Index

Index

From Search to Recommendation

"The Web is leaving the era of search and entering one of discovery. What's the difference?

***Search** is what you do when you're looking for something. **Discovery** is when something wonderful that you didn't know existed, or didn't know how to ask for, finds you." – CNN Money, "The race to create a 'smart' Google*

The value of recommendations

- Netflix: 2/3 of the movies watched are recommended
- Google News: recommendations generate 38% more click-throughs
- Amazon: 35% sales from recommendations
- Choicestream: 28% of the people would buy more music if they found what they liked.

The "Recommender problem"

Estimate a **utility function**
to **predict** how
a user will **like** an item.

The "Recommender problem"

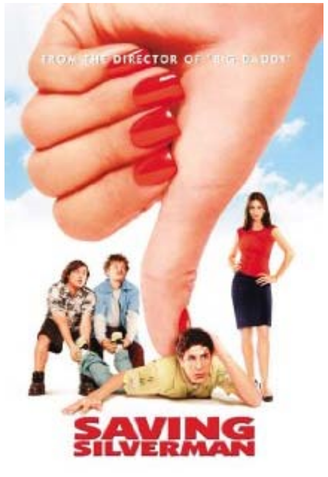
- C:= {users}
- S:= {recommendable items}
- u:= utility function, measures the usefulness of item s to user c,

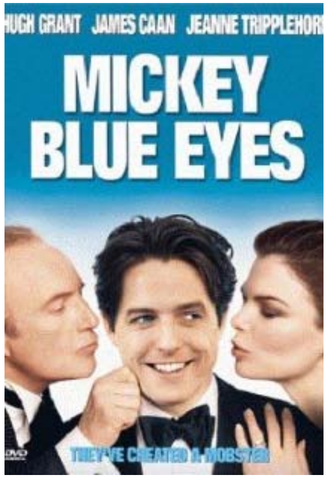
  $$u : C \times S \rightarrow R$$

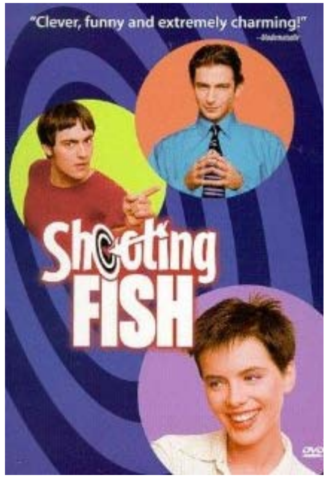
  where R:= {recommended items}.
- For each user c, we want to choose the items s that maximize u.

$$c \in C \quad s'_c = argmax_u u(c, s)$$

A good recommendation







is relevant to the user: *personalized*

A good recommendation

- is diverse:

  it represents all the possible interests of one user

A good recommendation

- Does not recommend items the user already knows or would have found anyway.
- Expands the user's taste into neighboring areas.

Serendipity* = *Unsought finding

Top k recommendations

Users take into account only few suggestions. There is a need to do better on the top scoring recommended items

Index

1. Introduction: What is a Recommender System?
2. Approaches
    1. Collaborative Filtering
    2. Content-based Recommendations
    3. Context-aware Recommendations
    4. Other Approaches
    5. Hybrid Recommender Systems
3. Research Directions
4. Conclusions
5. References

What works?

- Depends on the domain and particular problem

- Currently, the best approach is Collaborative Filtering.
- Other approaches can be combined to improve results

- What matters?
  - Data preprocessing: outlier removal, denoising, removal of global effects
  - "Smart" dimensionality reduction
  - Combining methods

Collaborative Filtering

The task of **predicting** (filtering) user preferences on new items by **collecting** taste information from many users (collaborative).

