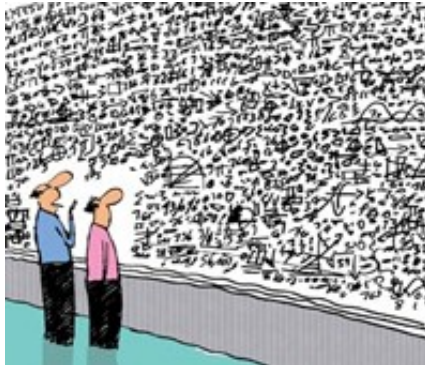


From Collaborative Filtering to Recommendation Systems

SSC12 - 12th Scandinavian Symposium on Chemometrics



Michael Jahrer
commendo research & consulting
Köflach, 2011



Table of contents

- **My background**
- **The collaborative filtering problem**
- **Types of data**
- **Error metrics**
- **Three Algorithms to approximate the sparse matrix**
- **Recommender systems**
- **Possible applications**

My background

- **University of Technology Graz (AUSTRIA) 2003-2010**
 - Since 2006 focused in machine learning
 - 2010 MSc
- **2008: Co-founder of „commendo research & consulting“**
- **2009: Co-Winner of the Netflix 1M \$ Price**
- **Now: Developer/Researcher/Programmer @ commendo**
 - Collaborative Filtering
 - Recommender Systems
 - Machine Learning in general

The collaborative filtering problem

- **Users make actions on items**

- **Users can be..**

- Humans
- Other processes

users					items
	x				
			x		
x					

x ... event (or action)

- **Items can be..**

- Products
- Movies
- Chemical substances
- Other things

- **We want..**

- To make predictions for events on unknown user/item pairs

Data types

■ Data in a collaborative filtering setup

users

	u0	u1	u2	u3	u4	u5	u6	u7	u8	u9	u10	u11	u12	u13	u14	u15	u16	
								3				1						i0
	1				1				1						1			i1
		1				1				4			1					i2
							5										1	i3
	5		1	1	1						1	5						i4
							1		1			1			4			i5
	1					1		2										i6
				1					1								1	i7
		1										1		1				i8

