

# **Part 13: Item-to-Item Collaborative Filtering and Matrix Factorization**



**Francesco Ricci**

# Content

- Item-to-item collaborative filtering
- Fast computing of predictions
- Comparison with non-personalized approaches
- What happen if:
  - we perturbate the data
  - or use less data?
- Clustering and collaborative filtering
- Matrix factorization techniques

# Item-to-Item Collaborative Filtering

Amazon.com: Professional Techniques for Black & White Digital Photography: Books...

File Edit View Go Bookmarks Tools Help

http://www.amazon.com/gp/product/1584281499/ref=pd\_ys\_ir\_b\_1/102-

### Better Together

Buy this book with [A Comprehensive Guide to Digital Black & White...](#) by John Clements today!

**Total List Price:** ~~\$52.90~~  
**Buy Together Today:** **\$35.38**

[Buy both now!](#)

---

### Customers who bought this also bought

- [A Comprehensive Guide to Digital Black & White Photography \(Digital Photography\) by John Clements](#)
- [Digital Black & White Photography by John Beardsworth](#)
- [Amphoto's Guide To Digital Black And White Printing: Techniques For Creating High Quality Prints by George Schaub](#)
- [Darkroom To Digital: Black And White Photography With Photoshop - The Art Of Transitions by Eddie Ephraums](#)
- [The Advanced Digital Photographer's Workbook : Professionals Creating and Outputting World-Class Images by Yvonne J. Butler](#)

**Explore similar items:** in [Books](#), in [Magazine Subscriptions](#)

---

### Editorial Reviews

#### Book Description

This professional guidebook provides tips to create technically correct and highly marketable digital black-and-white photographs. Designed to instruct professional and experienced amateur photographers, this highly visual format features 100 landscape, portrait, and wedding images from 20 leading digital imaging experts. Advice is included for utilizing professional digital effects, selecting an appropriate SLR camera, and managing difficulties and rewards associated with creating high-end black-and-white digital images.

Find:  Find Next Find Previous Highlight Match case

# Similarity of item i with item 17

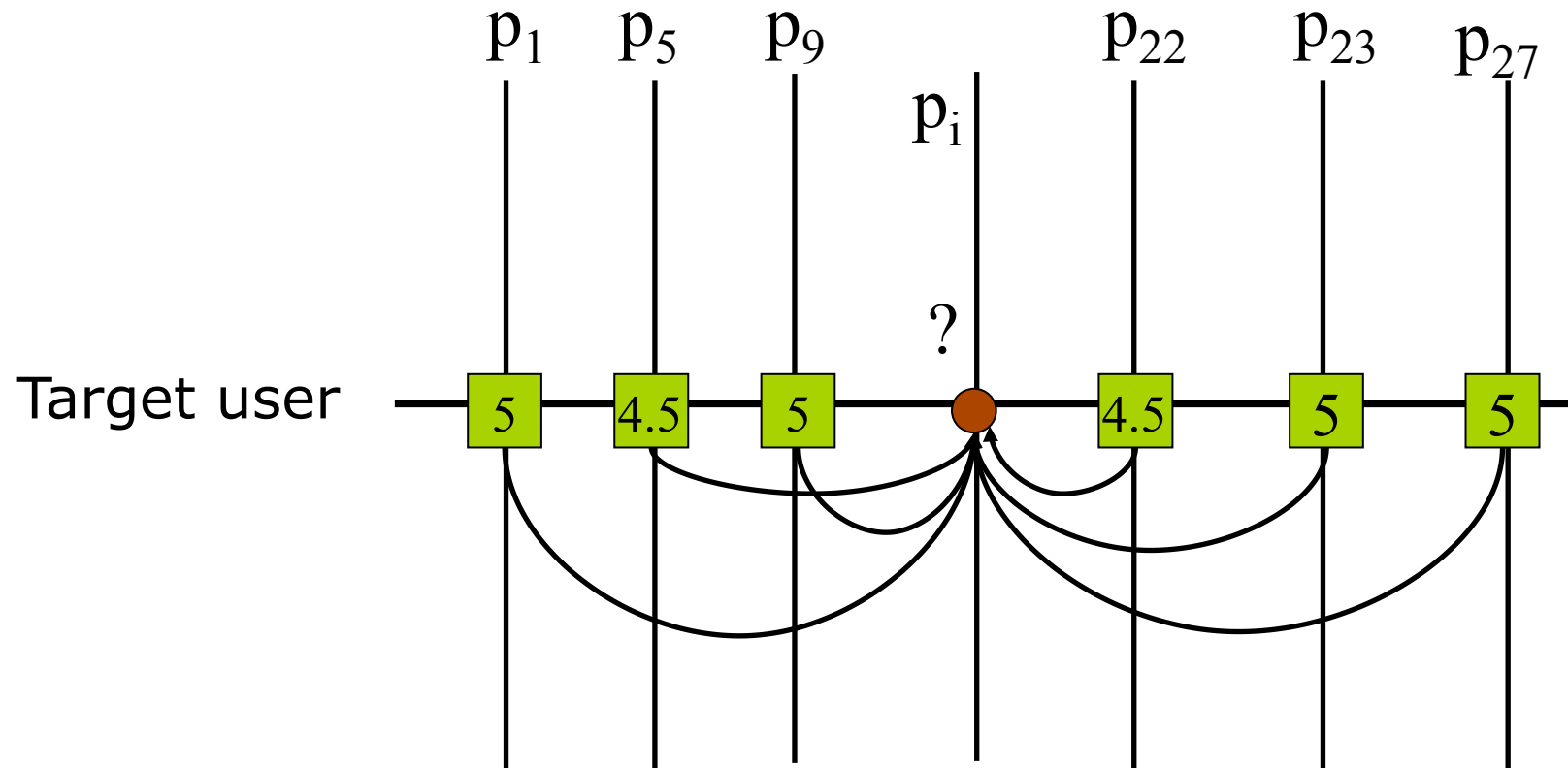
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
,1	,3	,6	,1	,3	,4	,3	,3	,2	,6	,2	,5	,4	,5	,5	,3	1	,3	,5	,4	,2	,4	,4	,5

