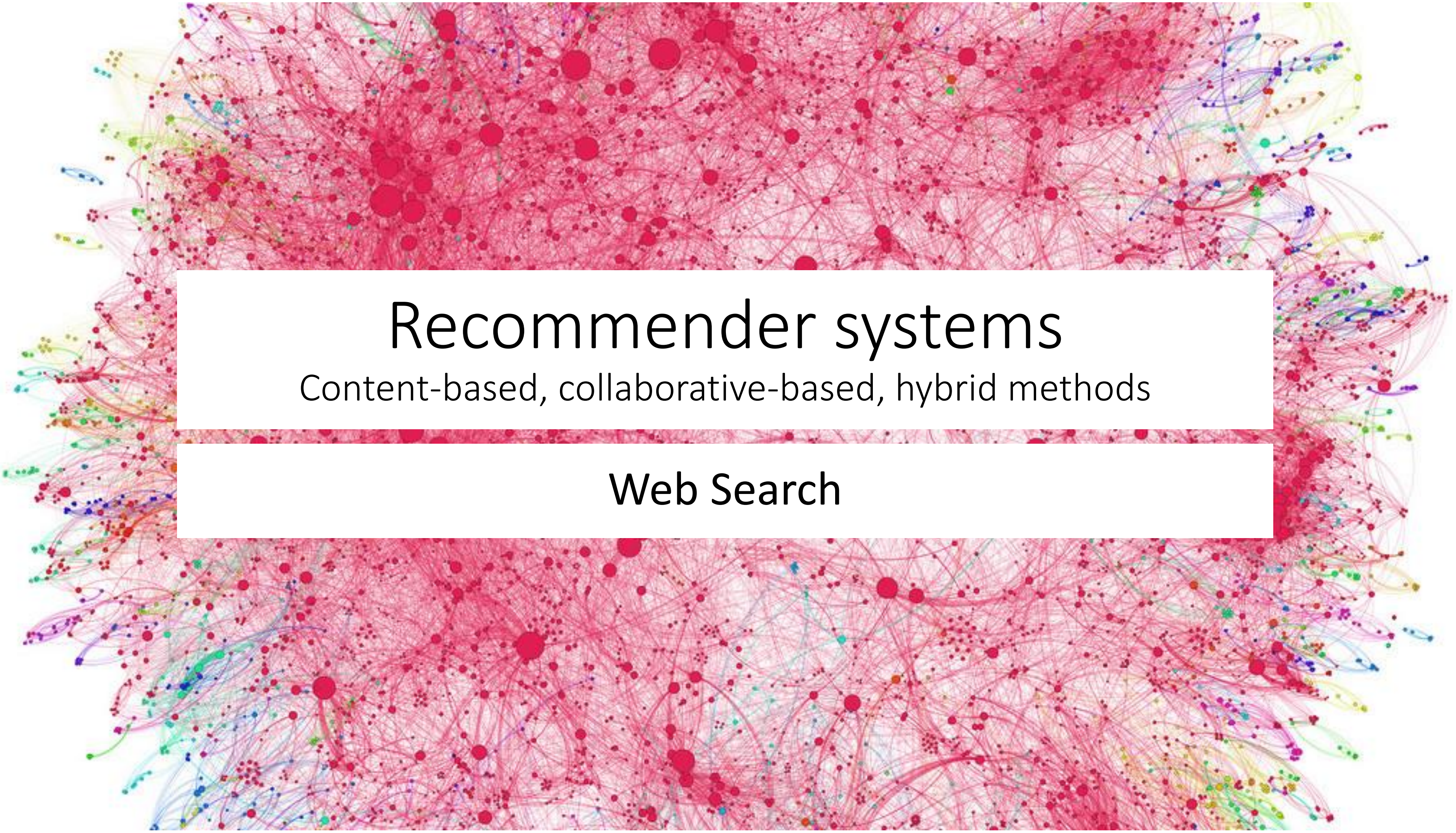
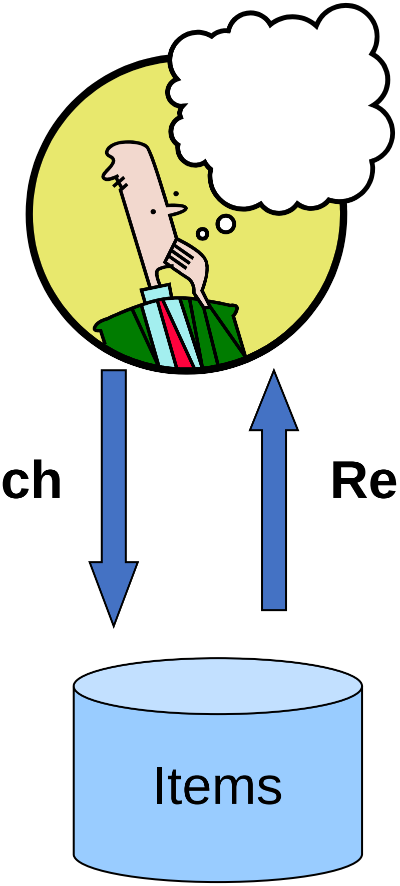


Recommender systems

Content-based, collaborative-based, hybrid methods

Web Search

Recommendations



Search

Recommendations

Items

Products, web sites, blogs, news items, …

Examples:

