Recommender Systems

MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5197422

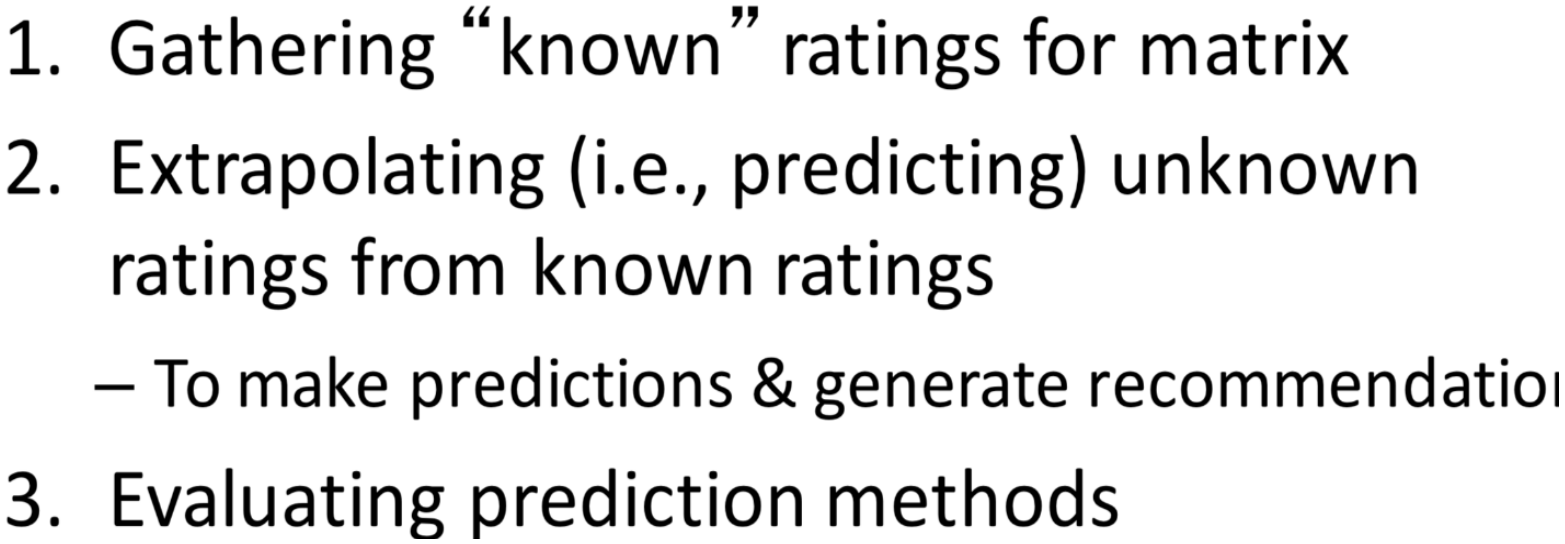
From Scarcity to abundance

- “Shelf space” is a scarce commodity for traditional retailers
  - Also: TV networks, movie theaters,…
- The web enables near-zero-cost dissemination of information about products
  - From scarcity to abundance
- More choice necessitates better filters
  - Recommendation engines

A Common Challenge

- Assume you’re a company selling **items** of some sort: movies, songs, products, etc.
- Company collects millions of **ratings** from **users** of their **items**
- To maximize profit / user happiness, you want to **recommend** items that users are likely to want

3 key challenges



1. Gathering "known" ratings for matrix
2. Extrapolating (i.e., predicting) unknown ratings from known ratings
   - To make predictions & generate recommendatio
3. Evaluating prediction methods

Recommender Systems

- **Setup:**
  - Items:
    movies, songs, products, etc.
    (often many thousands)
  - Users:
    watchers, listeners, purchasers, etc.
    (often many millions)
  - Feedback:
    5-star ratings, not-clicking ‘next’,
    purchases, etc.
- **Key Assumptions:**
  - Can represent ratings numerically as a user/item matrix
  - Users only rate a small number of items (the matrix is sparse)

	Doctor Strange	Star Trek: Beyond	Zootopia
Alice	1		5
Bob	3	4	
Charlie	3	5	2

Two Types of Recommender Systems

Content Filtering

- *Example*: Pandora.com music recommendations (Music Genome Project)
- **Con:** Assumes access to side information about items (e.g. properties of a song)
- **Pro:** Got a new item to add? No problem, just be sure to include the side information

Collaborative Filtering

- *Example*: Netflix movie recommendations
- **Pro:** Does not assume access to side information about items (e.g. does not need to know about movie genres)
- **Con:** Does not work on new items that have no ratings

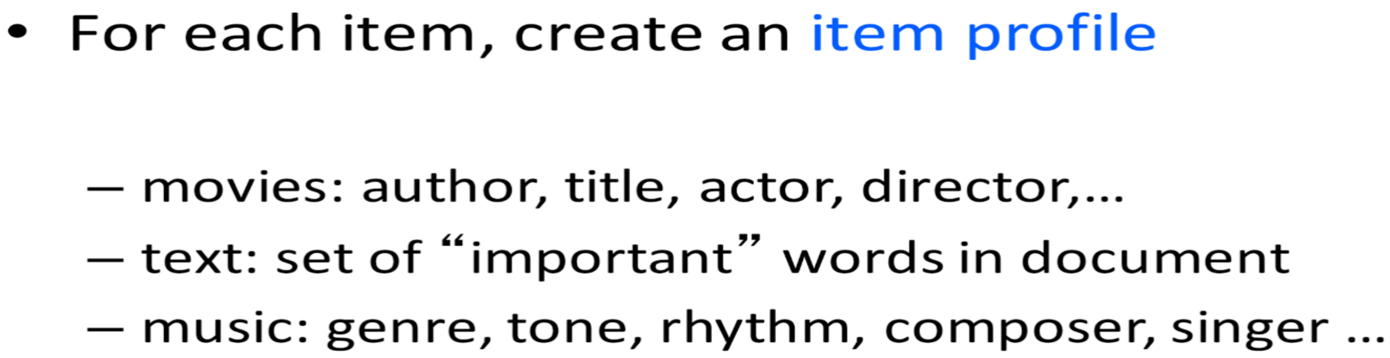
Content based filtering vs Collaborative filtering

- **Collaborative algorithm** uses “User Behavior” for recommending items. They exploit behavior of other users and items in terms of transaction history, ratings, selection and purchase information. Other users behavior and preferences over the items are used to recommend items to the new users.
- **Content-Based Recommendation** is that we have to know the content of both user and item. Usually we construct user-profile and item-profile using the content of shared attribute space. For example, for a movie, you represent it with the movie stars in it and the genres (using a binary coding for example). For user profile, you can do the same thing based on the users likes some movie stars/genres etc. To calculate how good a movie is to a user, we use cosine similarity

Content based recommendation

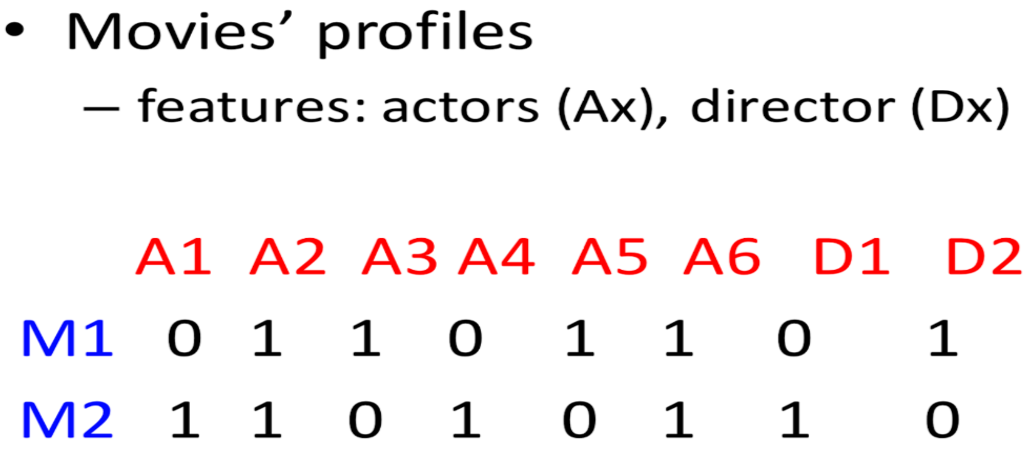
- Main idea: recommend items to customer C similar to previous items rated highly by C
- Movie recommendations
  - recommend movies with same actor(s), director, genre, …
- Websites, blogs, news
  - recommend other sites with “similar” content