items

1...Purchase

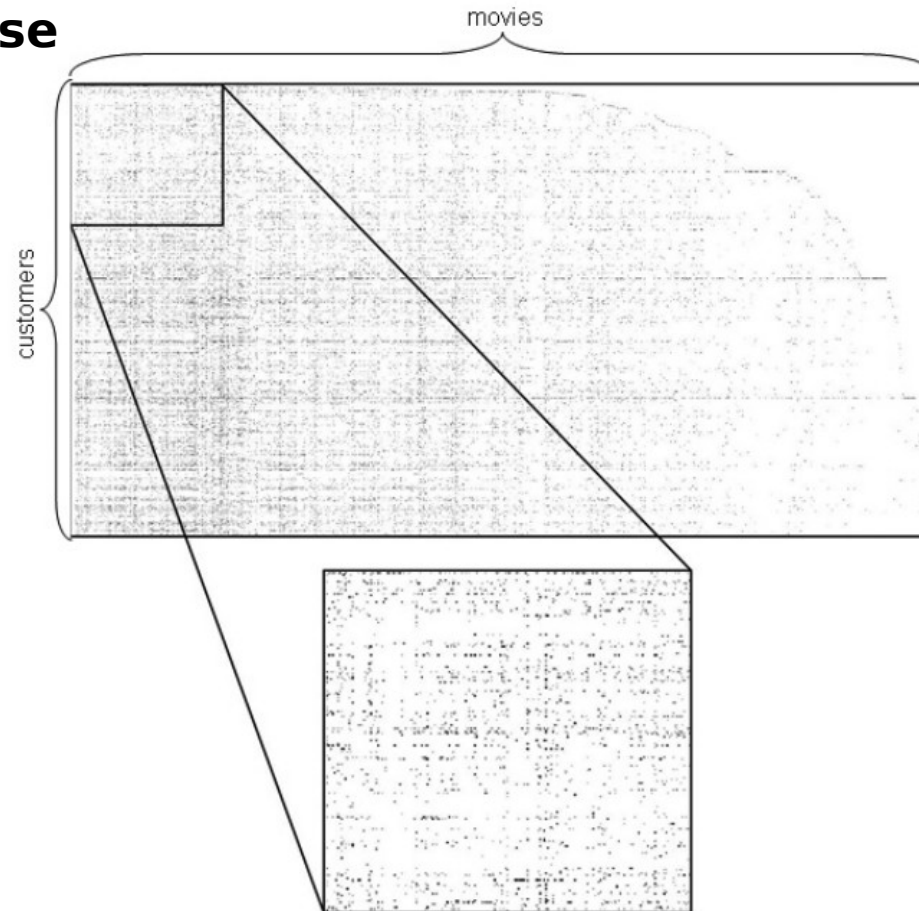
1...Add to wishlist

1...5 Rating (5-star)

events data

Matrix representation

- Large and sparse



Predicting Missing Ratings in Recommender Systems: Adapted Factorization Approach
(<http://www.cvc.uab.es/uploads/publicacions/F130949.pdf>)

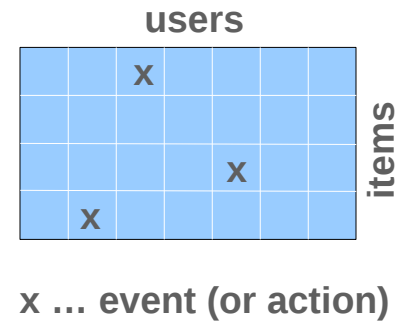
Error Function

■ Prediction accuracy

- RMSE (root mean square error)
- MAE (mean average error)

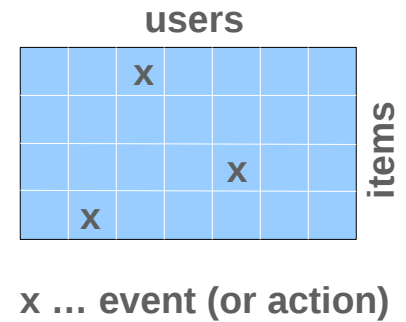
■ Rank-based

- Top-K hitrate
- AUC (area under curve)
- Precision / Recall



Three Algorithms to approximate the sparse matrix

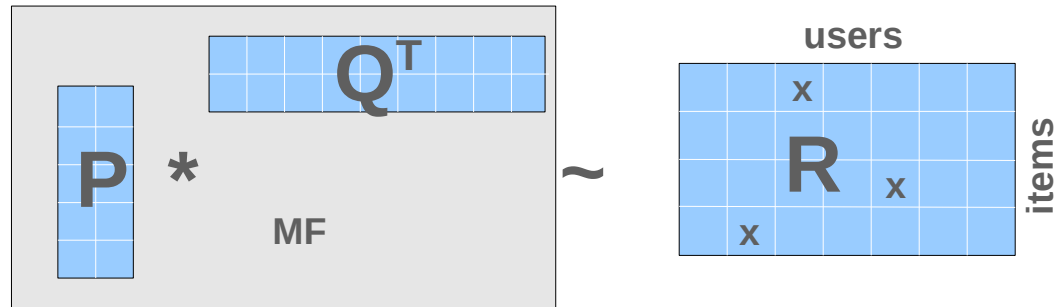
- (1) Matrix Factorization
- (2) Asymmetric Factor Model
- (3) K-Nearest Neighbors



(1) Matrix factorization

■ **R = matrix**

- rows = items
- columns = users
- $R = [\text{items} \times \text{users}]$



■ **We want to factorize $R = P * Q^T$**

- $P = \text{item feature matrix } [\text{items} \times F]$
- $Q = \text{user feature matrix } [\text{users} \times F]$

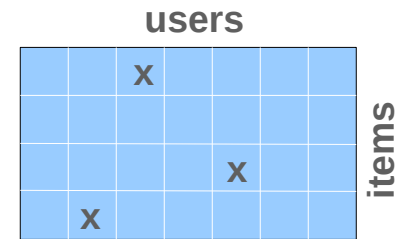
■ **Represent users and items in a low-dimensional space**

(1) Matrix factorization

■ **Prediction** $\hat{r}_{ui} = p_i \cdot q_u^T$ ← dot product

■ **Quadratic error over a list of ratings** $\{r_{ui}\}$

$$RMSE = \sqrt{\frac{1}{|R|} \sum_{u, i \in R} (\hat{r}_{ui} - r_{ui})^2}$$



x ... event (or action)

