with highlighted values:

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

Approximating the matrix decomposition

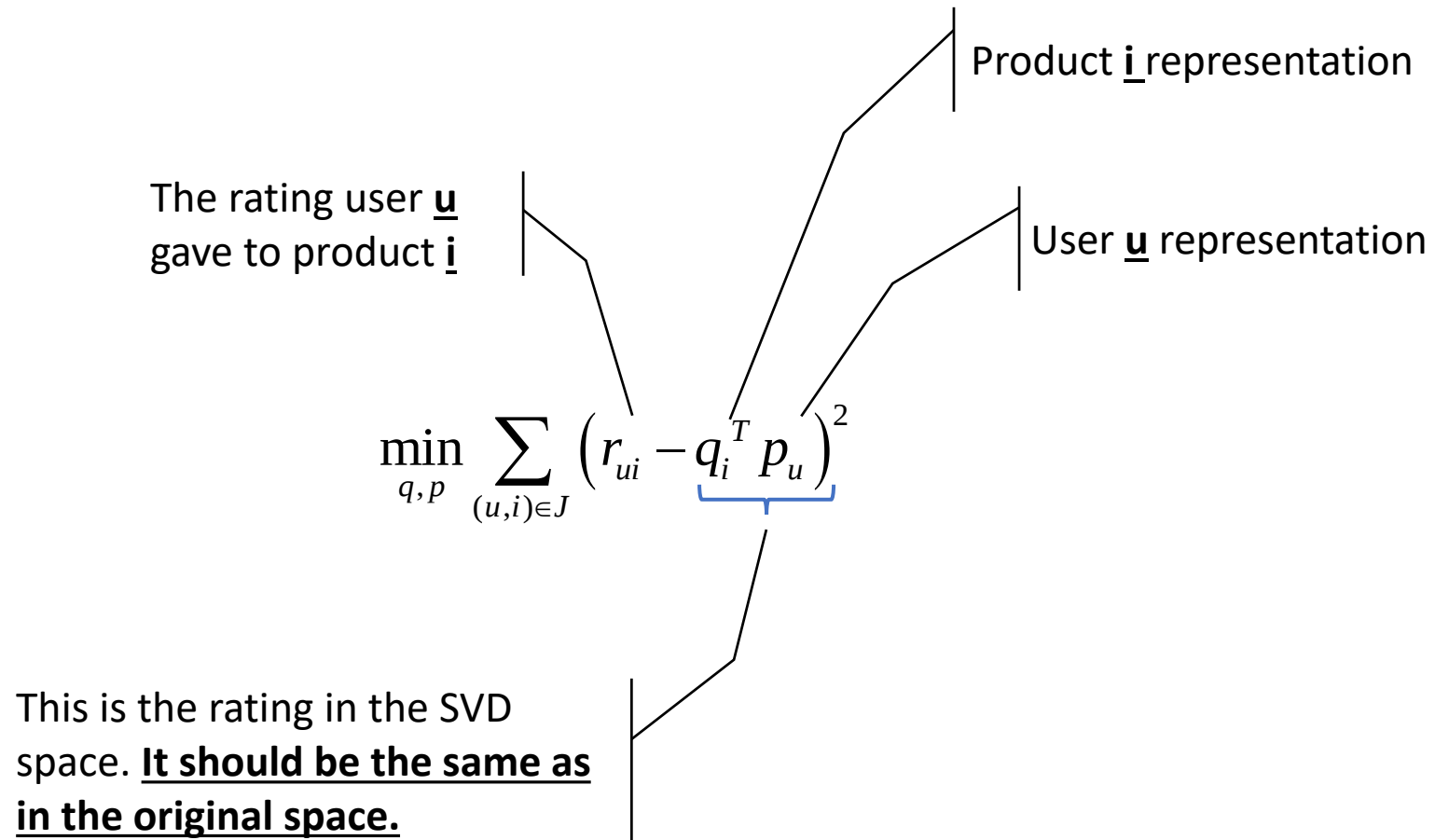
- Consider the products and users representation in the **k-dimensional** space :

$$q_i = (q_{i1}, \dots, q_{ik})^T \quad p_u = (p_{u1}, \dots, p_{uk})^T$$

- The SVD matrix decomposition into a **k latent factors** space is approximated by minimizing the difference between the set **J** of actual ratings and the ratings in the transformed space