Challenges:

- many items to choose from
- very few recommendations to propose
- few data per user
- no data for new user

Index

1. Introduction: What is a Recommender System?
2. Approaches
    1. Collaborative Filtering:
        1. Memory-based CF
            1. User-based CF
            2. Item-based CF
        2. Model-based CF

Memory-Based CF:
User-based CF & Item-based CF

Example

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

Each user has expressed an opinion for some items:

- Explicit opinion: rating score
- Implicit: purchase records or listen to tracks

Example: User-based CF

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

Target (or Active) user for whom the CF recommendation task is performed

Example: User-based CF

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

1. Identify set of items rated by the target user

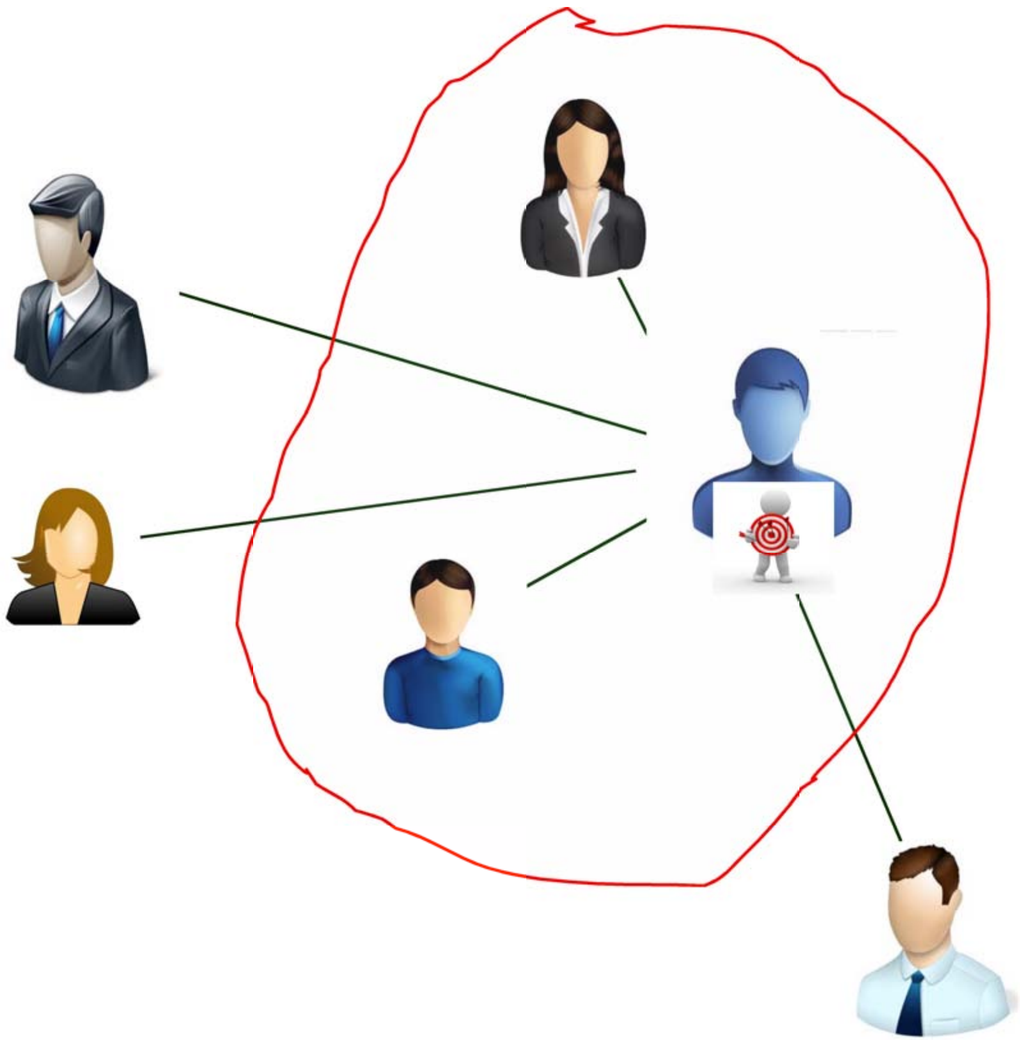
Example: User-based CF

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

1. Identify set of items rated by the target user

2. Identify which other users rated 1+ items in this set (neighborhood formation)

User-based Similarity



3. Compute how similar each neighbor is to the target user (similarity function)

4. In case, select k most similar neighbors

User-based CF

5. Predict ratings for the target user's unrated items (prediction function)

6. Recommend to the target user the top N products based on the predicted ratings

User-based CF

- Target user u, ratings matrix Y
- Yv,i → rating by user v for item i
- Similarity Pearson r correlation sim(u,v) between users u & v

$$sim(u,v) = \frac{\sum_{i \in I_{uv}} (y_{u,i} - \hat{y}_u)(y_{v,i} - \hat{y}_v)}{\sqrt{\sum_{i \in I_{uv}} (y_{u,i} - \hat{y}_u)^2 \sum_{i \in I_{uv}} (y_{v,i} - \hat{y}_v)^2}}$$