■ **Learning: Stochastic gradient descent**

$$ERR_{ui} = (p_i \cdot q_u^T - r_{ui})^2$$
 ← error on R(u,i)

$$\frac{\partial ERR_{ui}}{\partial p_{if}} = 2 \cdot (p_i \cdot q_u^T - r_{ui})^2 \cdot q_{uf}$$
 ← gradient for item features

$$\frac{\partial ERR_{ui}}{\partial q_{uf}} = 2 \cdot (p_i \cdot q_u^T - r_{ui})^2 \cdot p_{if}$$
 ← gradient for user features

$$p_{if} = p_{if} - lRate \cdot \frac{\partial ERR_{ui}}{\partial p_{if}}$$
 ← learning rule for item features

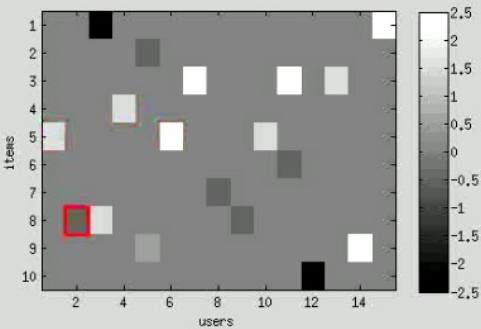
(1) Matrix factorization

- Training on random data

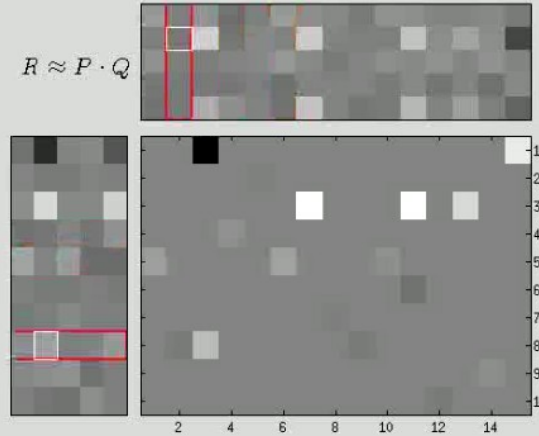
(1) Matrix factorization

■ Training on random data

R (values: -2.5...2.5)



$R \approx P \cdot Q$



[TRAIN ALGORITHM] P,Q init randomly
e:3 $RMSE_{train} = 1.5634$

```
for idx=1...nRatings
    user = R(idx).user           2
    item = R(idx).item          8
    target = R(idx).rating      -0.5
    prediction = P(item,:)*Q(user,:)T  -0.08634
    error = prediction - target  0.41366
    for f=1...F
        x = P(item,f)
        P(item,f) = P(item,f) -  $\eta$  * error * Q(user,f)
        Q(item,f) = Q(item,f) -  $\eta$  * error * x
    end
end
```

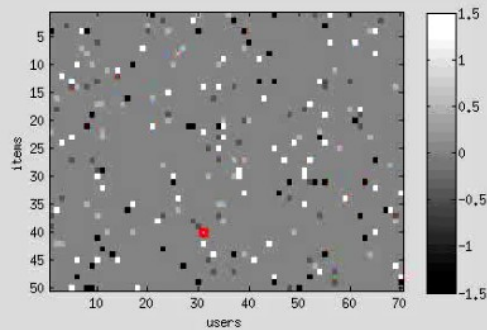
(1) Matrix factorization

- Training on larger random data

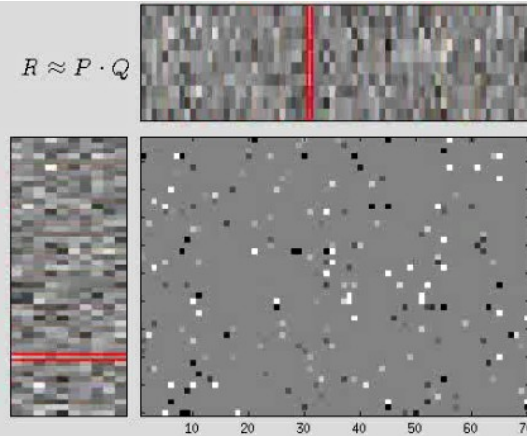
(1) Matrix factorization

■ Training on larger random data

R (values: -1.5...1.5)