- This is equivalent to:
$$\min_{q,p} \sum_{(u,i) \in J} (r_{ui} - q_i^T p_u)^2$$

Approximating the matrix decomposition



Accounting for user and product bias

- When rating products some users are more generous than others
 - This is the user bias: the average rating a user gives to products
- In general a product might receive higher ratings than others
 - This is the product bias: the average ratings the product receive
- Thus, the user preference for a given product must consider the average ratings, the product average rating and the user average rating

$$\min_{q,p} \sum_{(u,i) \in J} (r_{ui} - pr_{ui})^2$$

$$pr_{ui} = \mu + b_i + b_u + q_i^T p_u$$

Implicit preferences

- Cold start problem:
 - Some users provide very few ratings
 - Some products don't have many ratings
- Implicit preferences can be inferred by the system through the user profile
- Consider $N(u)$ the set of items for which user u expressed an implicit preference
- Consider $A(u)$ the set of user profile attributes such as age, gender, etc.

Implicit preferences

- Implicit product preferences are mapped into the factor model as:

$$\sum_{i \in N(u)} x_i \quad \frac{1}{\sqrt{|N(u)|}} \sum_{i \in N(u)} x_i$$

- Implicit profile preferences are mapped into the factor model as:

$$\sum_{i \in A(u)} y_i$$

- Thus, the SVD representation of the user u is completed with implicit preferences:

$$\min_{q, p} \sum_{(u, i) \in J} (r_{ui} - pr_{ui})^2$$

$$pr_{ui} = \mu + b_i + b_u + q_i^T \left(p_u + \frac{1}{\sqrt{|N(u)|}} \sum_{i \in N(u)} x_i + \sum_{a \in A(u)} y_a \right)$$