- Predicted rating $y^*(u,i)$

$$y^*(u,i) = \hat{y}_u + \frac{\sum_{j \in I_{y_{*j} \neq 0}} sim(v_j,u)(y_{v_j,i} - \hat{y}_{v_j})}{\sum_{j \in I_{y_{*j} \neq 0}} |sim(v_j,u)|}$$

Example: User-based CF

						sim(u,v)
2			4	5		NA
5		4			1	
		5		2		
	1		5		4	
		4			2	
4	5		1			NA

Example: User-based CF

						sim(u,v)
2			4	5		NA
5		4			1	0.87
		5		2		
	1		5		4	
		4			2	
4	5		1			NA

Example: User-based CF

						sim(u,v)
2			4	5		NA
5		4			1	0.87
		5		2		1
	1		5		4	
		4			2	
4	5		1			NA

Example: User-based CF

						sim(u,v)
2			4	5		NA
5		4			1	0.87
		5		2		1
	1		5		4	-1
		4			2	
4	5		1			NA

Example: User-based CF

						sim(u,v)
2			4	5		NA
5		4			1	0.87
		5		2		1
	1		5		4	-1
3.51*	3.81*	4	2.42*	2.48*	2	
4	5		1			NA

Example: Item-based CF

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

Target item: item for which the CF prediction task is performed.

Item-based CF

The basic steps:

- Identify set of users who rated the target item i
- Identify which other items (neighbours) were rated by the users set
- Compute similarity between each neighbour & target item (similarity function)
- In case, select k most similar neighbours
- Predict ratings for the target item (prediction function)

Item Based Similarity

Item Based Similarity

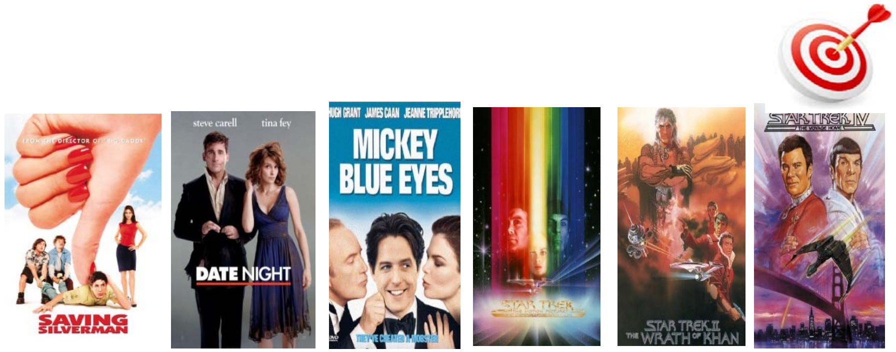
- Target item I
- Yu,j → rating of user u for item j $\hat{y}_j$ average rating for j.
- Similarity sim(i,j) between items i and j (Pearson-correlation)

$$sim(i,j) = \frac{\sum_{u \in I_{ij}} (y_{u,i} - \hat{y}_i)(y_{u,j} - \hat{y}_j)}{\sqrt{\sum_{u \in I_{ij}} (y_{u,i} - \hat{y}_i)^2 \sum_{u \in I_{ij}} (y_{u,j} - \hat{y}_j)^2}}$$