$R \approx P \cdot Q$



[TRAIN ALGORITHM] P,Q init randomly
e:4 $RMSE_{train}=0.60879$

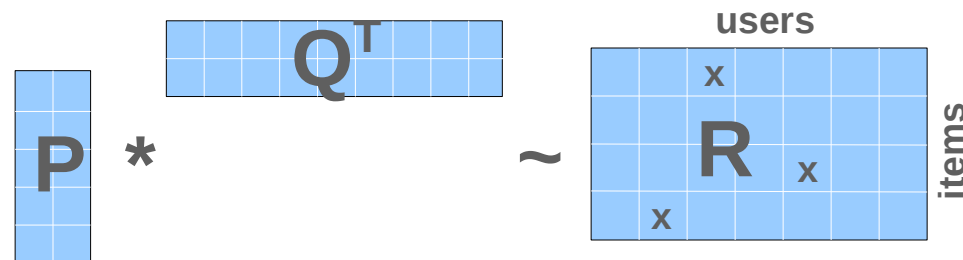
```
for idx=1...nRatings
  user = R(idx).user           31
  item = R(idx).item          40
  target = R(idx).rating      0.5
  prediction = P(item,:)*Q(user,:)T  0.40383
  error = prediction - target  -0.096167
  for f=1...F
    x = P(item,f)
    P(item,f) = P(item,f) -  $\eta$  * error * Q(user,f)
    Q(item,f) = Q(item,f) -  $\eta$  * error * x
  end
end
```

(1) Matrix factorization

■ Interesting Facts

- ▢ Users with similar rating profile have similar feature vectors
- ▢ Items with similar rating profile have similar feature vectors
- ▢ Users can be similar without any overlapping rated items
- ▢ Learns from all the data simultaneously (stochastic GD)

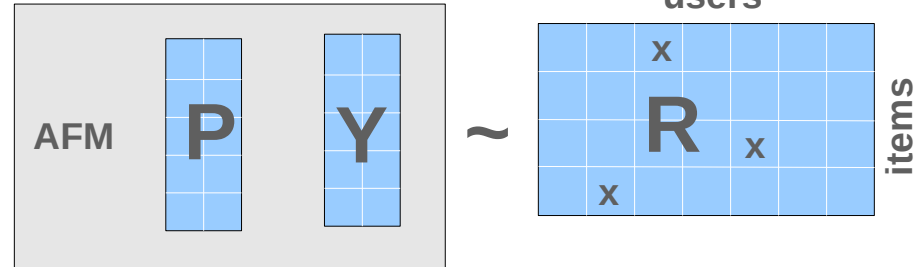
- ▢ Train time: $O(N)$ $N = \# \text{ratings}$
- ▢ Mem consumption: $O(\# \text{user} + \# \text{items})$
- ▢ Prediction time: $O(1)$



(2) Asymmetric Factor Model

■ Similar to matrix factorization

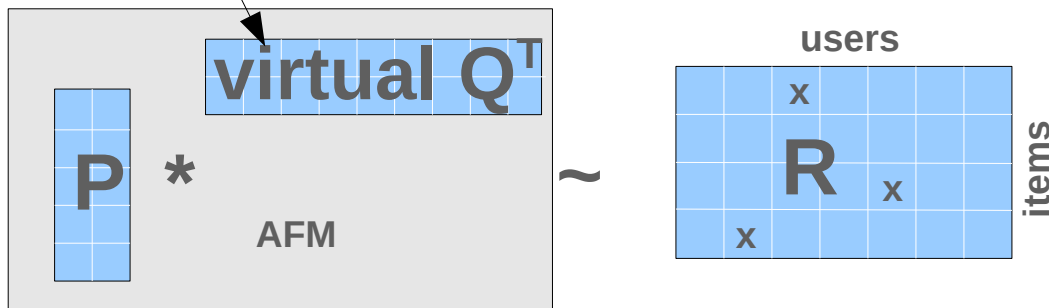
- Two item feature matrices: P, Y
- User = represented as „bag of items“
- Y = item features (user side)



$$q_u = \sqrt{\frac{1}{|R(u)|}} \cdot \sum_{i \in R(u)} y_i \quad \leftarrow \text{the „virtual“ user feature}$$