Items

Users

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
a			1		4	5			4		3					2			4		2				
b			4								3						5	1		3					
c		5		4			4						3		5				4		5				
d								3				5				3			4		2			3	
e		3					5			4	5				5					1			5	4	
f			4				1		3	5		4	1		5	4	4		4				3		
g	2	4				4		2			5		1	4	5		4	2	4		5			4	
h			2		1		4		3	5		4	2		5	4	5					5			
i			1				3			5				5		4	4		5			4		3	
j			4			4				5		1		5		4	4		4			4			
k		5				4			2		5		1	5		4		2		4				2	
l					3			3				4	1		4		4	2	4					3	
m	5		3					5	3		5	4		5	5	3			4	4	5	4		4	
n			1		4	5				4	5		1	5		4		3		4		4	3		
o			4			4				5		4		5		4	4	2		5		5		3	
p				4				5						5	4			2	4	4	5	4		2	
q					3			3					1	5		4	4		4			4		3	
r		4			1	4		2					2		5		4				5	4		4	
s			2		4		4			5			1			4			2	4		4		5	
t		1		4			3					4		5	5		4			4				3	
u			2		1		4		3				1		5	4			2	4		5	4		
v					4	5				4	3		5			2					2			5	
w				2			2		3			5			4	5			4	2		3	4		
x	4			5				3		3				4	5						1				
y			1			3				2	3						3	3		5		4			

# Item-to-item CF: the basic idea



- Can the ratings of the target user for similar items be exploited for predicting an unknown rating?
- Yes but not all the similar items should be equally relevant.

# Item-to-Item Collaborative Filtering

- ❑ Rather than matching user-to-user similarity, item-to-item CF matches items purchased or rated by a target user to **similar items** and combines those similar items in a recommendation list
- ❑ It seems like a *content-based filtering* method (see next lecture) as the match/similarity between items is used
- ❑ In fact it can work (in a simplified way) much like content-based methods will do:
  - One item describes the current interests of the user (user model)
  - The other similar items are those to be recommended.

[Linden et. al, 2003]

# Item-to-Item Similarity

- Similarity can be computed in a number of ways
  - Using the user ratings
  - Using some product description
  - Using co-occurrence of items in a bag or in the set of a user past purchased products
- A collection of user  $u_i, i=1, \dots, n$  and a collection of products  $p_j, j=1, \dots, m$
- A  $n \times m$  matrix of rates  $v_{ij}$ , with  $v_{ij} = ?$  if user  $i$  did not rate product  $j$

$$\cos(p_i, p_j) = \frac{\sum_{k=1}^m v_{ki} v_{kj}}{\sqrt{\sum_{k=1}^m v_{ki}^2 \sum_{k=1}^m v_{kj}^2}}$$

# Prediction Computation

- **Generating the prediction:** look into the target user's ratings and use techniques to obtain predictions based on the ratings of similar products
- Weighted Sum of the ratings of the active user to similar items

$$v_{ui}^* = \frac{\sum_j s_{ij} * v_{uj}}{\sum_j s_{ij}}$$

- The sum is over a subset (neighbor) of all the similar items (to the target i) that the user u has rated ( $v_{uj}$ ) –  $s_{ij}$  is the similarity of i and j

# Computing the Item-to-Item Similarity

- Build a product-to-product matrix of similarities by iterating through all possible pairs
  - *Inefficient because many pairs have no common customers!*
- A better approach for selecting pairs of items for which the similarity can be computed is:
  1. Scan the products, and for all the customers that bought a product, identify the other products bought by those customers
  2. Then compute the similarity only for these pairs

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
a			1		4	5			4		3				2			4		2					
b		4								3							5	1		3					
c		5		4			4					3		5						4		5			
d								3				5				3			4		2				3
e		3					5			4	5				5					1			5	4	
f			4					1		3	5		4	1		5	4	4		4					3
g	2	4				4		2			5		1	4	5			4	2	4		5			4
h			2		1		4		3	5		4	2		5	4	5								5
i		1					3			5				5	4	4			5			4			3
j		4			4					5			1		5		4		4		4				4
k		5				4			2		5		1	5		4		2			4				2
l					3			3				4	1		4		4	2	4						3
m	5		3					5	3		5	4		5	5	3				4	4	5	4		4
n			1		4	5				4	5		1	5		4		3		4		4		3	
o		4				4				5		4		5			4	2		5		5			3
p			4				5								5	4		2	4	4	5	4			2
q					3			3					1	5		4	4		4			4			3
r		4			1	4		2					2		5		4				5	4			4
s			2		4		4			5			1			4		2	4		4				5
t		1		4			3					4		5	5		4			4					3
u			2		1		4		3				1		5	4		2	4		5	4			
v					4	5				4	3		5			2					2				5
w				2			2		3			5			4	5		4	2		3	4			
x	4			5				3		3				4	5						1				
y			1				3				2	3						3	3		5				4

To compute the similarity of product 1 to the others

Discover that only g, m and x bought p1

Then take the union of the products bought by g, m and x

(2 6 8 11 13 14 15 17 18 19 21 24 3 9 12 16 20 22 4 10)

# Computing the Item-to-Item Similarity

For each item in product catalog,  $I_1$   
For each customer  $C$  who purchased  $I_1$   
For each item  $I_2$  purchased by  
customer  $C$   
Record that a customer purchased  $I_1$   
and  $I_2$   
For each item  $I_2$   
Compute the similarity between  $I_1$  and  $I_2$

We must have:

**For each customer**  $u_i$  the list of products bought by  $u_i$  (i.e. the indexes  $j$  of  $v_{ij}$  such that  $v_{ij} \neq ?$ )

For each product  $p_j$  the list of users that bought that (i.e. the indexes  $i$  of  $v_{ij}$  such that  $v_{ij} \neq ?$ )

Have you already seen these kind of indexes?

# Item-to-Item Predictions for a User

- If the goal is to find the recommendations for a user, then
- For each item  $i_j$ ,  $j=1, \dots, m$  in the profile of a user
  - Find the top- $n$  similar items of  $i_j$   $\text{Top-}n(i_j)$
  - This can be computed with standard IR techniques (inverted index linking each user to its "postings", i.e., items rated by the user)
- For the union of the items in  $\text{Top-}n(i_j)$  compute the predictions
  - You use the similarities with the items in the user's profile that you computed above.

# Performance Implications

- A **user profile** normally contains **less ratings** than a product profile!
- User-based CF – **similarity between users is dynamic**, pre-computing user neighborhood can lead to poor predictions
  - Because the similarity between users can change if only a few ratings are changing (the overlap between users' profiles is small)
- Item-based CF – **similarity between items is more static**
- This enables pre-computing of item-item similarity => prediction process involves only a table lookup for the similarity values & computation of the weighted sum.