Item Profiles



- For each item, create an item profile
  - movies: author, title, actor, director,…
  - text: set of “important” words in document
  - music: genre, tone, rhythm, composer, singer …

Example



- Movies’ profiles
  - features: actors (Ax), director (Dx)

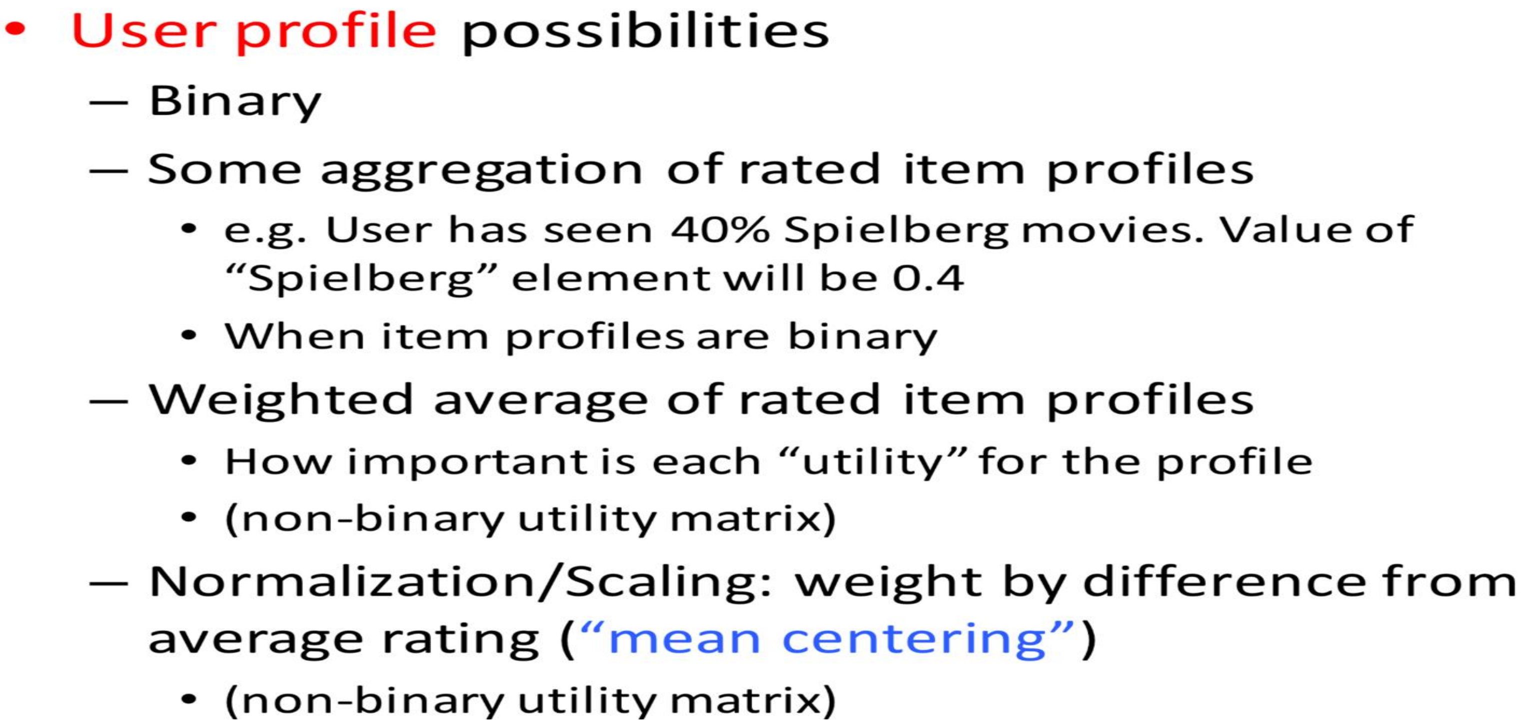
	A1	A2	A3	A4	A5	A6	D1	D2
M1	0	1	1	0	1	1	0	1
M2	1	1	0	1	0	1	1	0

User Profile

- For each user, create a user profile based on the content the user has consumed
- Should follow the same template as the item profiles
    - E.g.

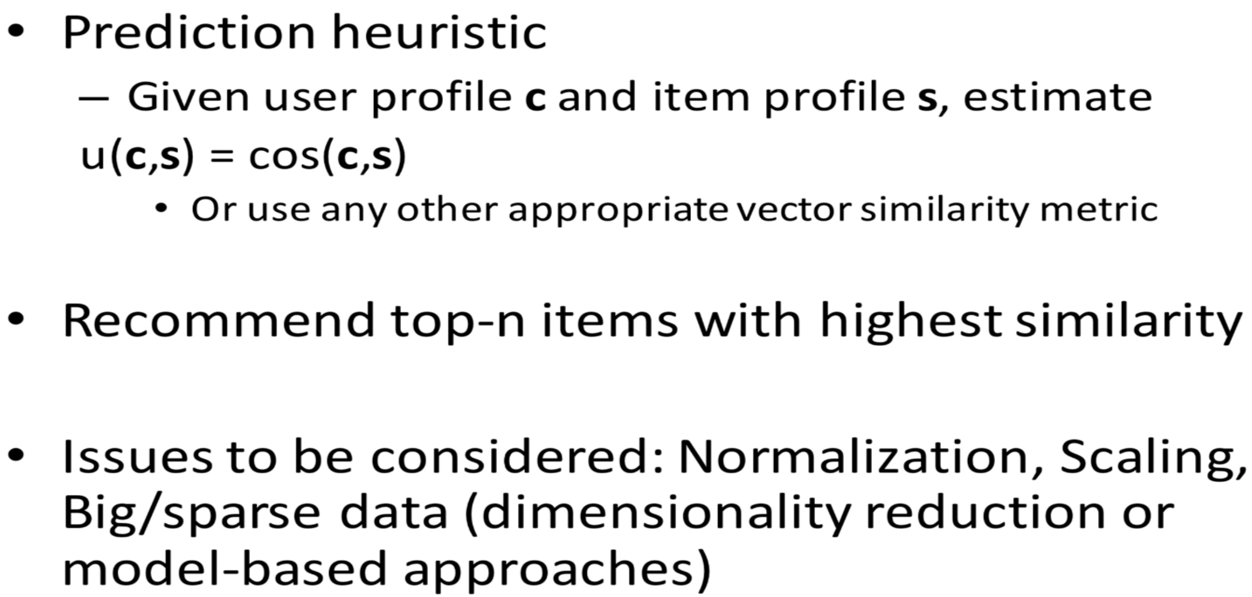
	A1	A2	A3	A4	A5	A6	D1	D2
U1	0	0	1	0	1	0	0	1

User Profiles



- User profile possibilities
  - Binary
  - Some aggregation of rated item profiles
    - e.g. User has seen 40% Spielberg movies. Value of "Spielberg" element will be 0.4
    - When item profiles are binary
  - Weighted average of rated item profiles
    - How important is each "utility" for the profile
    - (non-binary utility matrix)
  - Normalization/Scaling: weight by difference from average rating ("mean centering")
    - (non-binary utility matrix)