$R(u)$ \leftarrow list of rated items of u

$$\hat{r}_{ui} = p_i \cdot q_u^T \quad \leftarrow \text{prediction}$$



(2) Asymmetric Factor Model

■ The „virtual user feature“

$$q_u = \sqrt{\frac{1}{|R(u)|}} \cdot \sum_{i \in R(u)} y_i$$

$$q_0 = \sqrt{\frac{1}{2}} \left(y_0 + y_2 \right) \leftarrow \text{virtual user feature 0 = bag of } i_0 \text{ and } i_1$$

	u0	users			
i0	x				items
i1					
i2	x		R	x	
i3		x			

This is used for normalization

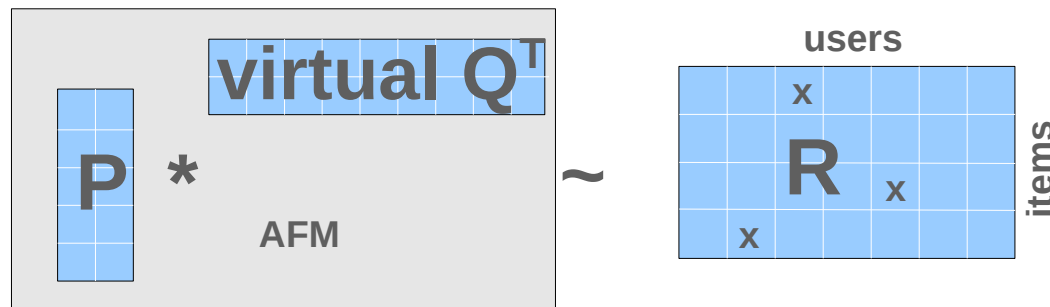
Assumption: values in Y are normal distributed

(2) Asymmetric Factor Model

■ Interesting Facts

- New users can be handled without re-training of the whole model
- The idea can also be applied on item side (item=bag of users)

- Train time: $O(N)$ $N = \# \text{ratings}$
- Mem consumption: $O(\# \text{items})$
- Prediction time: $O(1)$ if virtual user feature is precomputed



(3) K-Nearest Neighbors

- Based on item-item or user-user similarity
- Higher similarity indicates similar properties
- $\text{similarity}(u_0, u_1) = ? \rightarrow 2 \text{ overlaps}$
- $\text{similarity}(i_2, i_3) = ? \rightarrow 3 \text{ overlaps}$

		users																	
		u0	u1	u2	u3	u4	u5	u6	u7	u8	u9	u10	u11	u12	u13	u14	u15	u16	
items	i0	1	1												1				
	i1			1		1				1			1						
	i2		1		1		1		1		1			1	1				
	i3		1					1		1				1	1				1
	i4			1	1	1							1	1					
	i5		1						1		1			1					
	i6		1		1														
	i7					1					1			1					1
	i8		1												1				

1 ... Purchase

(3) K-Nearest Neighbors

■ Prediction is a weighted sum (item-item similarity)

$$\hat{r}_{ui} = \sum_{j \in R(u)} r_{uj} \cdot \text{similarity}(i, j)$$

$\text{similarity}(i, j)$ = % of users overlapping
 $R(u)$ = list of rated items of u

$$\hat{r}_{62} = 1 \cdot \frac{3}{17} + 1 \cdot \frac{1}{17} = 0.235$$

users

	u0	u1	u2	u3	u4	u5	u6	u7	u8	u9	u10	u11	u12	u13	u14	u15	u16	
i0	1	1												1				
i1			1		1				1			1						
i2		1		1		1	?	1		1			1	1				
i3		1					1		1				1	1			1	
i4			1	1	1						1	1						
i5	1	1					1		1			1						
i6	1		1															
i7				1					1			1					1	
i8	1													1				