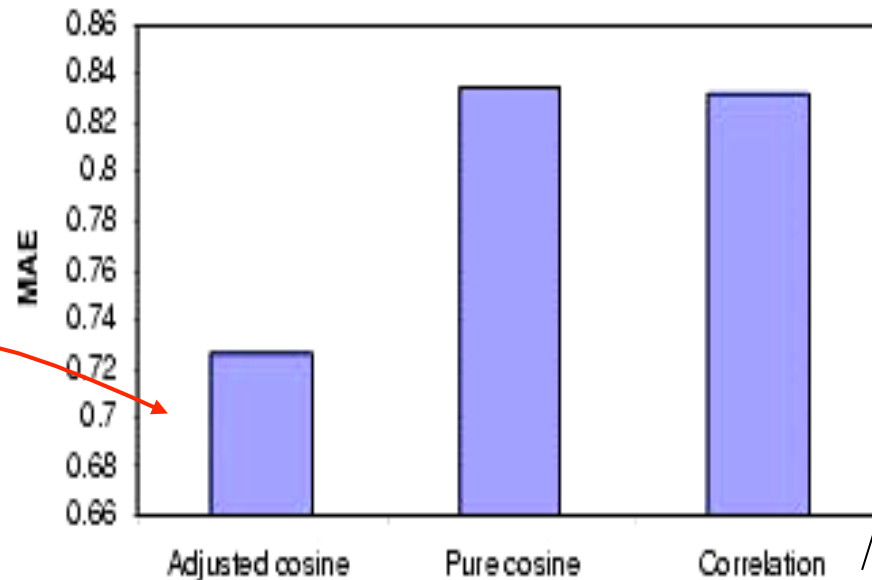
## Experiments : The Data Set

- ❑ MovieLens – a web-based movies recommender system with 43,000 users & over 3500 movies
- ❑ Used 100,000 ratings from the DB (only users who rated 20 or more movies).
- ❑ 80% of the data - training set.
- ❑ 20% of the data - test set.
- ❑ Data is in the form of user-item matrix. 943 rows (users), 1682 columns (items/movies – rated by at least one of the users).

[Sarwar et al., 2001]

# Effect of similarity Algorithms

- Impact of similarity computation measures on item-based CF algorithm

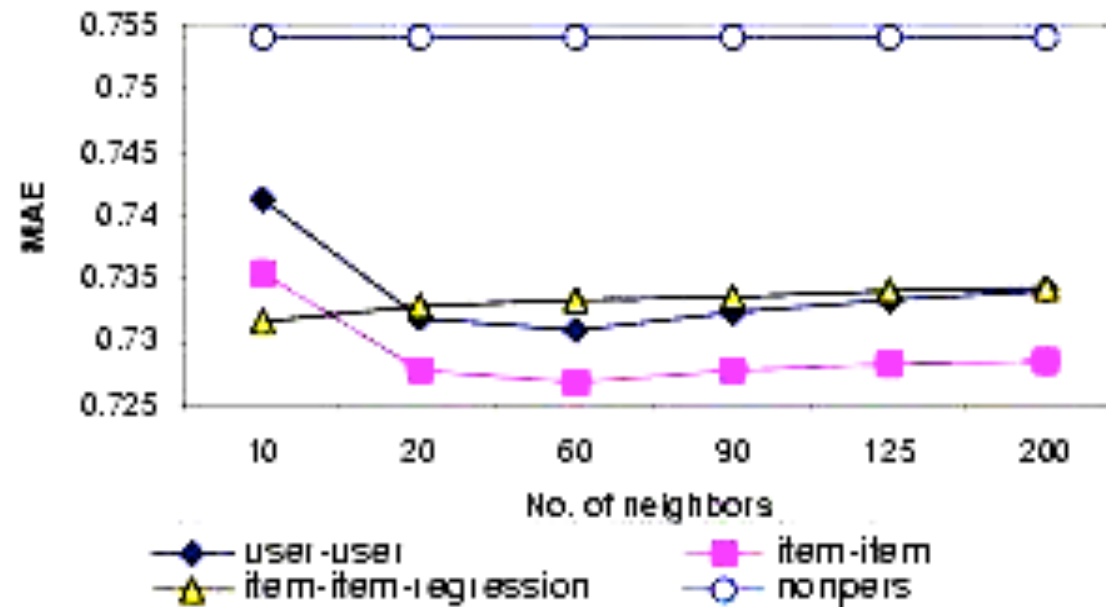


What is the difference between Adjusted cosine and Correlation?

- In adjusted cosine instead of using the ratings  $v_{uj}$ , they are used  $(v_{uj} - v_u)$  – where  $v_u$  is the average of the ratings of the user  $u$ .

# Quality Experiments

- Item-to-item vs. user-to-user based at selected neighborhood sizes
- (train/test ratio = 0.8)



## Item-to-item vs User-to-User

- ❑ item-item CF normally provides better predictions than the user-user CF
- ❑ Improvement is consistent over different neighborhood sizes and train/test ratio
- ❑ Improvement is not significantly large
- ❑ Item neighborhood is fairly static, hence enables pre-computation which improves online performance.

# Personalised vs Non-Personalised CF

- Collaborative-based recommendations are **personalized** since the rating “prediction” differs depending on the target user and it is based on:
  - User-to-user: the ratings (for a given item) expressed by users that are **similar** to the **active user**
  - Item-to-item: the **weighted** average of the ratings of the **active user** for **similar** items
- A **non-personalized** collaborative-based recommendation can be generated, for instance, by averaging the recommendations for ALL the users
  - All the users receive the same recommendations

# Personalised vs Non-Personalised CF

Data Set	users	items	Lower Upper rating	total ratings	Av. RatedIt	density	average	stddev	MAE Non Pers	MAE Pers
Jester	48483	100	-10 - +10	3519449	72,59	0,725	0,816	4,40	0,220	0,152
MovieLens	6040	3952	1 - 5	1000209	165,59	0,041	3,580	0,934	0,233	0,179
EachMovie	74424	1649	0 - 1	2811718	37,77	0,022	0,607	0,223	0,223	0,151