Prediction and recommendation



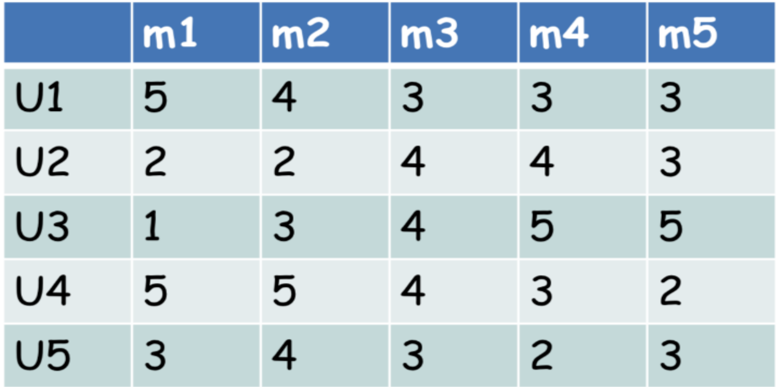
- Prediction heuristic
  - Given user profile **c** and item profile **s**, estimate u(**c**,**s**) = cos(**c**,**s**)
    - Or use any other appropriate vector similarity metric
- Recommend top-n items with highest similarity
- Issues to be considered: Normalization, Scaling, Big/sparse data (dimensionality reduction or model-based approaches)

Limitation and Challenges

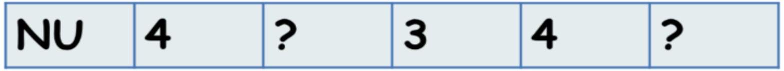
- Finding the appropriate features to represent the items
  - e.g., images, movies, music

- Overspecialization
  - Never recommends items outside user’ s content profile
  - People might have multiple interests

Collaborative Filtering



	m1	m2	m3	m4	m5
U1	5	4	3	3	3
U2	2	2	4	4	3
U3	1	3	4	5	5
U4	5	5	4	3	2
U5	3	4	3	2	3

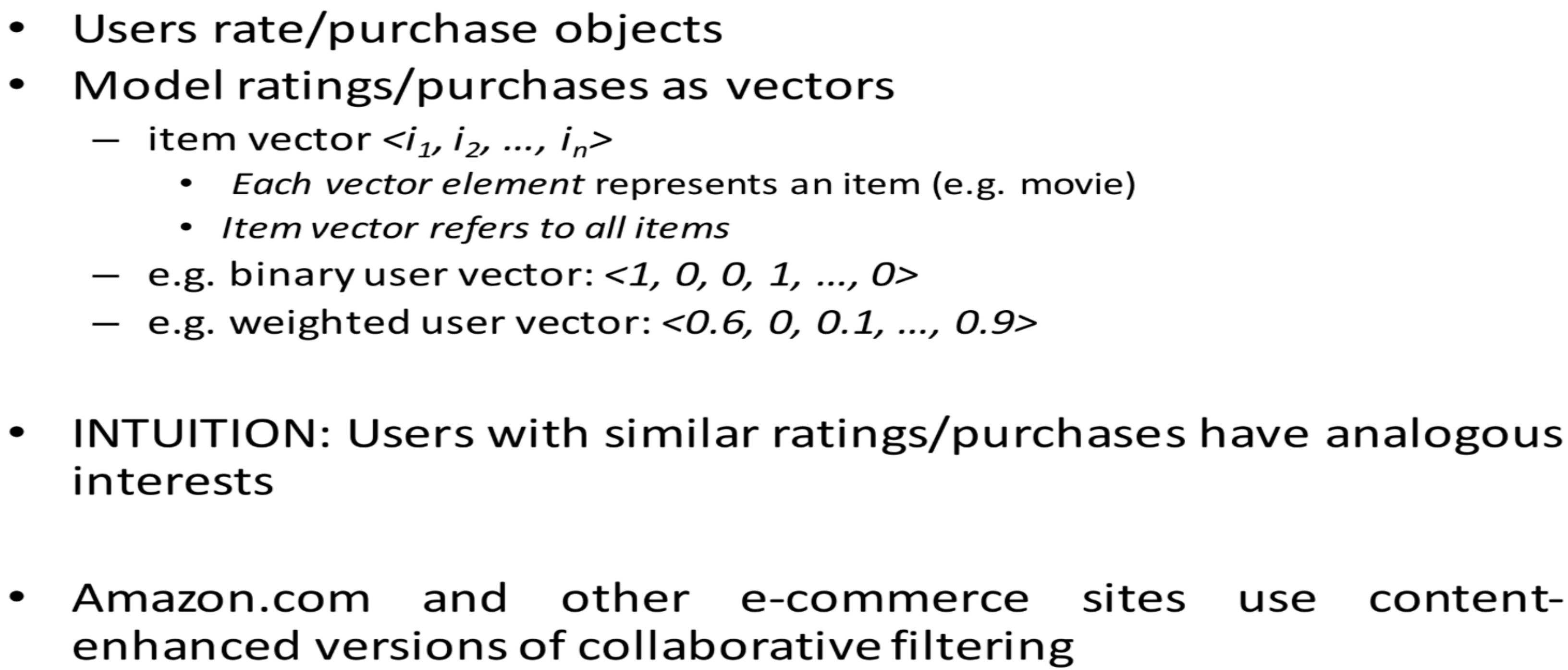


NU	4	?	3	4	?

- User-item matrix

- How to predict missing ratings?

Collaborative Filtering Intuition

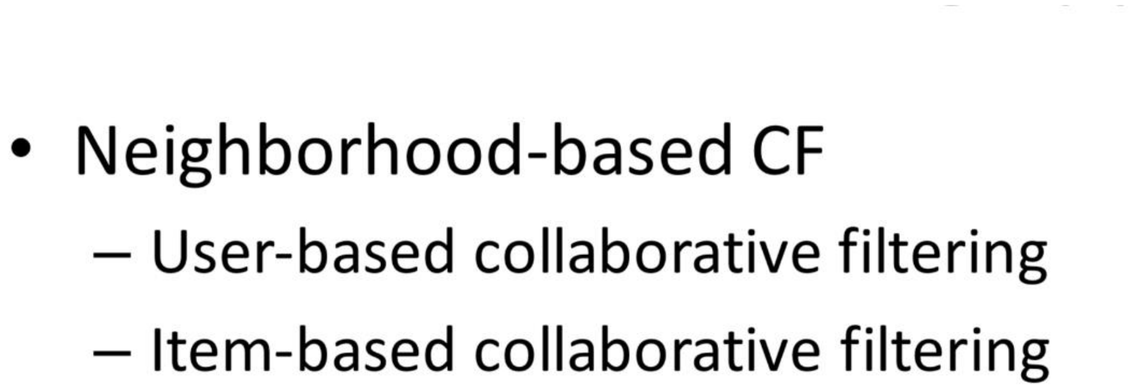


- Users rate/purchase objects
- Model ratings/purchases as vectors
  - item vector *<i_1, i_2, ..., i_n>*
    - *Each vector element* represents an item (e.g. movie)
    - *Item vector refers to all items*
  - e.g. binary user vector: *<1, 0, 0, 1, ..., 0>*
  - e.g. weighted user vector: *<0.6, 0, 0.1, ..., 0.9>*

- INTUITION: Users with similar ratings/purchases have analogous interests

- Amazon.com and other e-commerce sites use content-enhanced versions of collaborative filtering

Collaborative filtering approaches



- Neighborhood-based CF
  - User-based collaborative filtering
  - Item-based collaborative filtering