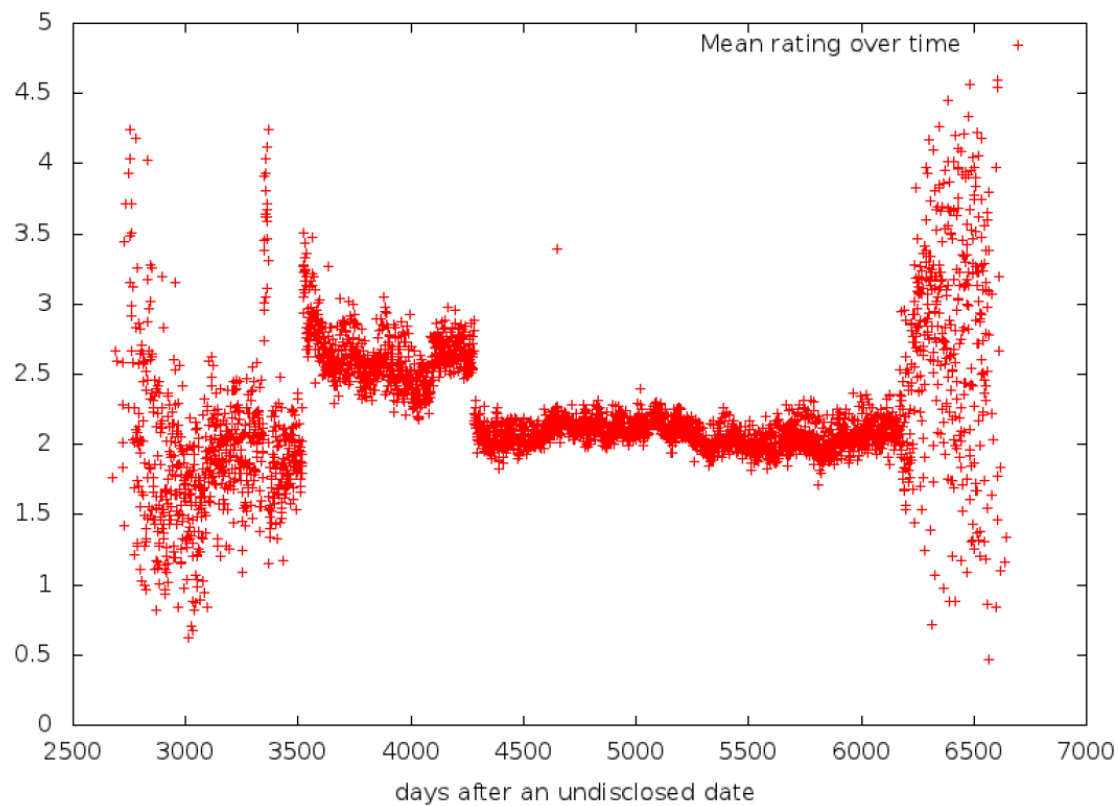
Clusters of users

- The above methods assume all users have the same bias and implicit preferences
- ... but users don't chose products randomly, they select products from a given group of products:
 - Their group of preferred produtcs.
- **Bias** and **implicit preferences** can in fact be computed from the group of users (cluster of users) to which the user belongs to.
- Clustering the products and the users will help in obtaining more accurate estimates of these values

Temporal dynamics

- User preferences change with time
 - Users tend to be more demanding or their preferences more refined and specific
 - A fan of thrillers might become a fan of crime dramas a year later
- Products popularity also change with time
 - Most of the time a product popularity decays with time
 - It can get popular after many months of its release (or years in some cases)
 - It can get popular again in the future (retro fashion, release of a movie remake)
- These dynamics might repeat over time.

Temporal dynamics



Temporal dynamics

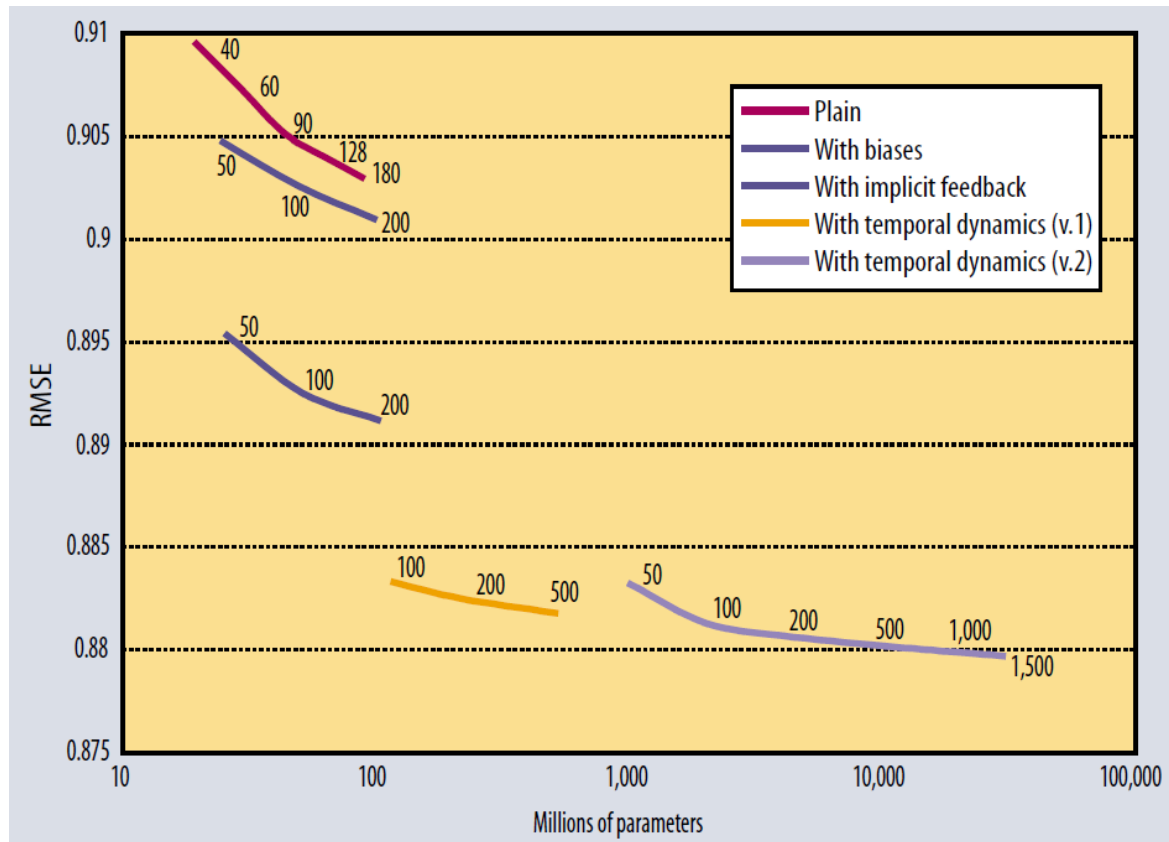
- The extension of factor models to incorporate temporal preferences is achieved by making biases and preferences a function of time