items

1 ... Purchase

commendo

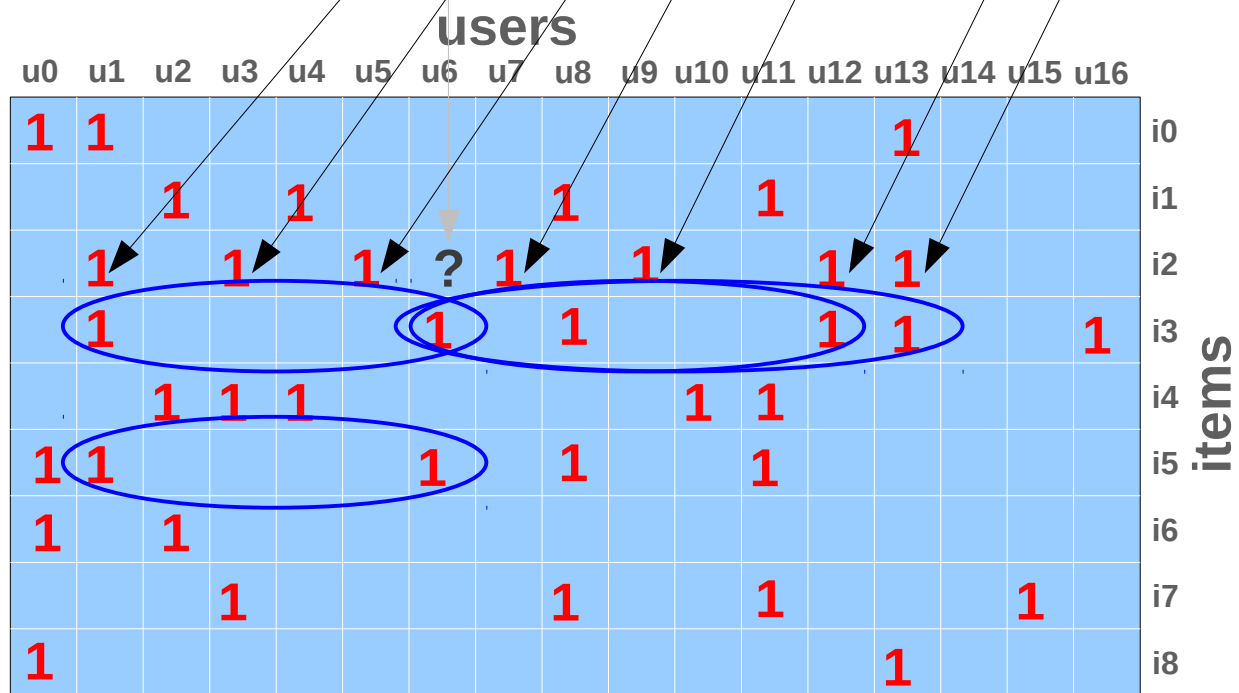
(3) K-Nearest Neighbors

■ Prediction is a weighted sum (user-user similarity)

$$\hat{r}_{ui} = \sum_{v \in R(i)} r_{vi} \cdot \text{similarity}(u, v)$$

$\text{similarity}(u, v) = \%$ of items overlapping
 $R(i)$ = list of users, rated i

$$\hat{r}_{62} = 1 \cdot \frac{2}{9} + 1 \cdot \frac{0}{9} + 1 \cdot \frac{0}{9} + 1 \cdot \frac{0}{9} + 1 \cdot \frac{0}{9} + 1 \cdot \frac{0}{9} + 1 \cdot \frac{1}{9} + 1 \cdot \frac{1}{9} = 0.444$$



1 ... Purchase



(3) K-Nearest Neighbors

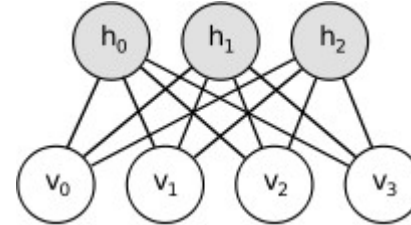
■ Interesting facts

- No training needed (in online mode)
- Predictions are explainable (because you bought xyz..)
- Accuracy is limited if the data matrix is sparse
 - Number of overlaps (user-user or item-item) is typically small

(?) Other models

■ RBM - Restricted Boltzmann machine

- Is a neural network
- #Visible units = #items
- #Hidden units = #features
- Trained with „contrastive divergence“