- Predicted rating $y^*(u,i)$

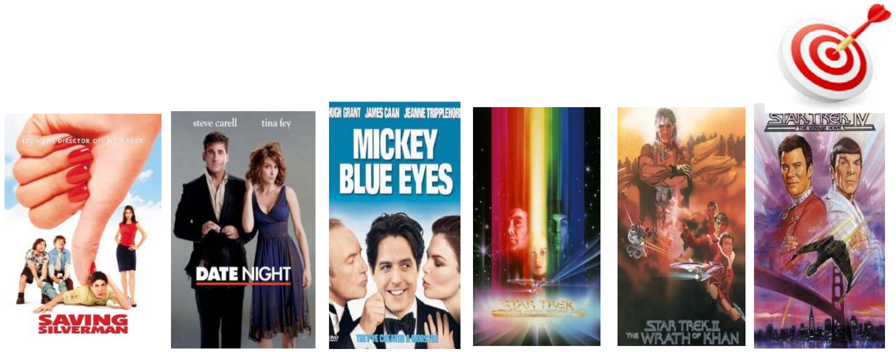
$$y^*(u,i) = \hat{y}_i + \frac{\sum_{v \in I_{y_{u*} \neq 0}} sim(i,j_v)(y_{u,j_v} - \hat{y}_{j_v})}{\sum_{v \in I_{y_{u*} \neq 0}} |sim(i,j_u)|}$$

Example: Item-based CF



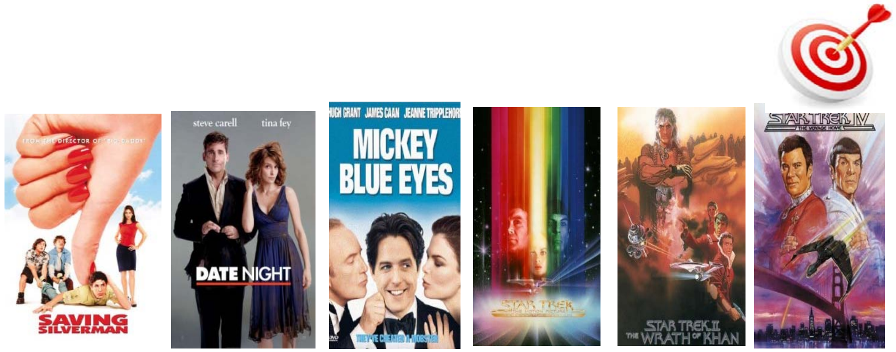
2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

Example: Item-based CF



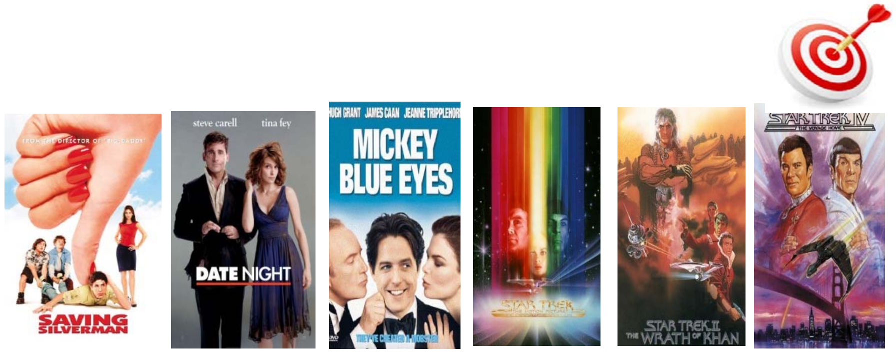
2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