amazon.com

PANDORA

StumbleUpon

del.icio.us

NETFLIX

movielens
helping you find the *right* movies

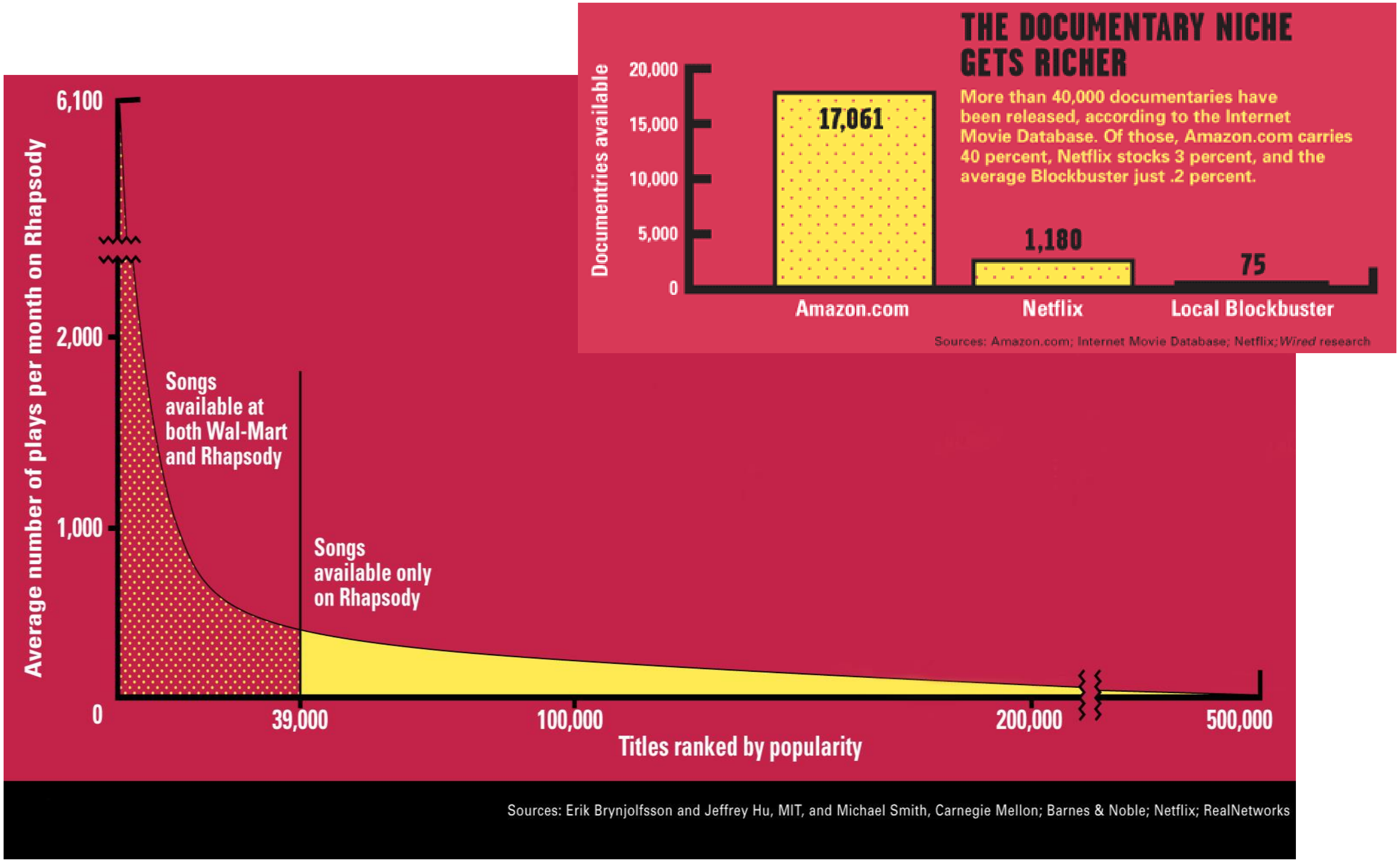
last.fm
the social music revolution

Google News

YouTube

XBOX LIVE

Sidenote: The Long Tail



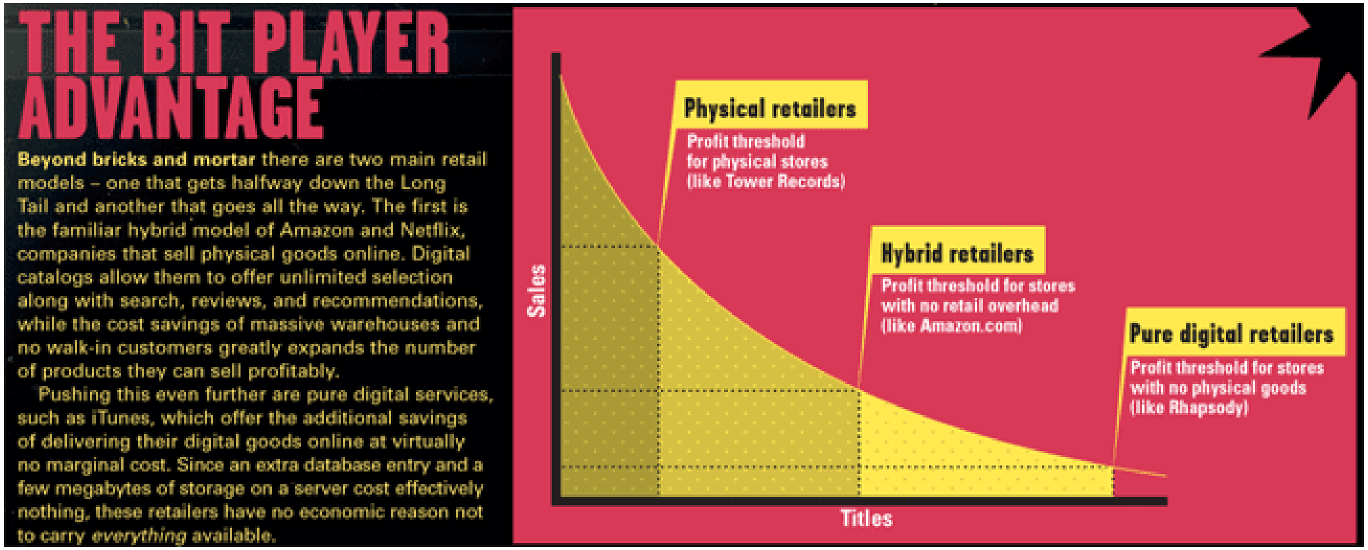
THE DOCUMENTARY NICHE GETS RICHER

More than 40,000 documentaries have been released, according to the Internet Movie Database. Of those, Amazon.com carries 40 percent, Netflix stocks 3 percent, and the average Blockbuster just .2 percent.