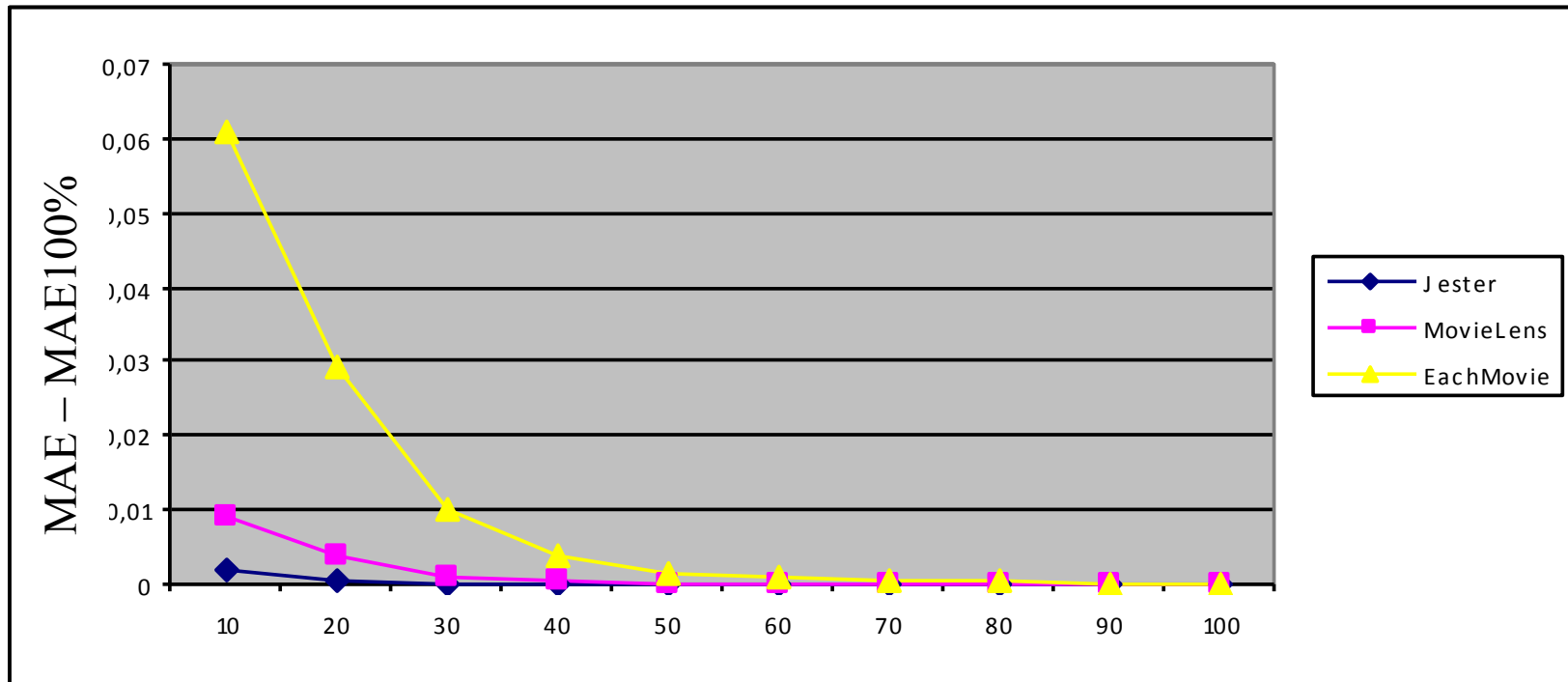
Normalized MAE

Is this difference important?

$$MAE(NP) = \frac{\sum_{i,j} |v_{ij} - v_j|}{\# ratings}$$

$v_{ij}$  is the rating of user  $i$  for product  $j$  and  $v_j$  is the average rating for product  $j$

# Using Less Data

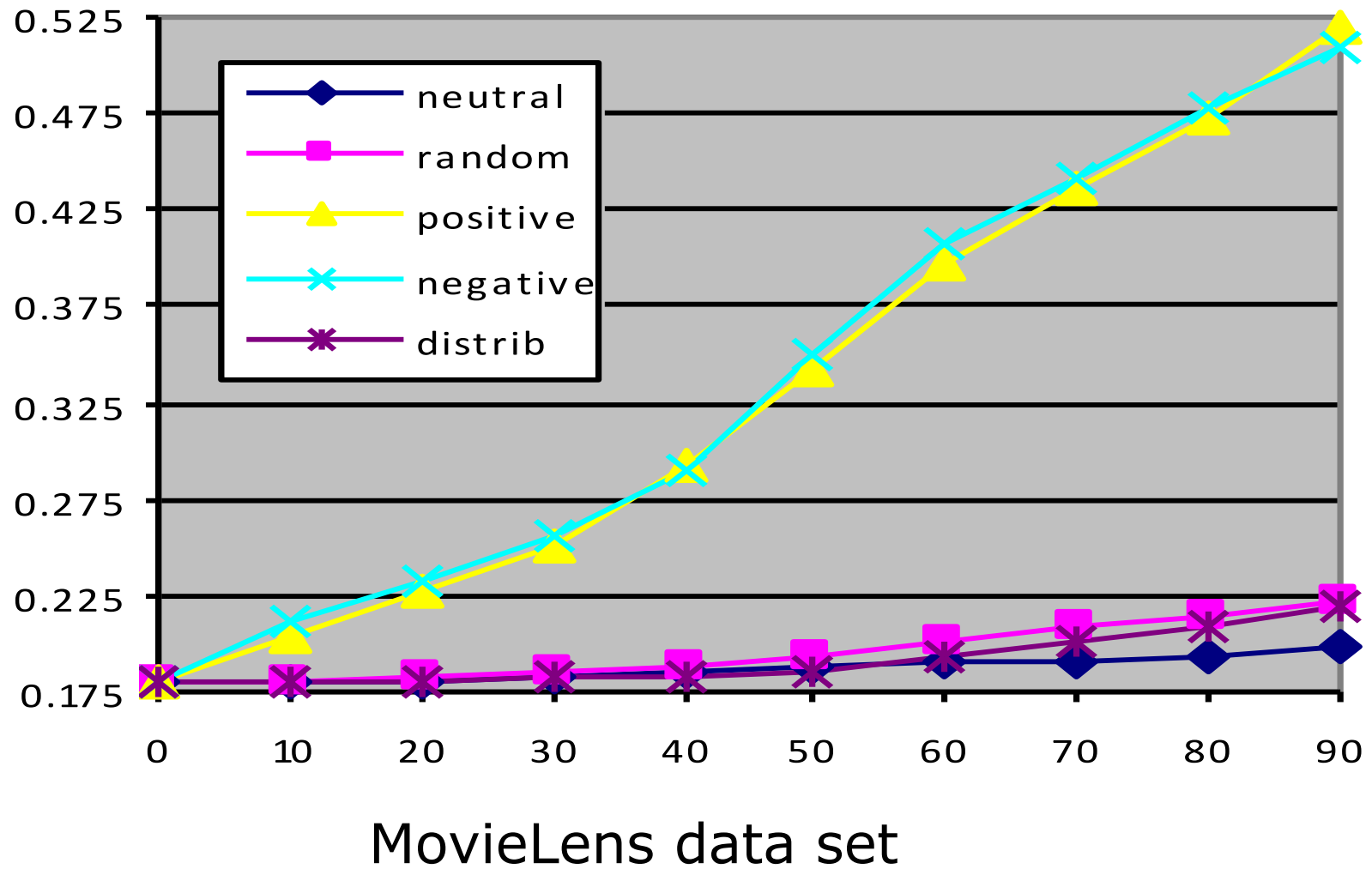


- ❑ The **difference** between the (normalized) MAE computed using only x% of the original data and the MAE with 100% of the data
- ❑ Surprisingly good results can be achieved even with a small subset of the data, **even if the data sets are already sparse!**
- ❑ **The point is that a smaller and smaller number of predictions is made, but on these predictions the system is correct.**