Example: Item-based CF



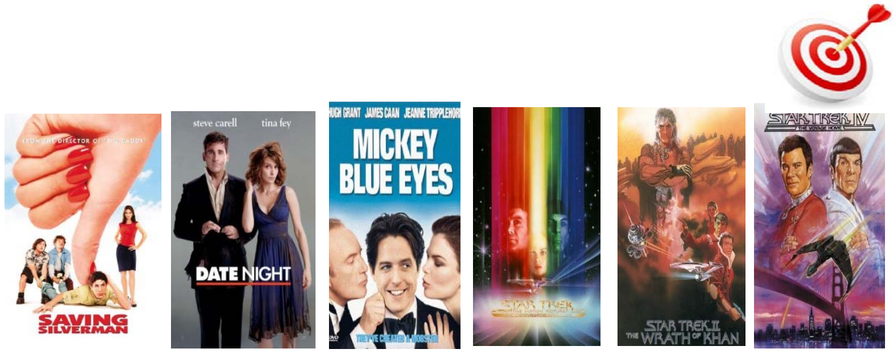
2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

Example: Item-based CF



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

Example: Item-based CF



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

sim(6,5) cannot be calculated

Example: Item-based CF

2			4	5	2.94*
5		4			1
		5		2	2.48*
	1		5		4
		4			2
4	5		1		1.12*

Item Similarity Computation

- **Pearson *r* correlation-based *Similarity***

  does *not* account for user rating biases

- **Cosine-based *Similarity***

  does *not* account for user rating biases

$$cos(i,j) = \frac{\sum_{u \in I_{ij}} y_{u,i} y_{u,j}}{\sqrt{\sum_{u \in I_{ij}} y_{u,i}^2 \sum_{u \in I_{ij}} y_{u,j}^2}}$$

- **Adjusted *Cosine Similarity***

  takes care of user rating biases as each pair in the co-rated set corresponds to a different user.

$$sim(i,j) = \frac{\sum_{u \in I_{ij}} (y_{u,i} - \hat{y}_u)(y_{u,j} - \hat{y}_u)}{\sqrt{\sum_{u \in I_{ij}} (y_{u,i} - \hat{y}_u)^2 \sum_{j \in I_{uu}} (y_{u,j} - \hat{y}_u)^2}}$$