Sources: Amazon.com; Internet Movie Database; Netflix; *Wired* research

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

Brick-and-Mortar 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.

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

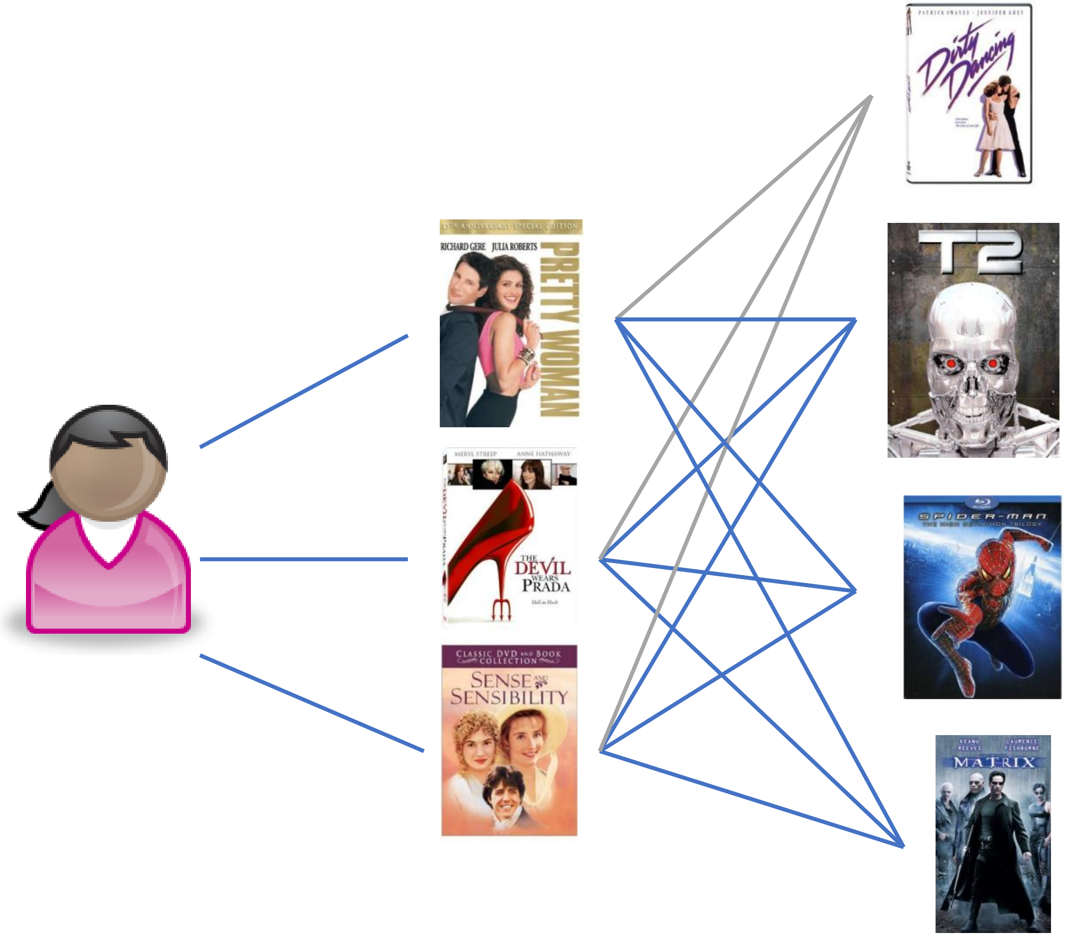
Recommender systems

- 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



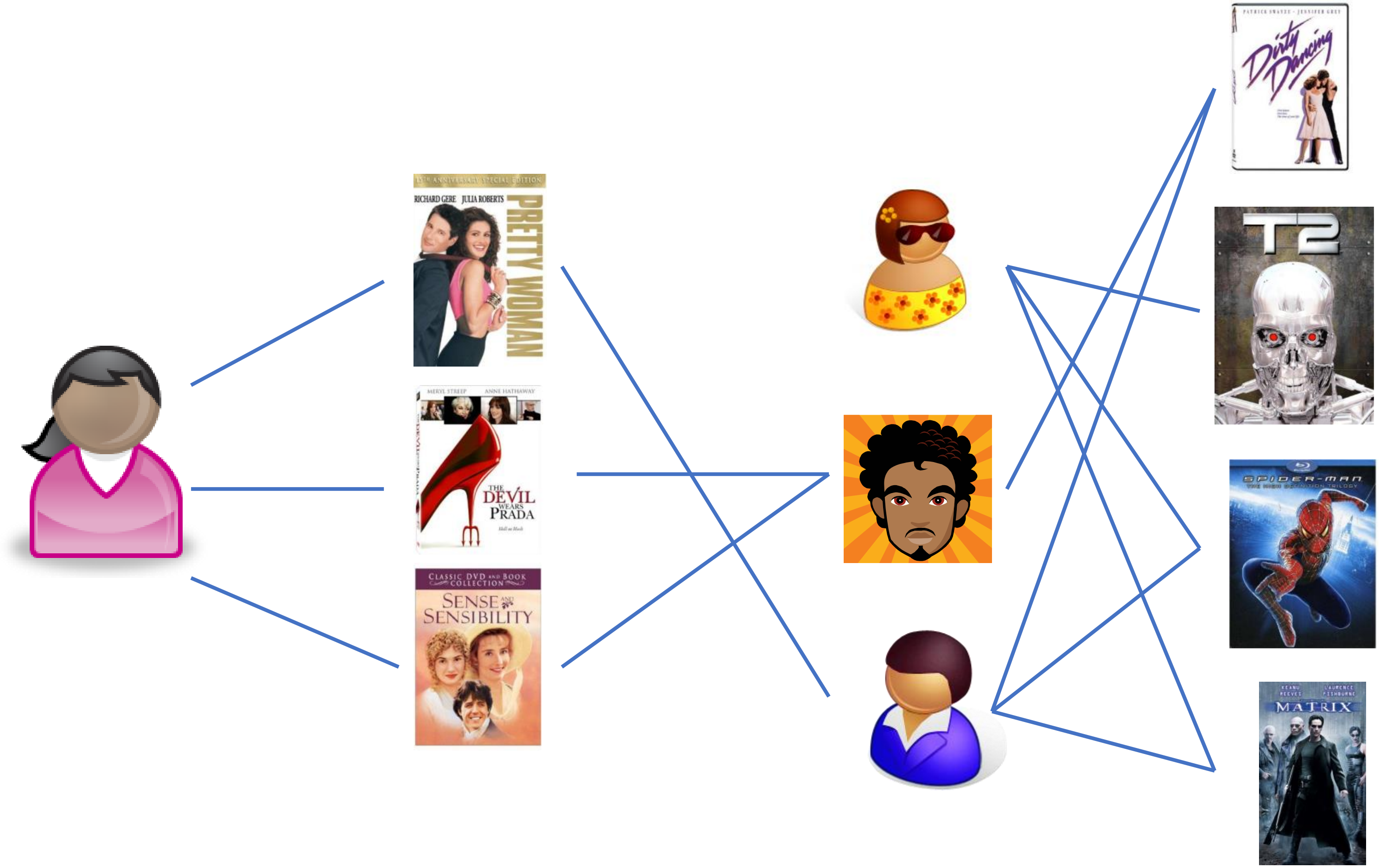
Content-based recommendations

- 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
    - <u>This is equivalent to the vector space discussed previously</u>

- 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
  - 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 from 1 (hate it) to 5 (love it)

- 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

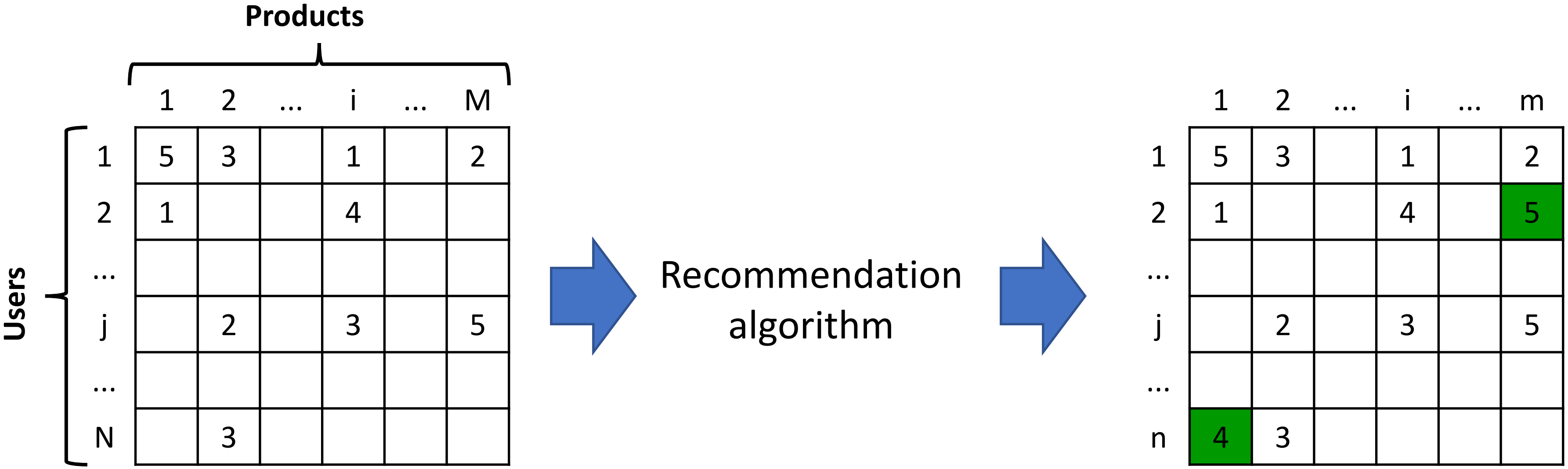
Products

Users

	1	2	...	i	...	M
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



	1	2	…	i	…	M
1	5	3		1		2
2	1			4		
…						
j		2		3		5
…						
N		3				

Recommendation algorithm

	1	2	…	i	…	m
1	5	3		1		2
2	1			4		5
…						
j		2		3		5
…						
n	4	3				

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 similarity 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:

$$w_{a,u} = \frac{\sum_{i \in I} \left(r_{a,i} - \overline{r_a} \right) \left(r_{u,i} - \overline{r_u} \right)}{\sqrt{\sum_{i \in I} \left(r_{a,i} - \overline{r_a} \right)^2} \sqrt{\sum_{i \in I} \left(r_{u,i} - \overline{r_u} \right)^2}}$$

Products

Users