# Perturbation of the Data

- ❑ **Positive** - substitutes the real rating with the highest positive rating in the dataset (+10 for Jester, 5 for MovieLens and 1 EachMovie)
- ❑ **Negative** - substitutes the real rating with lowest negative rating in the dataset (-10 for Jester, 1 for MovieLens and 0 EachMovie)
- ❑ **Neutral** - substitutes the real rating with neutral rating, i.e., an average between the maximal and minimal possible ratings in the dataset (0 for Jester, 3 for MovieLens, and 0.5 for EachMovie)
- ❑ **Random** - substitutes the real rating with a random rating in the range of ratings in the respective dataset (between -10 to 10 for Jester, between 1 to 5 in MovieLens and between 0 to 1 in EachMovie)
- ❑ **Distribution** - substitutes the real rating with a rating reflecting the real distribution of ratings in the dataset (in terms of average and variance).

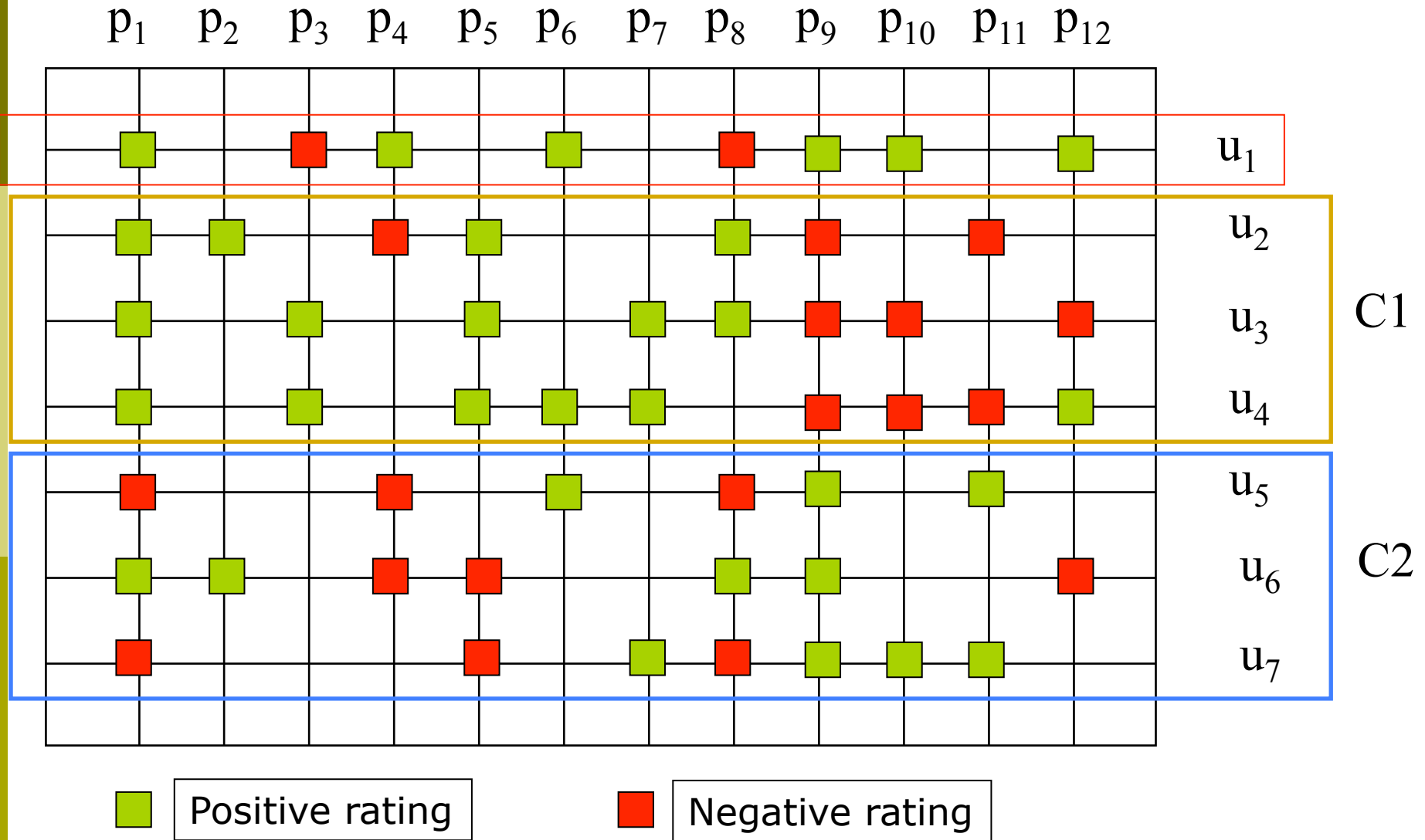
# Perturbate the Data



# User-Based Clustering and CF

- In **user-based clustering** users are clustered according to the similarity of their ratings (Pearson or Cosine)
- Each cluster center  $C_k$  is represented by the  $m$ -dimensional vector of  $C_k = (v_{1k}, \dots, v_{mk})$ , where  $v_{jk}$  is the average rating of the  $j$ -th product in the  $k$ -th cluster (Centroid)
- When a target user is considered the system can search for the cluster with the closest center and then base the prediction on the users in the cluster
- Various clustering methods have been used (K-means, EM, Gibbs Sampling) [Ungar & Foster, 1998]

# User-Based Cluster Example



	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12
u1	1		0	1		1		0	1	1		1
u2	1	1		0	1			1	0		0	
u3	1		1		1		1	1	0	0		0
u4	1		1		1	1	1		0	0	0	1
u5	0			0		1		0	1		1	
u6	1	1		0	0			1	1			0
u7	0				0		1	0	1	1	1	
C1	1	1	1	0	1	1	1	1	0	0	0	0,5
C2	0,3	1		0	0	1	1	0,3	1	1	1	0