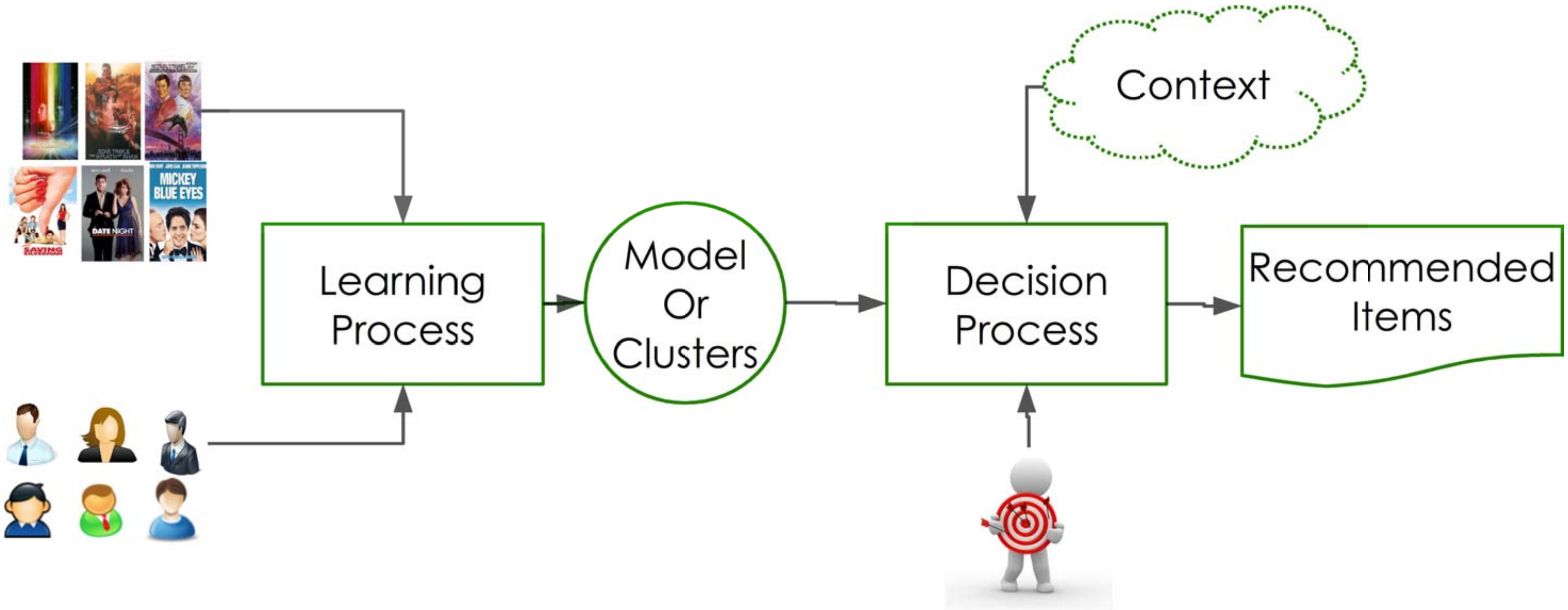
Performance Implications

- Bottleneck: Similarity computation.

  Time complexity, highly time consuming with millions of users & items in the database.
- Two-step process:
  - “off-line component” / “model”:

    similarity computation, precomputed & stored.
  - “on-line component”: prediction process.

Two-step process



Offline

Online

Performance Implications

- User-based similarity is more <u>dynamic</u>.

  Precomputing user neighbourhood can lead to poor predictions.

- Item-based similarity is <u>static</u>.

  We can precompute item neighbourhood. Online computation of the predicted ratings.

Memory based CF

+ Requires minimal knowledge engineering efforts

+ Users and products are symbols without any internal structure or characteristics

+ Produces good-enough results in most cases

- Requires a large number of explicit and reliable "ratings"

- Requires standardized products: users should have bought exactly the same product

- Assumes that prior behaviour determines current behaviour without taking into account "contextual" knowledge

Personalised vs Non-Personalised CF

- CF recommendations are personalized: the prediction is based on the ratings expressed by similar users; neighbours are different for each target user
- A non-personalized collaborative-based recommendation can be generated by averaging the recommendations of ALL users
- How would the two approaches compare?

Personalised vs Non-Personalised CF

Data Set	users	items	total	density	MAE Non Pers	MAE Pers
Jester	48483	100	3519449	0,725	0,220	0,152
MovieLens	6040	3952	1000209	0,041	0,233	0,179
EachMovie	74424	1649	2811718	0,022	0,223	0,151

Mean Average Error Non Personalized:

$$MAE_{NP} = \frac{\sum_{i,j} |v_{ij} - v_j|}{num.ratings}$$

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

The Sparsity Problem

Typically large product sets & few user ratings e.g. Amazon:

- in a catalogue of 1 million books, the probability that two users who bought 100 books each, have a book in common is 0.01
- in a catalogue of 10 million books, the probability that two users who bought 50 books each, have a book in common is 0.0002

- CF must have a number of users ~ 10% of the product catalogue size

The Sparsity Problem

Methods for dimensionality reduction

- Matrix Factorization
- SVD
- Clustering

Model-Based
Collaborative Filtering

Model Based CF Algorithms

Models are learned from the underlying data rather than heuristics.

Models of user ratings (or purchases):

- Clustering (classification)
- Association rules
- Matrix Factorization
- Restricted Boltzmann Machines
- Other models:
  - Bayesian network (probabilistic)
  - Probabilistic Latent Semantic Analysis ...