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

- The resulting vector is the <u>similarity between user a and all other N users</u>:

	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

$$
\begin{array}{cc|c|c|c|c|}
 & & \multicolumn{4}{c}{\textbf{Users}} \\
 & & 1 & 2 & \dots & N \\
\hline
 & 1 & 1 & w_{1,2} & \dots & w_{1,N} \\
\hline
 & 2 & w_{2,1} & 1 & \dots & w_{2,M} \\
\hline
\textbf{Users} & \dots & \dots & \dots & \dots & \dots \\
\hline
 & N & w_{N,1} & w_{N,2} & \dots & 1 \\
\hline
\end{array}
$$

- For each user <u>**a**</u> the top <u>**k**</u> most similar users are selected as the neighborhood of <u>**a**</u>.

Preference predictions

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

$$p_{a,i} = \overline{r_a} + \frac{\sum_{u \in K} \left(r_{u,i} - \overline{r_u} \right) \cdot w_{a,u}}{\sum_{u \in K} w_{a,u}}$$

- Fom 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:

$$w_{i,j} = \frac{\sum_{u \in U} \left(r_{u,i} - \overline{r_i}\right)\left(r_{u,j} - \overline{r_j}\right)}{\sqrt{\sum_{u \in U} \left(r_{u,i} - \overline{r_i}\right)^2 \sum_{u \in U} \left(r_{u,j} - \overline{r_j}\right)^2}}$$

Item-based collaborative filtering

- Given the matrix of product similarities

Products

Products	1	2	…	M
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 **a** for product **i** is given by: $$p_{a,i} = \frac{\sum_{j\in K} r_{a,j} \cdot w_{i,j}}{\sum_{j\in K} \left| w_{i,j} \right|}$$

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.