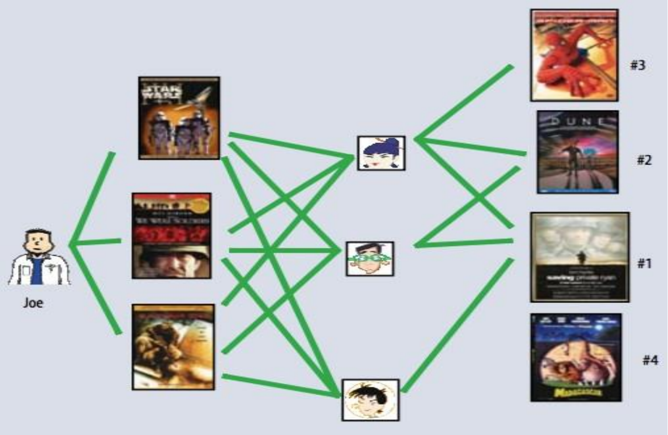
Traditional Recommendation Setup

Users

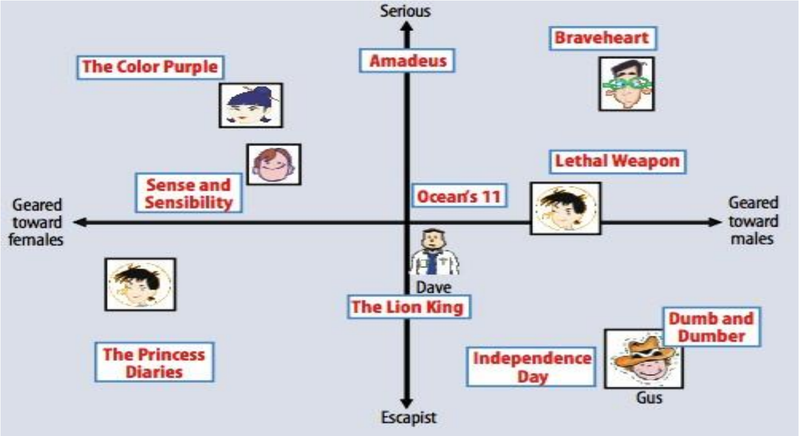
	0	1	0	1	0
	0	0	1	1	0
	1	0	0	1	1
	0	1	0	0	0
	0	0	0	0	1

Two Types of Collaborative Filtering

1. Neighborhood Methods



2. Latent Factor Methods

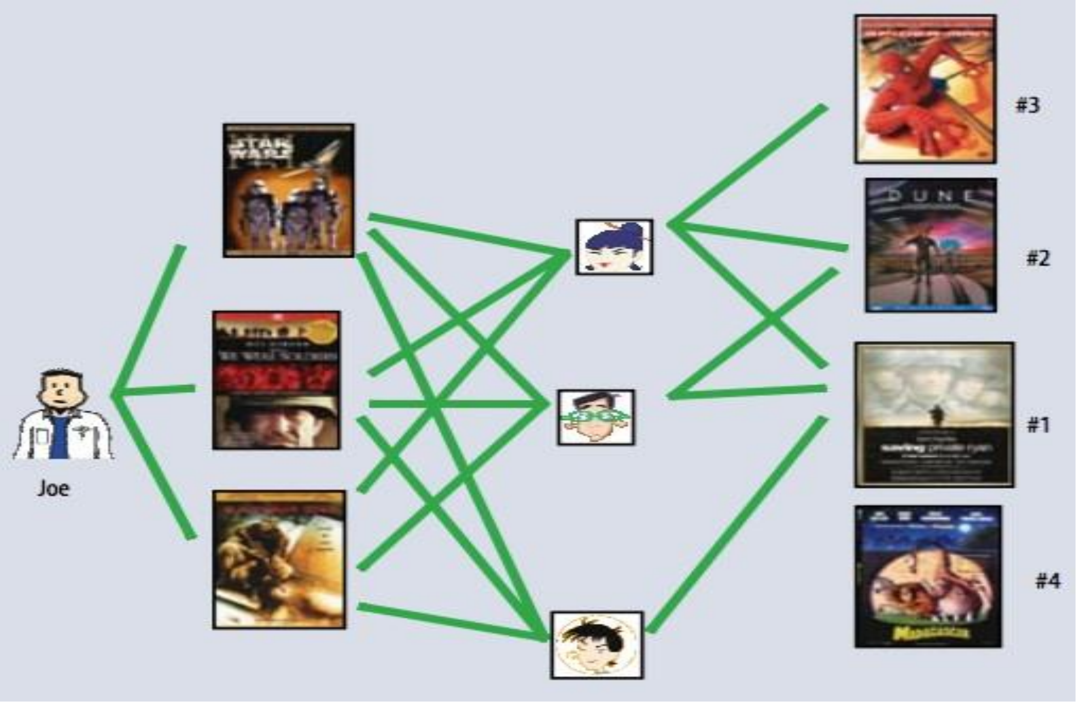


Figures from Koren et al. (2009)

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5197422

Two Types of Collaborative Filtering

1. Neighborhood Methods



- In the figure, assume that a green line indicates the movie was **watched**

- **Algorithm:**

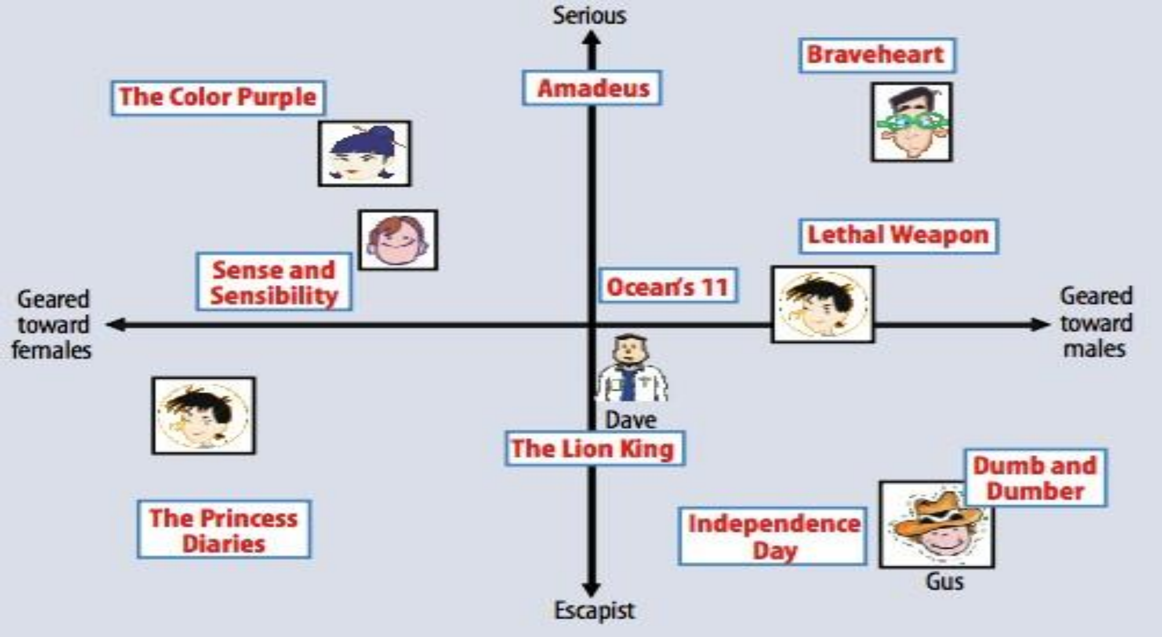
1. **Find neighbors** based on similarity of movie preferences
2. **Recommend** movies that those neighbors watched

Figures from Koren et al. (2009)

Two Types of Collaborative Filtering

2. Latent Factor Methods

- Assume that both movies and users live in some **low-- dimensional space** describing their properties
- **Recommend** a movie based on its **proximity** to the user in the latent space



Figures from Koren et al. (2009)

Matrix Factorization

- Many different ways of factorizing a matrix
- We'll consider three:
    1. Unconstrained Matrix Factorization
    2. Singular Value Decomposition
    3. Non-negative Matrix Factorization
- MF is just another example of a **common recipe**:
    1. define a model
    2. define an objective function
    3. optimize with SGD