Clustering

- **Cluster** customers into categories based on preferences & past purchases

- **Compute** recommendations at the cluster level:

all customers within a cluster receive the same recommendations

Clustering

	BOOK 1	BOOK 2	BOOK 3	BOOK 4	BOOK 5	BOOK 6
CUSTOMER A	X			X		
CUSTOMER B		X	X		X	
CUSTOMER C		X	X			
CUSTOMER D		X				X
CUSTOMER E	X				X	

B, C & D form 1 CLUSTER vs. A & E form another cluster.

- « Typical » preferences for CLUSTER are:
  - Book 2, very high
  - Book 3, high
  - Books 5 & 6, may be recommended

Clustering

	BOOK 1	BOOK 2	BOOK 3	BOOK 4	BOOK 5	BOOK 6
CUSTOMER A	X			X		
CUSTOMER B		X	X		X	
CUSTOMER C		X	X			
CUSTOMER D		X				X
CUSTOMER E	X				X	

Clustering

+ It can also be applied for selecting the k most relevant neighbours in a CF algorithm

+ Faster: recommendations are per cluster

- less personalized: recommendations are per cluster vs. in CF they are per user

Association rules

Past purchases used to find relationships of common purchases

	BOOK 1	BOOK 2	BOOK 3	BOOK 4	BOOK 5	BOOK 6
CUSTOMER A	X			X		
CUSTOMER B		X	X		X	
CUSTOMER C		X	X			
CUSTOMER D		X				X
CUSTOMER E	X				X	
CUSTOMER F			X		X	

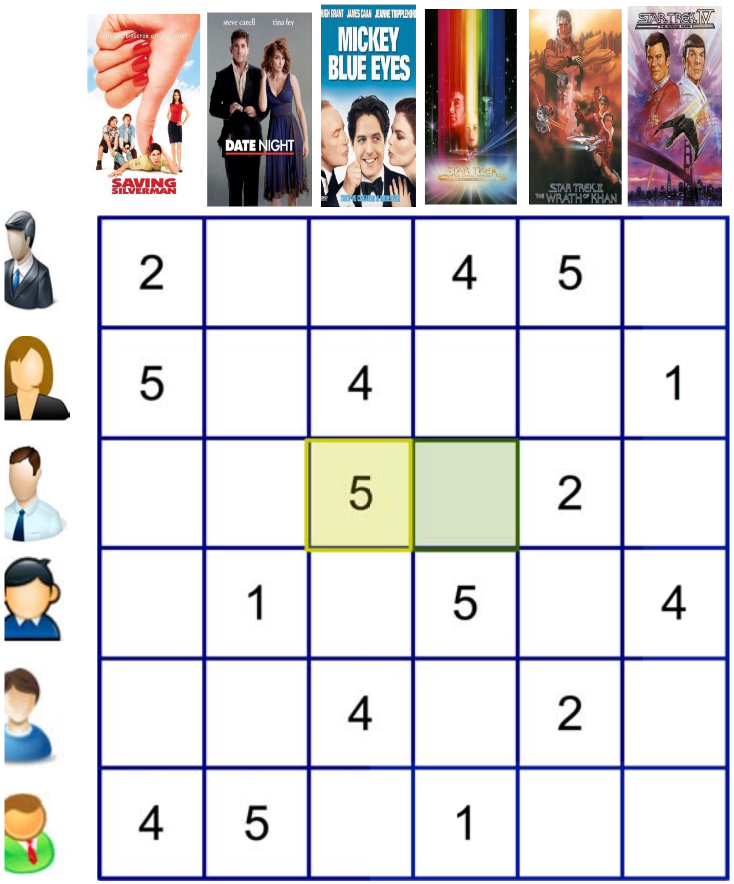
	BOOK 1	BOOK 2	BOOK 3	BOOK 4	BOOK 5	BOOK 6
BOOK 1				1	1	
BOOK 2			2		1	1
BOOK 3		2			2	
BOOK 4	1					
BOOK 5	1	1	2			
BOOK 6		1				