Koren, Y., Bell, R., Volinsky, C. (2009). Matrix factorization techniques for recommender systems. IEEE Computer 42(8)

Matrix factorization methods

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

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

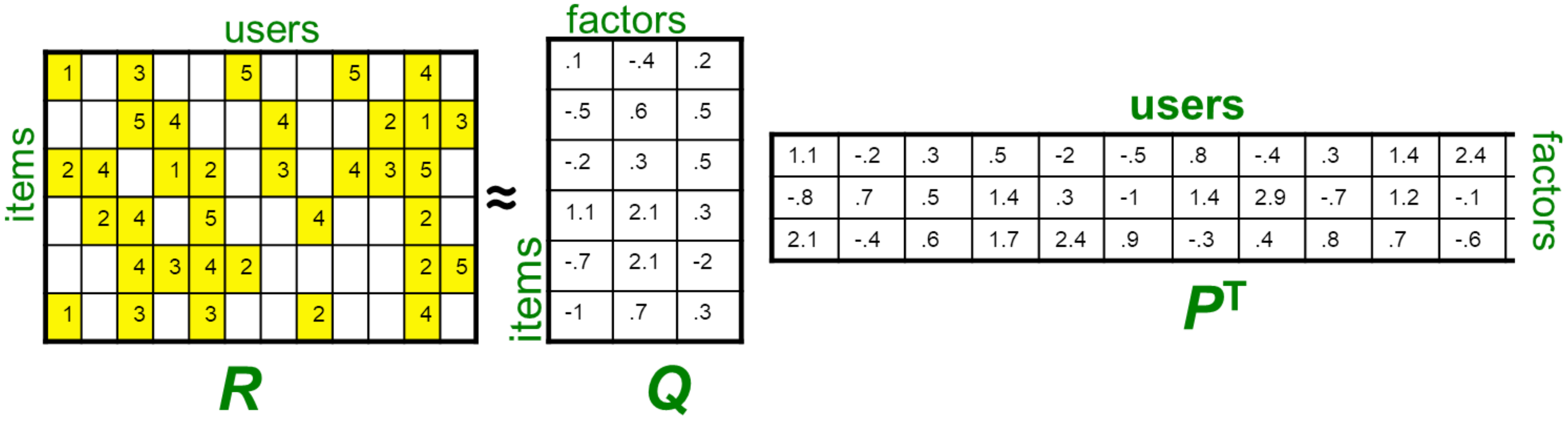
is decomposed into a ***k*** dimensional space of latent factors (each one corresponding to a dimmension)

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

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

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

Latent factor models



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	

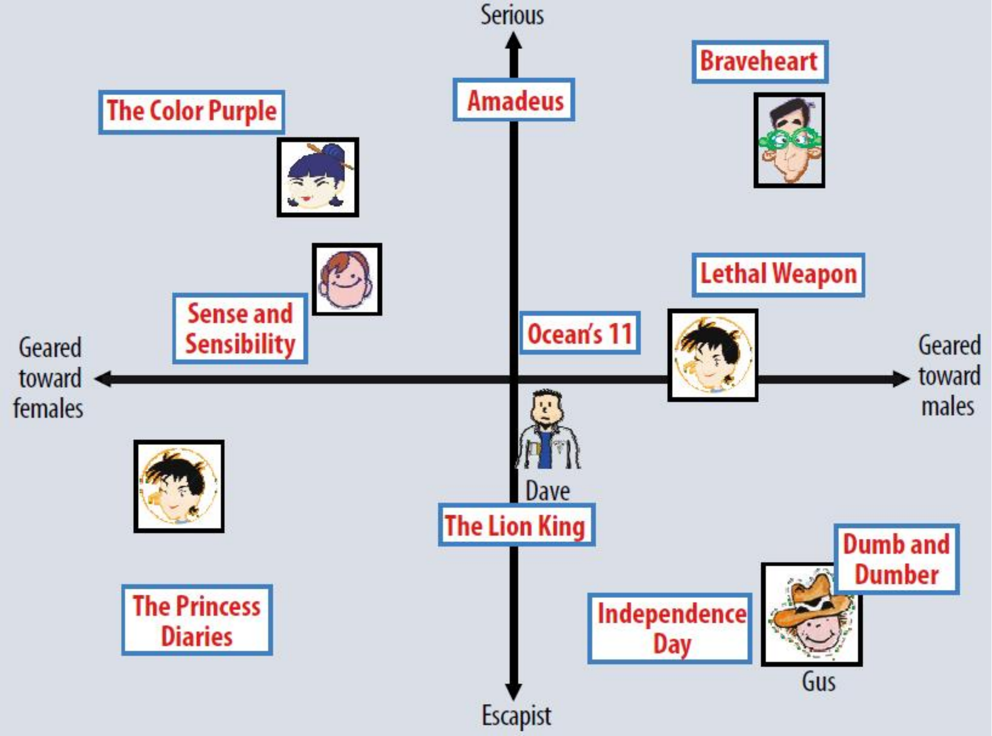
.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

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 ***R*** as a product of "thin" ***Q · 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:



Koren, Y., Bell, R., Volinsky, C. (2009). Matrix factorization techniques for recommender systems. IEEE Computer 42(8)