Matrix factorization methods

Item

User

$$
\begin{array}{c|cccc}
 & W & X & Y & Z \\
\hline
A & & 4.5 & 2.0 & \\
B & 4.0 & & 3.5 & \\
C & & 5.0 & & 2.0 \\
D & & 3.5 & 4.0 & 1.0
\end{array}
=
\begin{array}{c|cc}
A & 1.2 & 0.8 \\
B & 1.4 & 0.9 \\
C & 1.5 & 1.0 \\
D & 1.2 & 0.8
\end{array}
\times
\begin{array}{cccc}
W & X & Y & Z \\
\hline
1.5 & 1.2 & 1.0 & 0.8 \\
1.7 & 0.6 & 1.1 & 0.4
\end{array}
$$

Rating Matrix

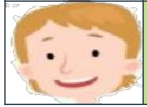
User Matrix

Item Matrix

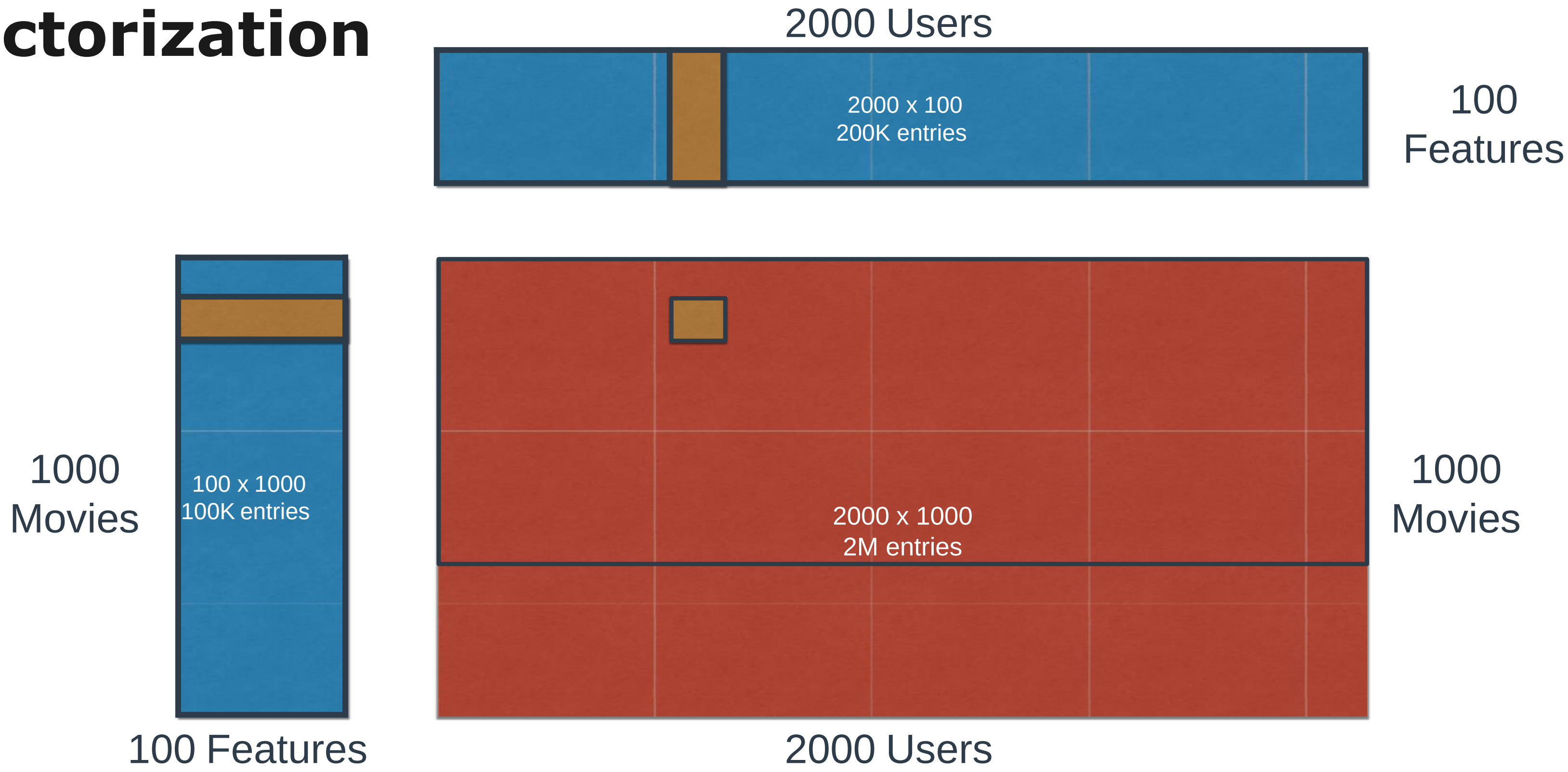
Matrix Factorization

	M1	M2	M3	M4	M5
Comedy	3	1	1	3	1
Action	1	2	4	1	3

	Comedy	Action
A	✓	✗
B	✗	✓
C	✓	✗
D	✓	✓

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

Matrix Factorization



2000 Users

2000 x 100
200K entries

100
Features

1000
Movies

100 x 1000
100K entries

2000 x 1000
2M entries

1000
Movies

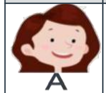
100 Features

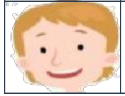
2000 Users

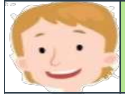
Error Function

	M1	M2	M3	M4	M5
F11.2	3.1	0.3	2.5	0.2	
F22.4	1.5	4.4	0.4	1.1	

$$\text{Error} = (3 - 1.44)^2$$

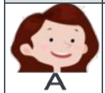
	F1	F2
 A	0.2	0.5
B	0.3	0.4
C	0.7	0.8
D	0.4	0.5

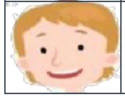
	M1	M2	M3	M4	M5
	1.44				
					

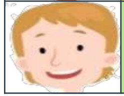
	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

Error Function

	M1	M2	M3	M4	M5
F11.2	3.1	0.3	2.5	0.2	
F22.4	1.5	4.4	0.4	1.1	

	F1	F2
 A	0.2	0.5
B	0.3	0.4
C	0.7	0.8
D	0.4	0.5

	M1	M2	M3	M4	M5
	1.44	1.37			
					

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

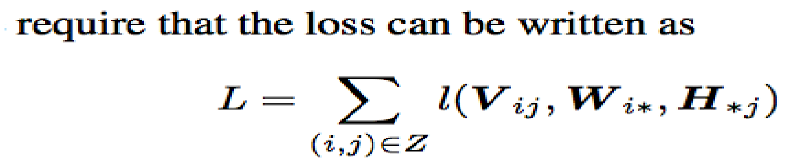
Derivative

$$\text{Error} = (3 - 1.44)^2 + (1 - 1.37)^2 + \ldots$$

Matrix Factorization

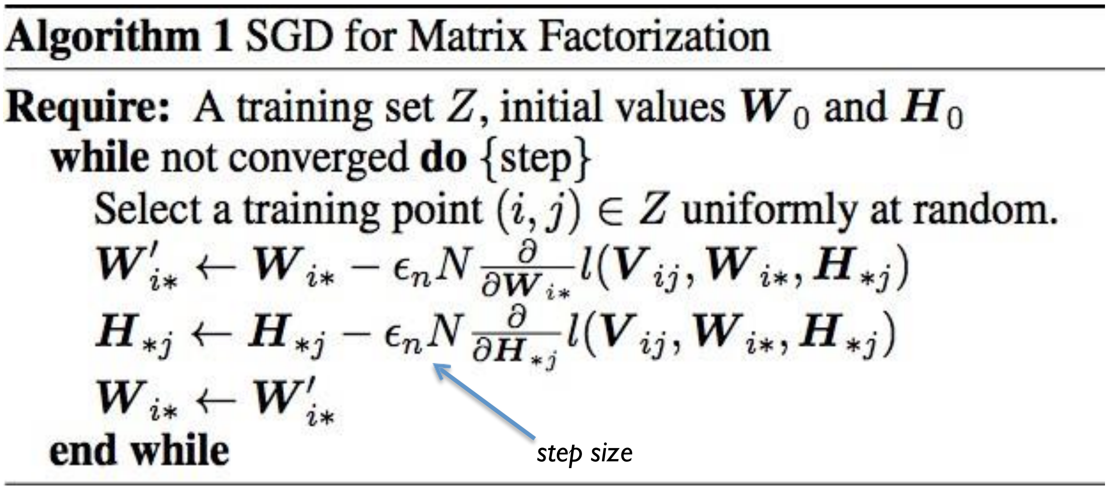
(with matrices)