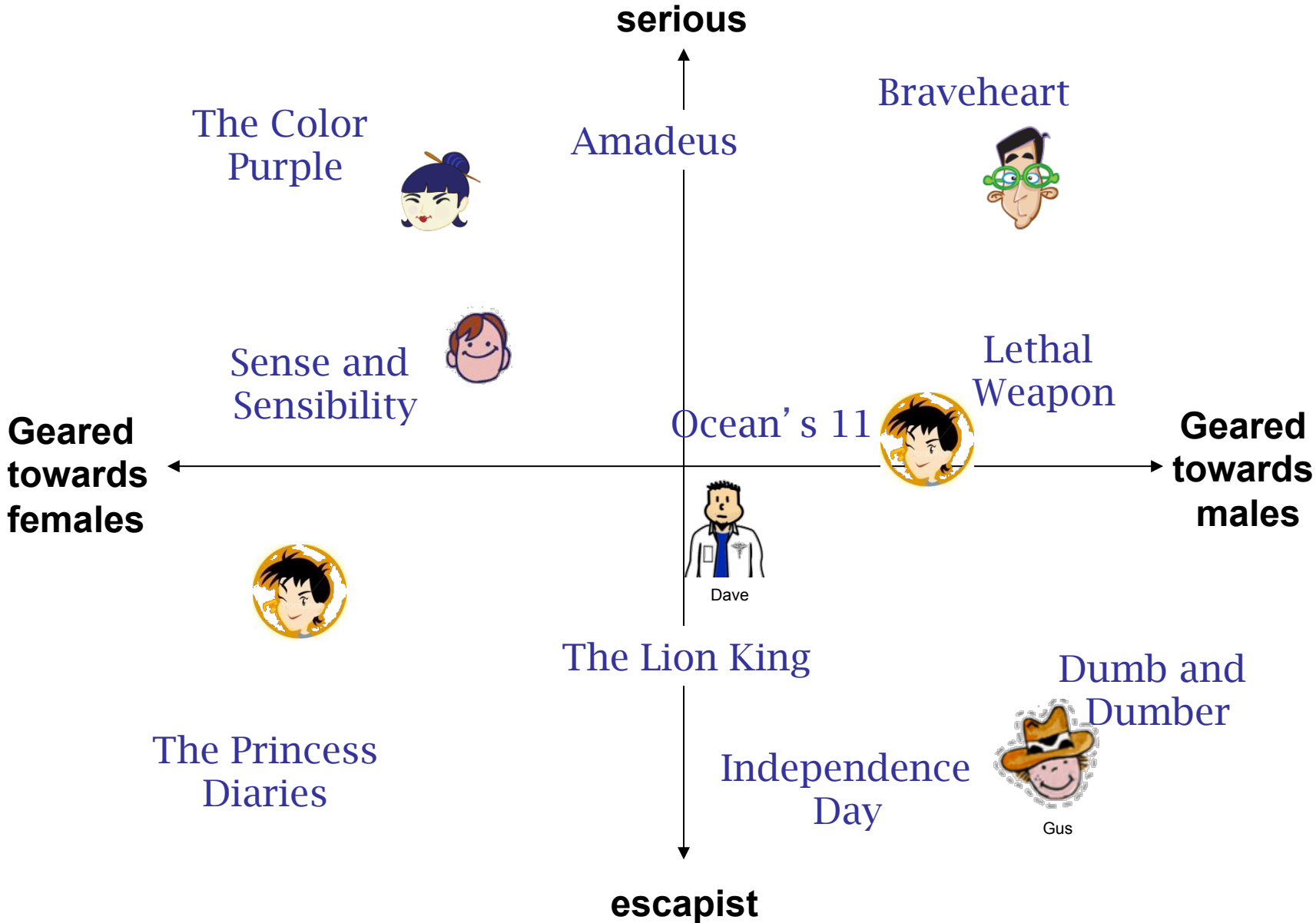
	u1	u2	u3	u4	u5	u6	u7	c1	c2
u1	1,00	-0,58	-0,71	-0,32	0,41	-0,41	0,58	-0,54	0,18
u2		1	1	1	-0,67	0,25	-1	1	-0,29
u3			1	0,73	-1	0,167	-0,71	0,942	-0,29
u4				1	-0,58	-0,58	-0,71	0,943	-0,59
u5					1	0,333	1	-0,33	0,962
u6						1	0,333	0,281	0,795
u7							1	-0,75	0,966
c1								1	-0,12
c2									1

Similarity (correlation)

# Item-Based Clustering and CF

- In **item-based clustering** the items are clustered according to their ratings' similarities
- There are two possible use
  - **Generalize** a product to its cluster and apply CF to the set of clusters – then recommend products in the most highly rated cluster (*Not applied yet?*)
  - **Decompose** the products in subsets corresponding to the clusters and when to make a recommendation for a product use only the ratings for products in the same cluster [O' Connor & Herlocker, 1999] (no improvement).

# Latent Factor Models



# Basic Matrix Factorization Model

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	

●

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

A rank-3 SVD approximation

Here we assume that 3 factors are sufficient

# Estimate Unknown Ratings

items

	1		3			5			5		4	
users			5	?		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
users	-.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

A rank-3 SVD approximation

# Estimate Unknown Ratings

users

1		3			5			5		4
		5	?		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

users

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

items

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

A rank-3 SVD approximation

# Estimate Unknown Ratings

$$-0.5*(-2) + 0.6*0.3 + 0.5*2.4 = 2.4$$

items

	1		3			5			5		4	
users			5	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

users	.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

A rank-3 SVD approximation

# Matrix Factorization Model

- Each item  $i$  and user  $u$  is associated with a  $f$ -dimensional real vector  $q_i$  and  $p_u$
- The elements of  $q_i = (q_{i1}, \dots, q_{if})$  measure the extent to which the item  $i$  possesses those factors, positive or negative
- The elements of  $p_u = (p_{u1}, \dots, p_{uf})$  measure the extent of interest  $u$  has in items that are high on the corresponding factors, positive or negative
  - $\hat{r}_{ui} = p_u^T q_i$  is the **predicted** user's overall interest in the item's characteristics
- **Problem:** how to compute  $q_i$  and  $p_u$ ?
- **Solution:** *try to determine  $q_i$  and  $p_u$  s.t. on known ratings the prediction is correct*
  - *Standard SVD (singular value decomposition) is undefined when knowledge about the matrix is incomplete*
  - *Carelessly addressing only the relatively few known entries is highly prone to overfitting.*

# Matrix factorization as a cost function

$$\text{Min}_{p^*, q^*} \sum_{\text{known } r_{ui}} \left( r_{ui} - p_u^T q_i \right)^2 + \lambda \left( \|p_u\|^2 + \|q_i\|^2 \right)$$