Ratings as products of factors

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

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

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

q_i = row *i* of ***Q***

p_x = column ***x*** of P^{T}

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	

≈

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

•

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

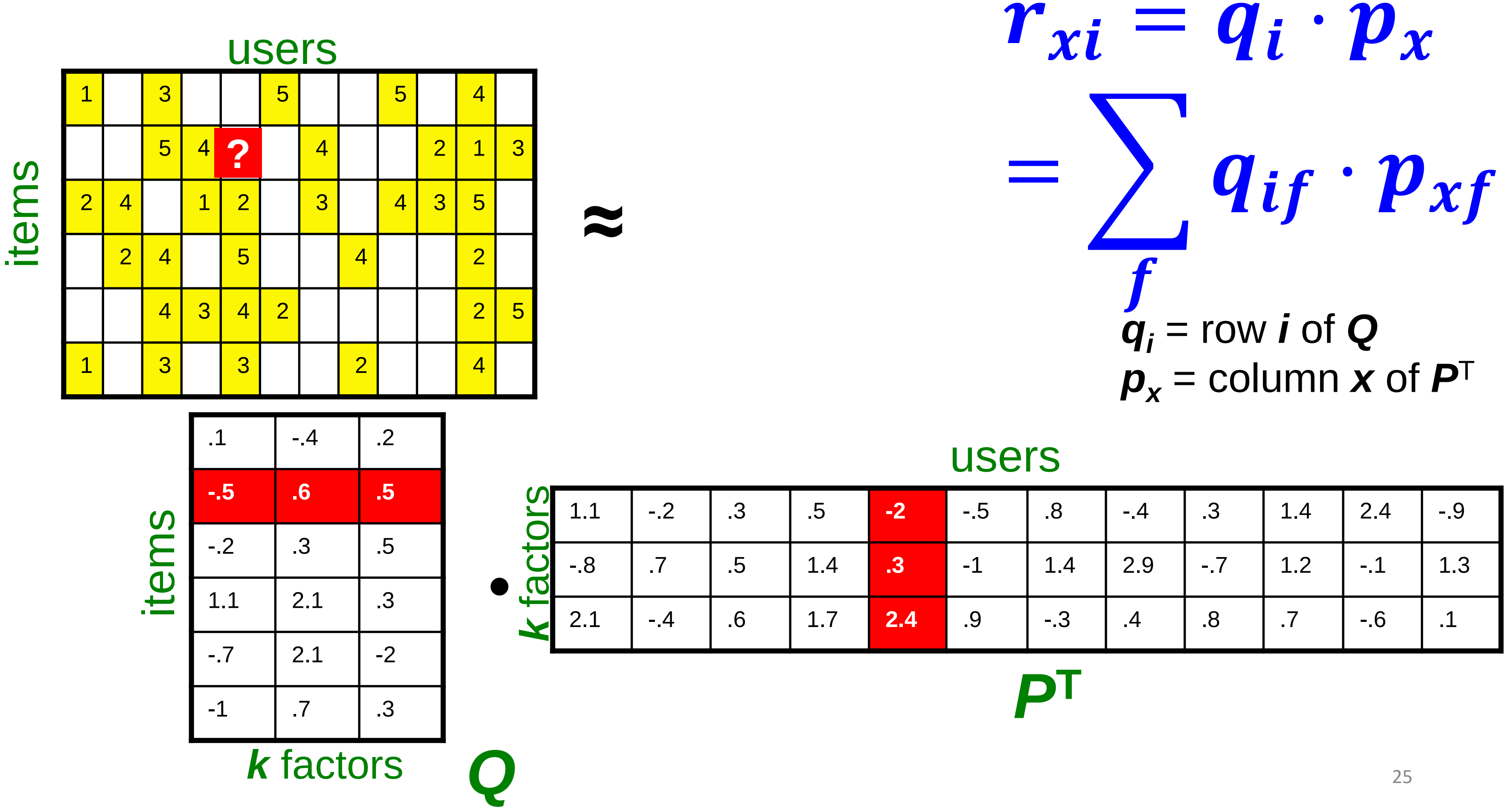
P^{T}

Ratings as products of factors

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

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

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



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	

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

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