- SGD



require that the loss can be written as

$$L = \sum_{(i,j) \in Z} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$$

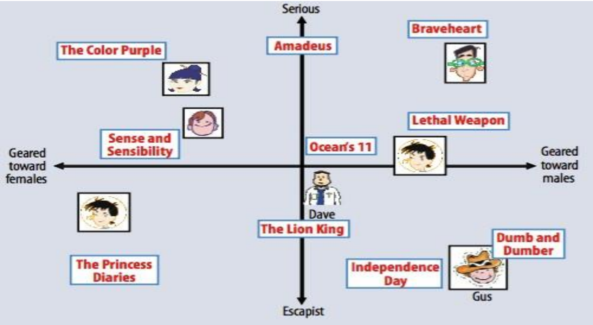


Algorithm 1 SGD for Matrix Factorization

```
Require: A training set $Z$, initial values $\boldsymbol{W}_0$ and $\boldsymbol{H}_0$
  while not converged do {step}
    Select a training point $(i, j) \in Z$ uniformly at random.
    $\boldsymbol{W}'_{i*} \leftarrow \boldsymbol{W}_{i*} - \epsilon_n N \frac{\partial}{\partial \boldsymbol{W}_{i*}} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$
    $\boldsymbol{H}_{*j} \leftarrow \boldsymbol{H}_{*j} - \epsilon_n N \frac{\partial}{\partial \boldsymbol{H}_{*j}} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$
    $\boldsymbol{W}_{i*} \leftarrow \boldsymbol{W}'_{i*}$
  end while
```

step size

Figure from Gemulla et al. (2011)



Figures from Koren et al. (2009)

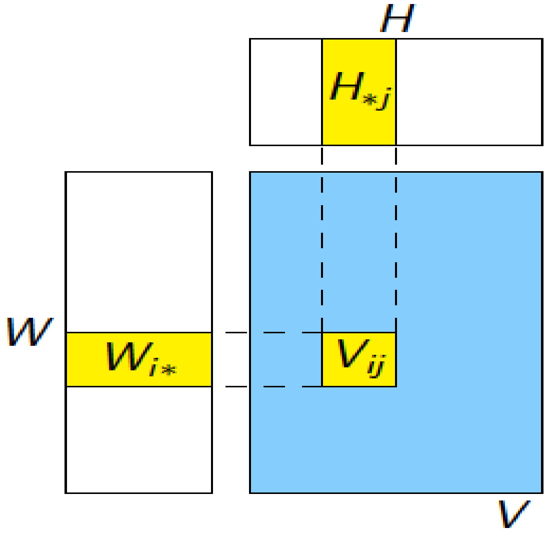
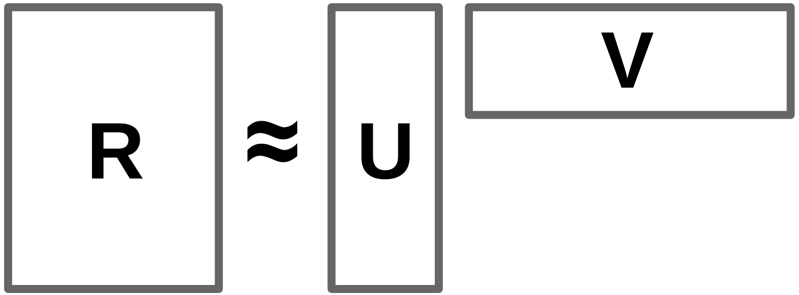


Figure from Gemulla et al. (2011)

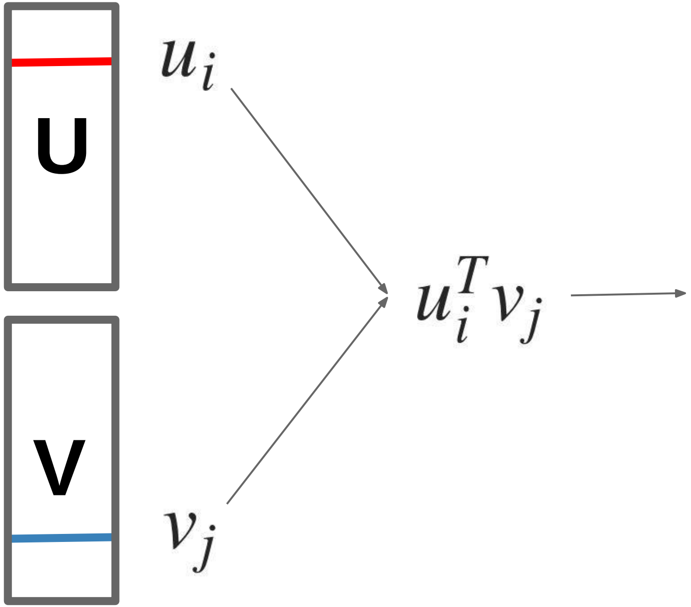
https://www.researchgate.net/publication/275522425_Large-Scale_Matrix_Factorization_with_Distributed_Stochastic_Gradient_Descent

A Matrix Factorization view



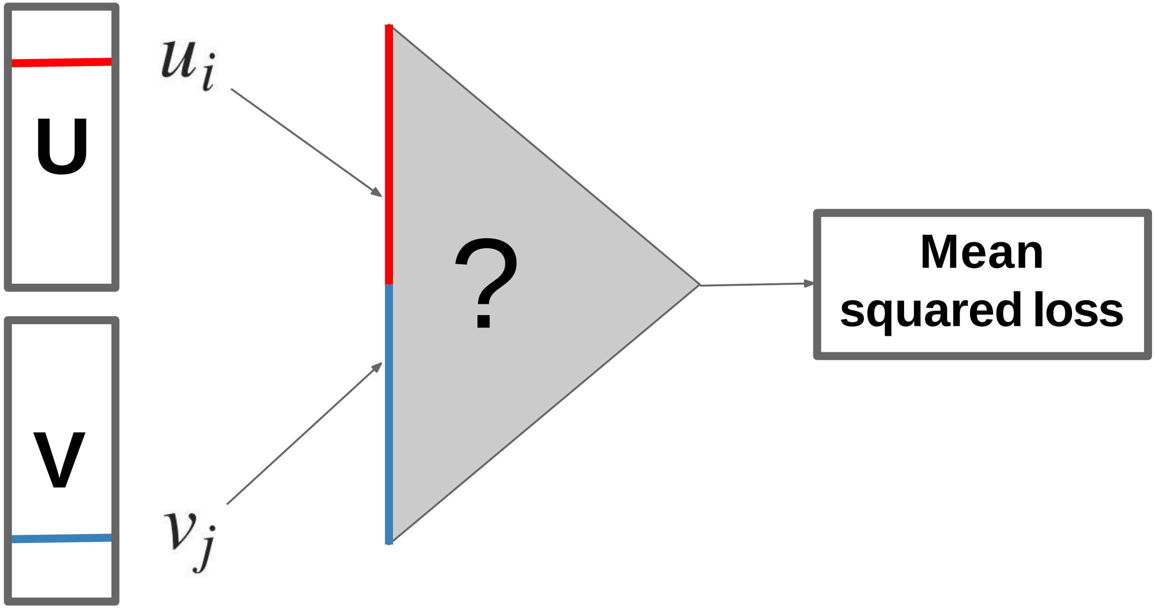
$$\min_{u,v} \sum_{i,j\in R} (r_{i,j} - u_i^T v_j) + \lambda(\|u_i\|^2 + \|v_j\|^2)$$

A Feed-Forward Networkview



$$\min_{u,v} \sum_{i,j\in R} (r_{i,j} - u_i^T v_j) + \lambda(\|u_i\|^2 + \|v_j\|^2)$$

A (deeper) feed-forward view



What’s going on?