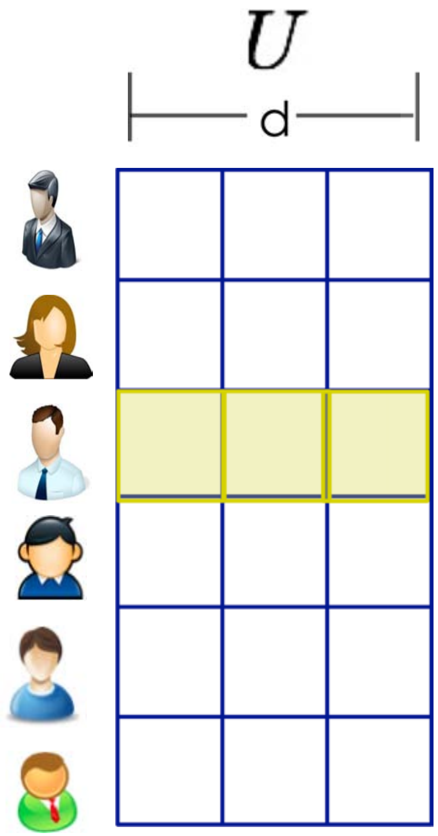
Association rules

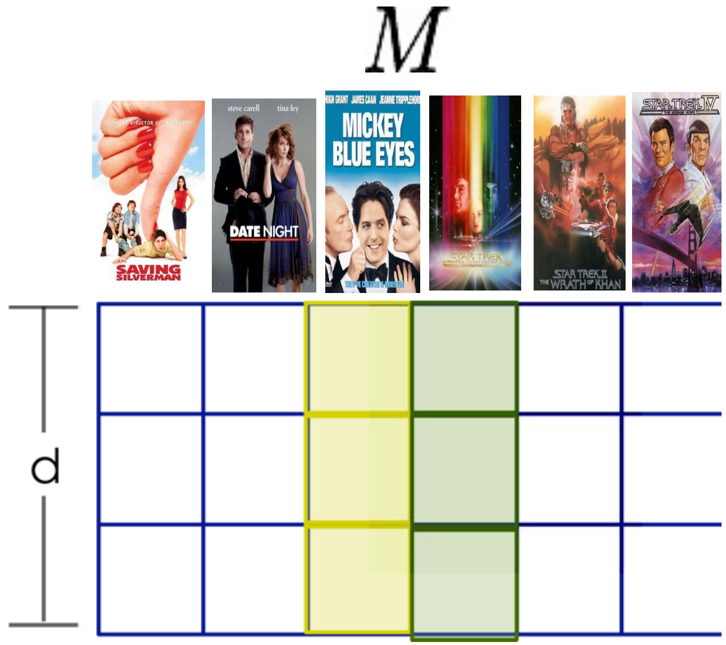
+ Fast to implement
+ Fast to execute
+ Not much storage space required
+ Not « individual » specific
+ Very successful in broad applications for large populations, such as shelf layout in retail stores

- Not suitable if preferences change rapidly
- Rules can be used only when enough data validates them. False associations can arise

Matrix Factorization







$$f_{ij} = \langle U_{i*}, M_{*,j} \rangle$$

$$E(U, M) = L(y_{i,j}, f_{ij}) + \Omega(U, M$$

Loss Functions for MF

- Squared error loss: $L(y_{i,j}, f_{i,j}) = \frac{1}{2}(y_{i,j} - f_{i,j})^2$
- Mean Average Error: $L(y_{i,j}, f_{i,j}) = |y_{i,j} - f_{i,j}|$
- Binary Hinge loss: $L(y_{i,j}, f_{i,j}) = max(0, 1 - y_{i,j}, f_{i,j})$

Learning: Stochastic Gradient Descent