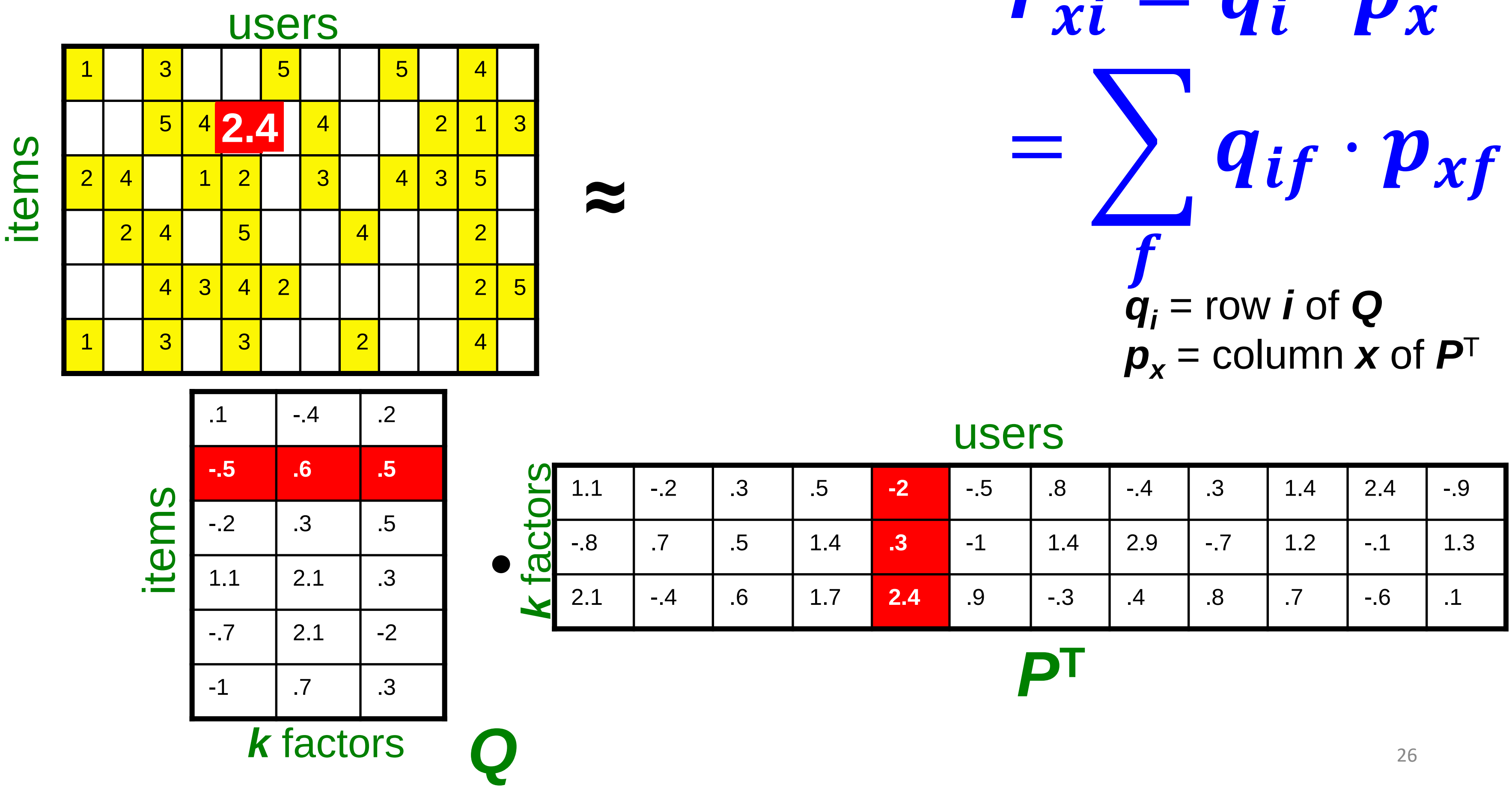
Ratings as products of factors

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

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

q_i = row i of Q

p_x = column x of P^T



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	

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

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 :

$$q_i = \left(q_{i1}, \ldots, q_{ik}\right)^T \qquad p_u = \left(p_{u1}, \ldots, p_{uk}\right)^T$$

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

Approximating the matrix decomposition

Product **i** representation

The rating user **u** gave to product **i**

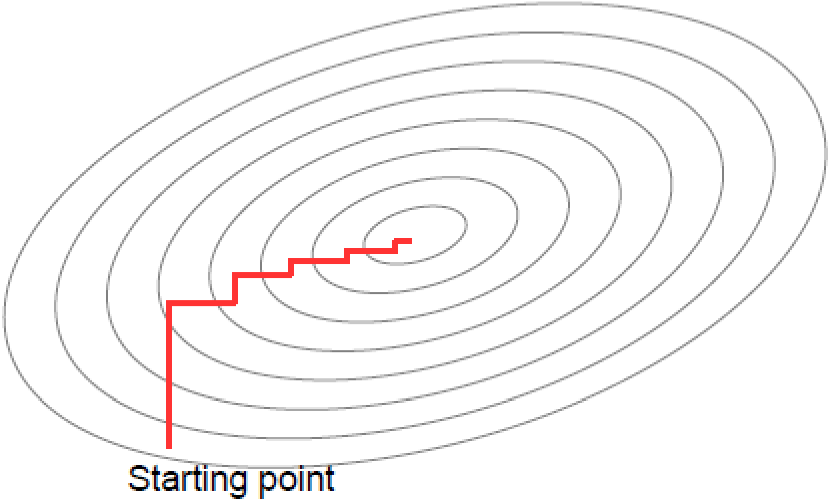
User **u** representation

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

This is the rating in the SVD space. **It should be the same as in the original space.**