$p_u$  - user-factors of  $u$

$q_i$  - item-factors of  $i$

$r_{ui}$  - rating by  $u$  for  $i$

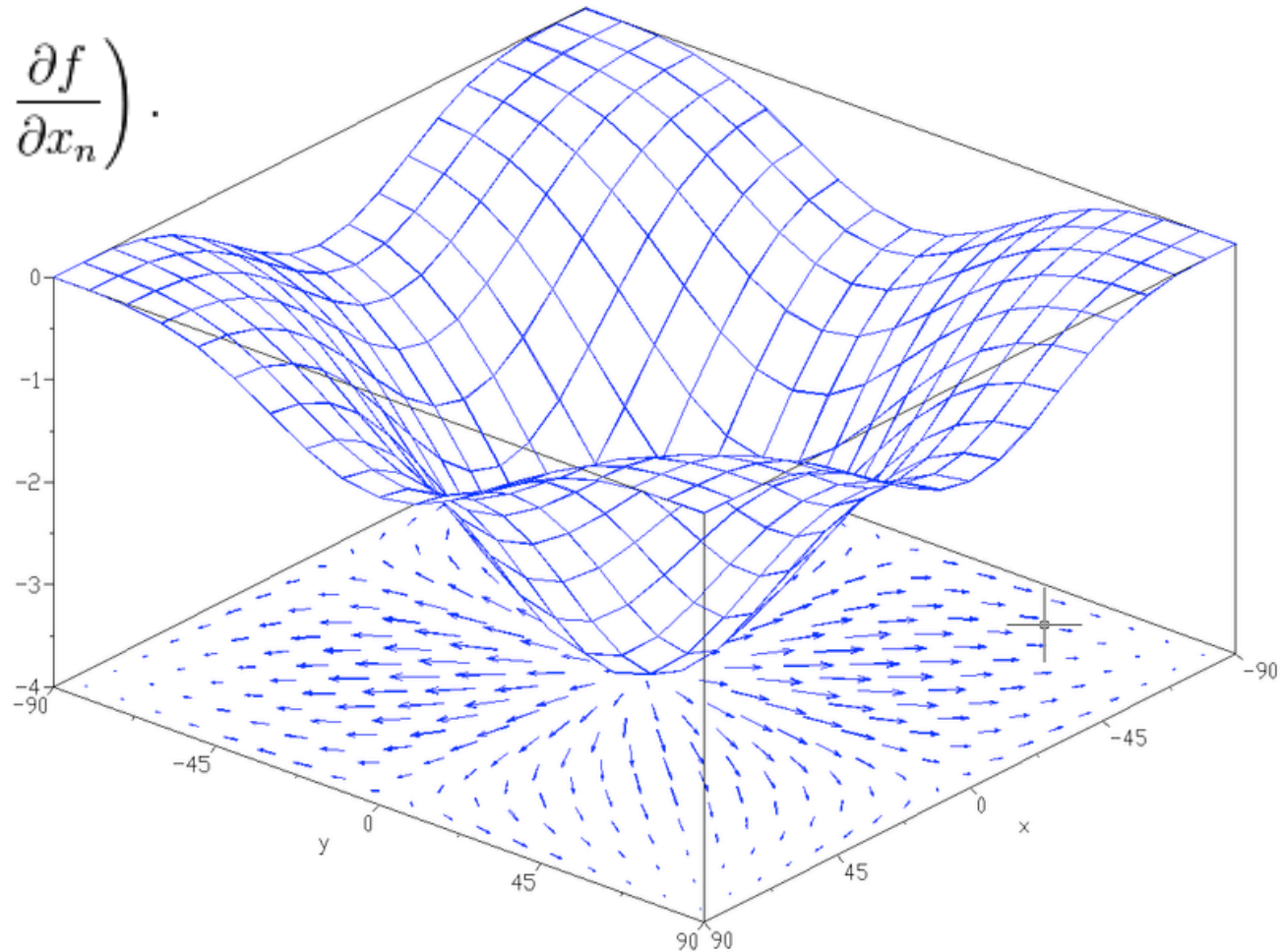
regularization

- Optimize by either **stochastic gradient-descent** or **alternating least squares**

Why this?  
What is happening  
if  $\lambda$  is large?

# Gradient of a Function

$$\nabla f = \left( \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right).$$



The gradient of the function  $f(x,y) = -(\cos^2x + \cos^2y)^2$  depicted as a vector field on the bottom plane

# Stochastic Gradient Descent

Perform till convergence:

- For each training example  $r_{ui}$  :
  - Compute prediction error:  $e_{ui} = r_{ui} - p_u^T q_i$
  - Update item factor:  $q_i \leftarrow q_i + \gamma(p_u e_{ui} - \lambda q_i)$
  - Update user factor:  $p_u \leftarrow p_u + \gamma(q_i e_{ui} - \lambda p_u)$
  
- The parameters are modified by a magnitude proportional to  $\gamma$  in the opposite direction of the gradient (of the function that we want to minimize)
- Two constants to tune:  $\gamma$  (step size) and  $\lambda$  (regularization)
- **The true goal is to find values that minimize error on **test** set.**

# Adding Biases

- **Much of the observed variation in rating values is due to effects associated to either users or items (individually)**
- Example: certain users give higher ratings and certain items are widely perceived as better
- First order approximation of the bias involved in rating  $r_{ui}$  is:
  - $b_{ui} = \mu + b_u + b_i$
  - Where  $\mu$  is the overall average rating
- **Example:** If  $\mu = 3.7$ , if Titanic is a movie that tends to be rated 0.5 better than an average movie, and Marius is a critical user who tends to rate 0.3 stars lower than the average
  - $b_{\text{marius, titanic}} = 3.7 - 0.3 + 0.5 = 3.9$

# Adding Biases

- The rating prediction function is now
  - $\hat{r}_{ui} = \mu + b_u + b_i + p_u^T q_i$
- And the corresponding new error function that we must minimize is:

$$\min_{p^*, q^*, b^*} \sum_{(u,i) \in K} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda (\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2)$$

- K is the set of known ratings

# Implicit Feedback and User Attributes

- Consider that the user  $u$  gave an **implicit feedback** to the items in  $N(u)$ , e.g., he bought them
- We can use an additional set of factor vectors, one for each item  $i$ ,  $x_i$  – expressing how much a user that showed an implicit feedback on  $i$  is loading the factors
- An additional component of the user model is then given by

$$|N(u)|^{-0.5} \sum_{i \in N(u)} x_i \quad \text{Implicit feedback}$$

- If we have also some user attributes  $A(u)$  (Boolean) that could be used to model the load of the different factors we have another component

$$\sum_{a \in A(u)} y_a \quad \text{User attributes}$$

## Adding Implicit Feedback and Attributes

- The rating prediction function is now:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T \left[ p_u + |N(u)|^{-0.5} \sum_{i \in N(u)} x_i + \sum_{a \in A(u)} y_a \right]$$

- Gradient descent can still be applied to minimize the corresponding error function.