- Very similar models: representation learning through embeddings, MSE loss, gradient-based optimization
- Main difference is that we can learn a different embedding combination than a dot product
- … but embeddings are arbitrary representations
- … and capturing pairwise interactions through a feed-forward net requires a very large model

Conclusion?

- Not much benefit in the ‘traditional’ recommendation setup of a deep versus a properly tuned model

Breaking the ‘traditional’ recsys setup

- Adding extra data / inputs
- Modeling different facets of users and items
- ***Alternative framings of the problem***

Content-based side information

- VBPR: helping cold-start by augmenting item factors with visual factors from CNNs [He et. al., 2015]
- Content2Vec [Nedelec et. al., 2017]
- Learning to approximate MF item embeddings from content [Dieleman, 2014]

Metadata-based side information

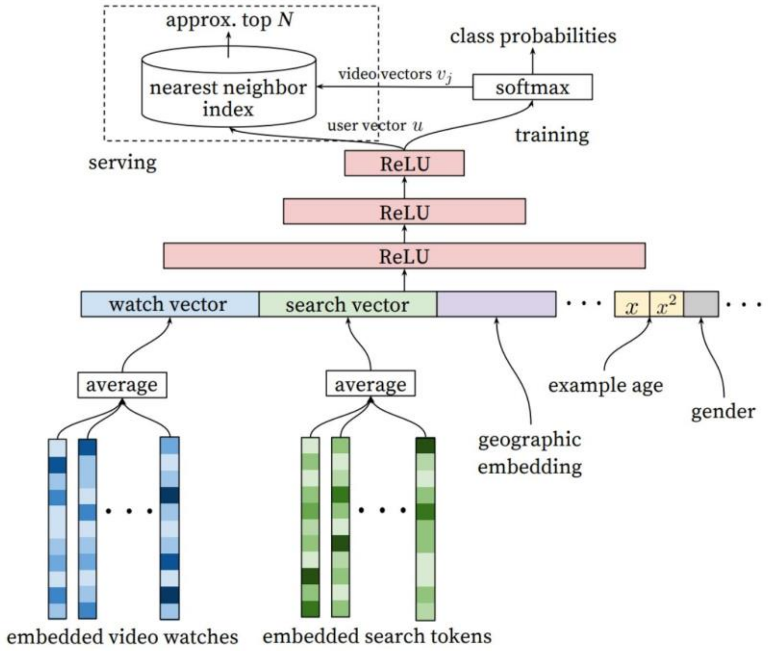
- Factorization Machines [Rendle, 2010] with side-information
  - Extending the factorization framework to an arbitrary number of inputs
- Meta-Prod2Vec [Vasile et. al., 2016]
  - Regularize item embeddings using side-information
- DCF [Li et al., 2016]
- Using associated textual information for recommendations [Bansal et. al., 2016]

YouTube Recommendations

- Two stage ranker: candidate generation (shrinking set of items to rank) and ranking (classifying actual impressions)

- Two feed-forward, fully connected, networks with hundreds of features

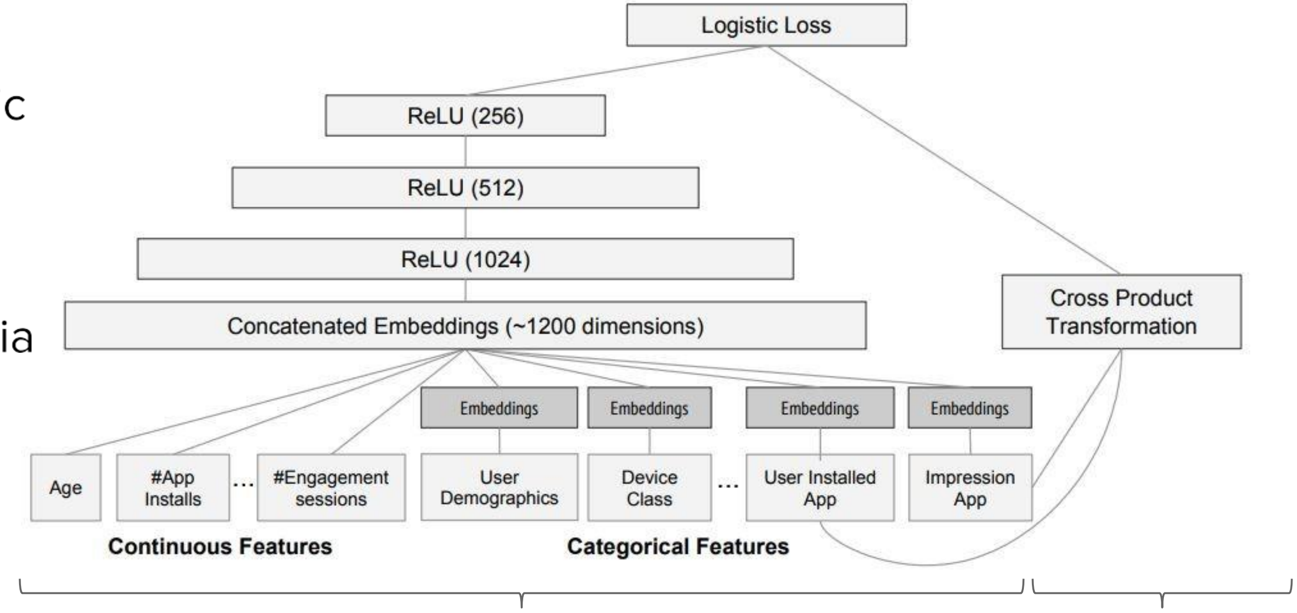
[Covington et. al., 2016]



https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45530.pdf

Wide +Deep models

- Wide model: memorize sparse, specific rules
- Deep model: generalize to similar items via embeddings



Deep

Wide
(many parameters due to cross product)

[Cheng et. al., 2016]