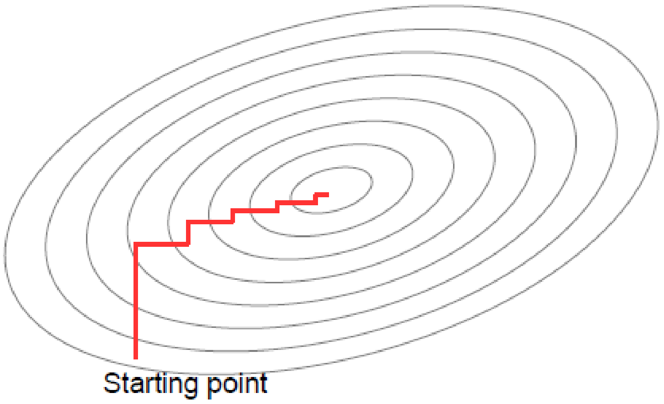
Minimizing the prediction error

- 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, lrate = 0.001

for iter_descent = 1:100
  for c = 1: factors
        for iter = 1:100
                for i,j where r(i,j) !=0
                   $err = r_{ui} - {q_i}^T p_u$
                   $p_{ic} = p_{ic} + l_{rate} \cdot \left(q_{jc} \cdot err\right)$
                   $q_{jc} = q_{jc} + l_{rate} \cdot \underbrace{\left(p_{jc} \cdot err\right)}_{\text{Gradient of the cost function}}$
                end
        end
  end
end
```

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

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

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

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

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:

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

- 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:

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

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

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

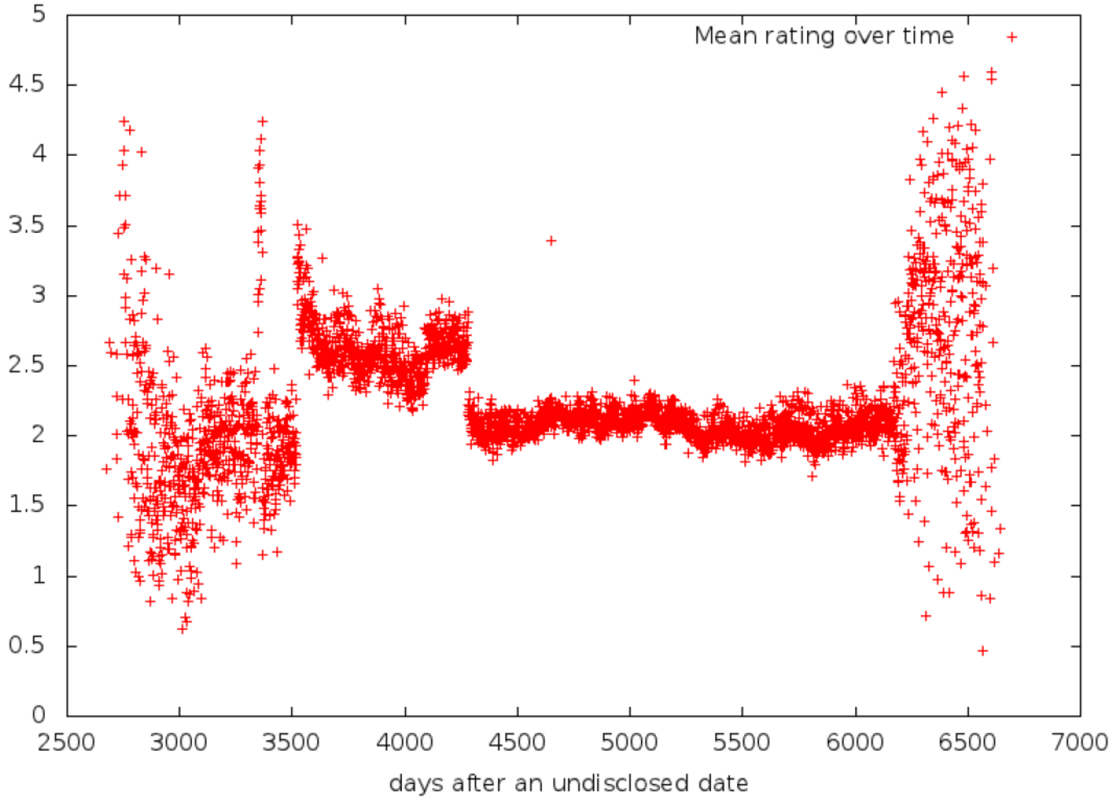
Temporal dynamics

- 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

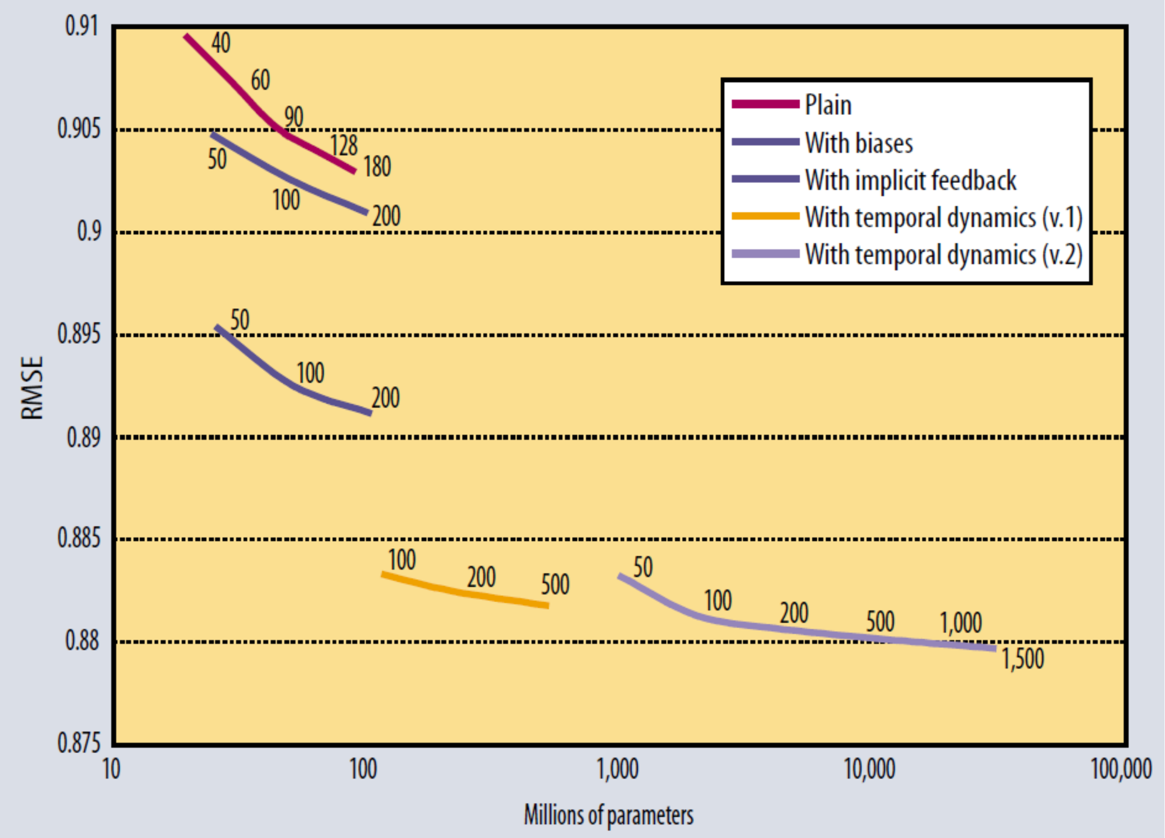
$$\min_{q,p} \sum_{(u,i)\in J} (r_{ui} - pr_{ui})^2$$

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

- 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



Koren, Y., Bell, R., Volinsky, C. (2009). Matrix factorization techniques for recommender systems. IEEE Computer 42(8)

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, Sindhwani, 2010) for more references.

Summary

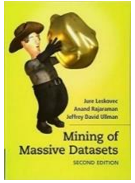
- Content-based recommendations

- Collaborative filtering
  - Neighborhood methods
  - Matrix factorization methods

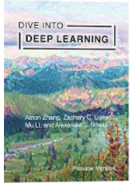
- Hybrid recommender systems

Readings

- Koren, Y., Bell, R., Volinsky, C. (2009). Matrix factorization techniques for recommender systems. IEEE Computer 42(8).



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



Chapter 16 of Aston Zhang, Zack C. Lipton, Mu Li, Alex J. Smola, “**Dive into Deep Learning”**

- Software:
  - https://deepctr-torch.readthedocs.io/en/latest/