# Temporal Dynamics and Confidence

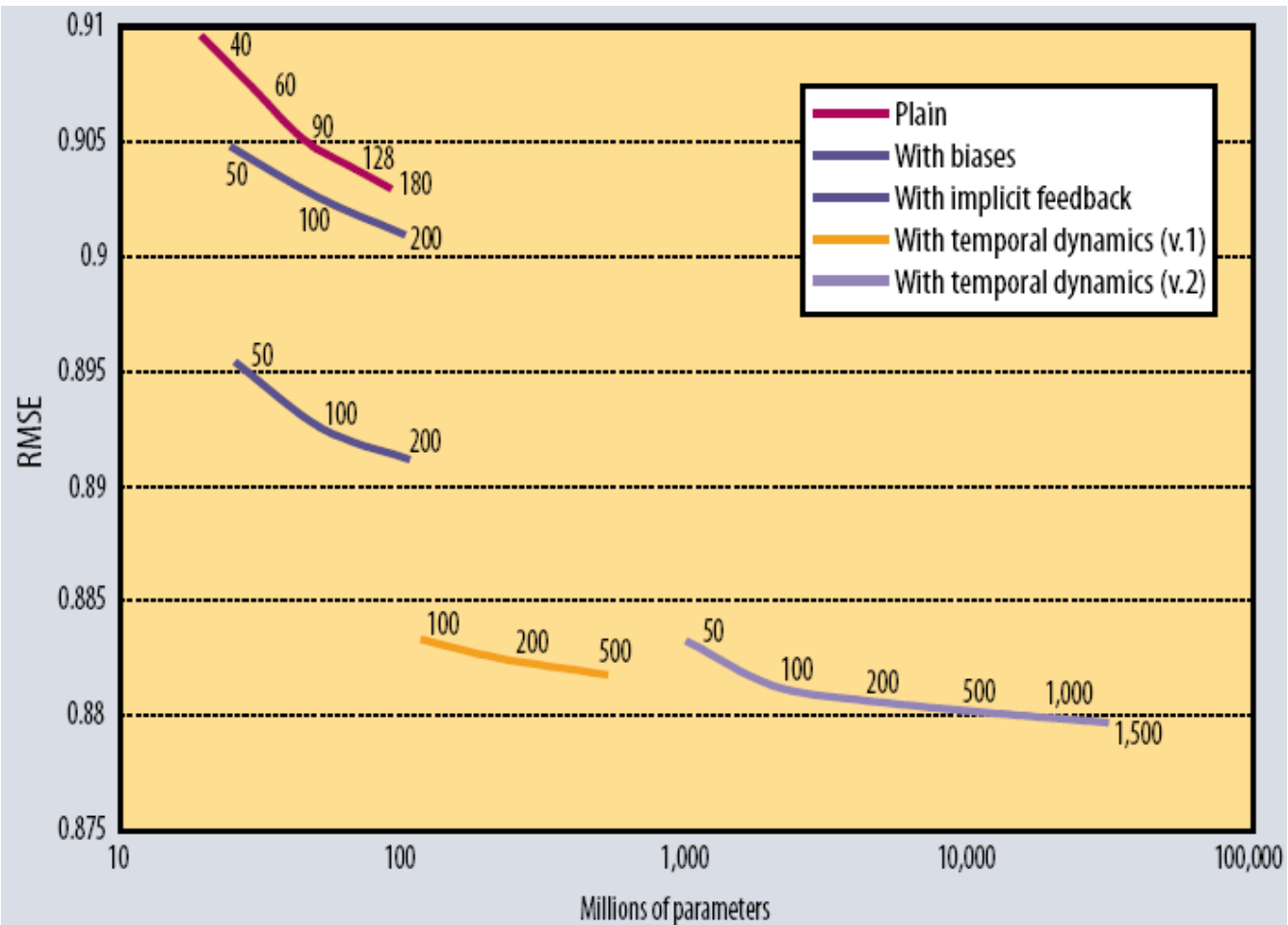
- **Ratings can change with time** because users change the way they rate or items change their relevance

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

- Some ratings can be more reliable than others
  - E.g. you may infer that ratings too old, or ratings produced after a massive advertisement, are not much reliable
- This can be easily incorporated in the prediction model:

$$\min_{p^*, q^*, b^*} \sum_{(u,i) \in K} c_{ui} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda (\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2)$$

$c_{ui}$  is the reliability of rating  $r_{ui}$



**Figure 4. Matrix factorization models' accuracy.** The plots show the root-mean-square error of each of four individual factor models (lower is better). Accuracy improves when the factor model's dimensionality (denoted by numbers on the charts) increases. In addition, the more refined factor models, whose descriptions involve more distinct sets of parameters, are more accurate. For comparison, the Netflix system achieves  $RMSE = 0.9514$  on the same dataset, while the grand prize's required accuracy is  $RMSE = 0.8563$ .

# Collaborative-Based Filtering

- **Pros:** require minimal knowledge engineering efforts (knowledge poor)
  - Users and products are symbols without any internal structure or characteristics
- **Cons:**
  - Requires a large number of **explicit** and **reliable** “rates” to bootstrap
  - Requires products to be standardized (users should have bought **exactly** the same product)
  - Assumes that **prior behavior determines current behavior** without taking into account “contextual” knowledge (session-level)
  - Does not provide information about products or explanations for the recommendations
  - Does not support sequential decision making or recommendation of “good bundling”, e.g., a travel package.

## Summary

- Introduced Item-to-item collaborative recommendations (the method that is really used now in MovieLens and Amazon)
- Discussed its advantages on user-to-user collaborative filtering
- Shown the effects of sparsity and perturbation of the data
- Presented the more recent approaches based on factor models.

# Questions

- ❑ Why item-to-item collaborative filtering is preferred to user-to-user collaborative filtering?
- ❑ What are the methods used for computing the similarity of products?
- ❑ Could you imagine other similarity methods for products?
- ❑ What is the user model in a item-to-item collaborative filtering system?
- ❑ Why clustering is important in collaborative filtering?
- ❑ What kind of clusters could be used in CF?
- ❑ How many factors should be used in matrix factorization techniques?
- ❑ What is the role of the  $\lambda$  parameter in matrix factorization techniques?