$$\min_{q,p} \sum_{(u,i) \in J} (r_{ui} - pr_{ui})^2.$$

$$pr_{ui} = \mu + b_i(t) + b_u(t) + q_i^T p_u(t)$$

- Classical methods include window based weighting and decaying weights
- Other more elaborate models can detect temporal patterns and predict a series of product selections

Example: performance results on NetFlix data



Million \$ Awarded Sept 21st 2009



Hybrid recommender systems

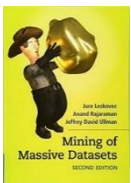
- Hybrid recommender systems combine both content-based profiles for each user and the collaborative ratings of products
- The simplest approach creates two separate rankings and combines them
- Other more elaborate and effective methods exist...

Hybrid recommender systems

- Content-based filtering methods can be used to learn a model about the products a user enjoys
 - This model can then predict the ratings of unrated products and this way reduce the sparsity of the ratings matrix
 - A collaborative filtering method can be applied next
- With content-based filtering methods clusters of users can be created by looking into their profiles
 - Predictions are made by applying collaborative filtering for the groups of users
- See (Melville, Sindhvani, 2010) for more references.

Summary

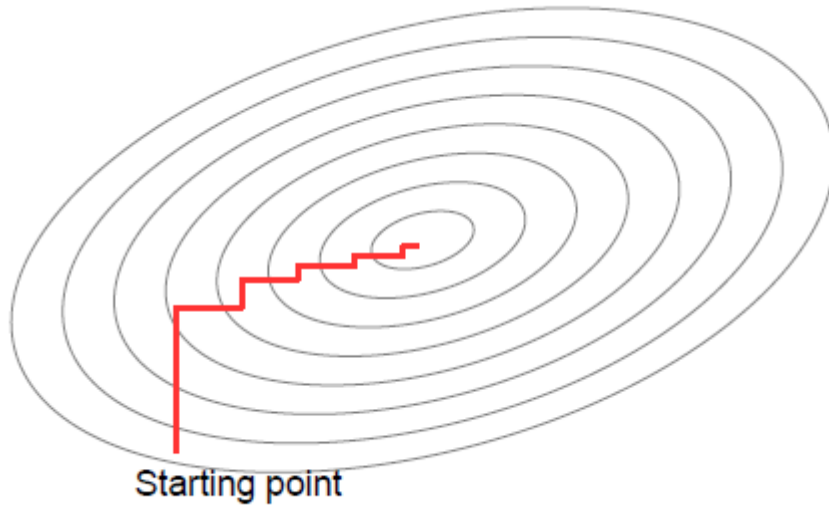
- Content-based recommendations
- Collaborative filtering
 - Neighborhood methods
 - Matrix factorization methods
- Hybrid recommender systems
- References:



[Chapter 9](#) of Jure Leskovec, Anand Rajaraman, Jeff Ullman, “**Mining of Massive Datasets**”, Cambridge University Press, 2011.

Minimizing the prediction error

- Coordinate descent algorithm performs successive line searches along the axes.



$$\min_{q,p} \sum_{(u,i) \in J} \left(r_{ui} - q_i^T p_u \right)^2$$

Algorithm

$p=0.1$, $q=0.1$, $l_{rate} = 0.001$, $\lambda = 0.02$

```
for iter_descent = 1:100
```

```
  for c = 1: num_coordinates
```

```
    for iter = 1:100
```

```
      for i,j where  $r(i,j) \neq 0$ 
```

$$err = r_{ui} - q_i^T p_u$$

$$p_{ic} += l_{rate} \cdot q_{jc} \cdot err - \lambda \cdot p_{ic}$$

$$q_{jc} += l_{rate} \cdot q_{jc} \cdot err - \lambda \cdot q_{jc}$$

```
    end
```

```
  end
```

```
end
```

```
end
```