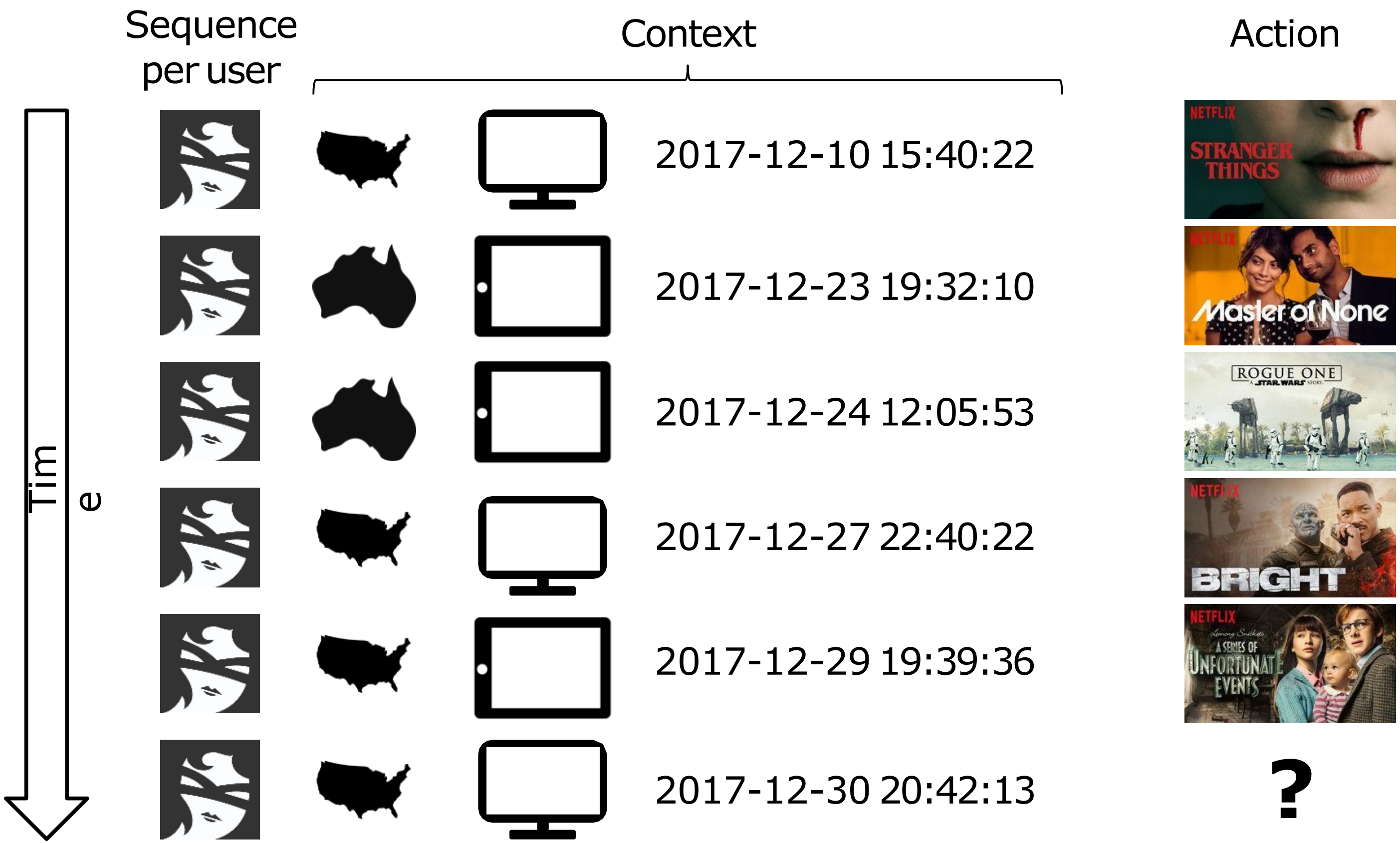
Contextual sequence prediction

- Input: sequence of *contextual* user actions, plus current *context*
- Output: probability of next action

- *E.g. "Given all the actions a user has taken so far, what's the most likely video they're going to play right now?"*

- e.g. [Smirnova & Vasile, 2017], [Beutel et. al., 2018]

Contextual sequence data



Sequence per user

Context

Action

2017-12-10 15:40:22

2017-12-23 19:32:10

2017-12-24 12:05:53

2017-12-27 22:40:22

2017-12-29 19:39:36

2017-12-30 20:42:13

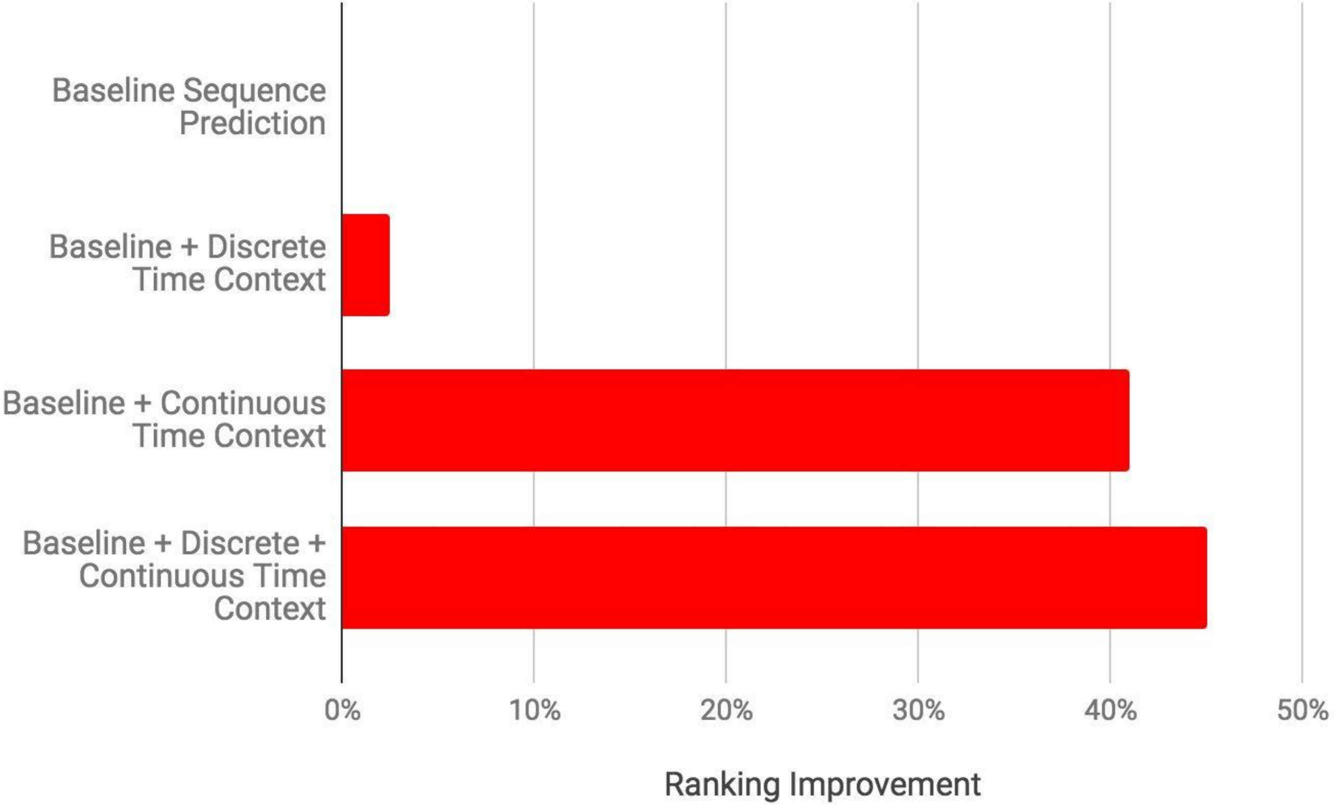
?

Time-sensitive sequence prediction

- Recommendations are actions at a moment in time
  - Proper modeling of time and system dynamics is critical

- Experiment on a Netflix internal dataset
  - Context:
    - Discrete time
      - Day-of-week: Sunday, Monday, ...
      - Hour-of-day
    - Continuous time (Timestamp)
  - Predict next play (temporal split data)

Results



Baseline Sequence Prediction
Baseline + Discrete Time Context
Baseline + Continuous Time Context
Baseline + Discrete + Continuous Time Context
0%
10%
20%
30%
40%
50%
Ranking Improvement

Other framings

- Causality in recommendations
  - Explicitly modeling the consequence of a recommender systems’ intervention [Schnabel et al., 2016]
- Recommendation as question answering
  - E.g. “I loved Billy Madison, My Neighbor Totoro, Blades of Glory, Bio-Dome, Clue, and Happy Gilmore. I’m looking for a Music movie.” [Dodge et al., 2016]
- Deep Reinforcement Learning for recommendations [Zhao et al, 2017]

https://arxiv.org/pdf/1801.00209.pdf

Conclusion

- Deep Learning can work well for Recommendations… when you go beyond the classic problem definition
- Similarities between DL and MF are a good thing: Lots of MF work can be translated to DL
- Lots of open areas to improve recommendations using deep learning
- Think beyond solving existing problems with new tools and instead what new problems they can solve