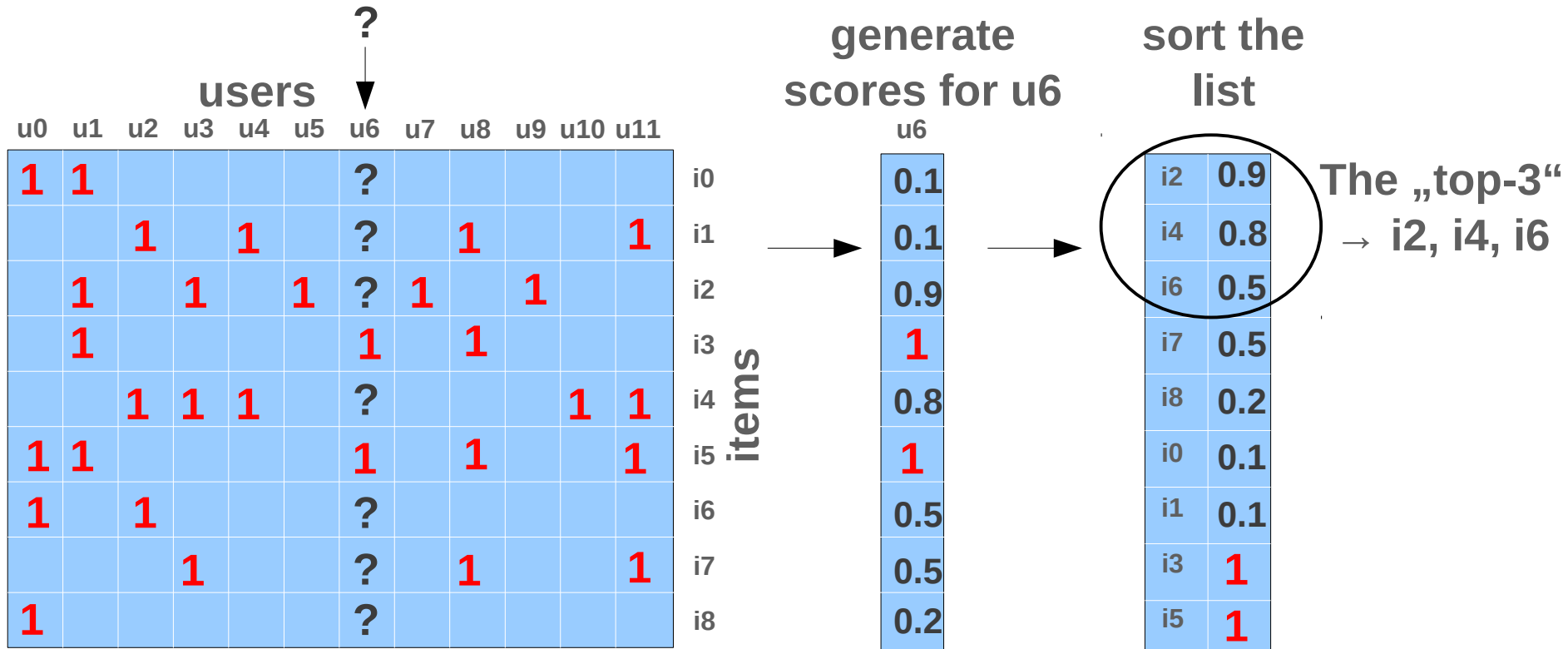
■ Global effects

- Data cleaning
- User/item offsets
- Time effects

■ Clustering

Recommender system

- Which items should be recommended to the user?
- top-K recommendations



Possible applications (beyond recommender systems)

■ Personalized medicine

- Users ↔ patients
- Items ↔ medicine
- Ratings ↔ impact of this medicine on this patient
- Prediction: which medicine does the patient need

■ Health care

- Users ↔ patients
- Items ↔ claims (hospitalizations, surgeries, used drugs)
- Ratings ↔ binary indicators (0/1) of the patient/claim pair
- Prediction: identify similar users

■ Pharmaceutical - drug design

- Users ↔ effects (e.g. docking property, toxicity, characteristics)
- Items ↔ substances (e.g. proteins, vitamins)
- Ratings ↔ how large is the effect of this substance
- Prediction: unknown effects of substances

	eff 1	eff 2	eff 3	eff 4	eff 5	eff 6	eff 7	eff 8	eff 9
sub 1	0.3				0.7			0.9	
sub 2		0.2		0.1		0.6			
sub 3					0.1				0.2
sub 4	0.8		0.6					0.5	
sub 5				0.2			0.4		

Active Substance ↔ Effect
Rating scale [0..1] for probability of effect.
0.0: never
0.1: hardly ever
0.4: frequently
0.6: more often
0.9: almost all cases
1.0: full applicable

Thank you!

commendo research & consulting GmbH

www.commendo.at