Recommender Systems

Weinan Zhang
Shanghai Jiao Tong University
http://wnzhang.net

http://wnzhang.net/teaching/ee448/index.html

Content of This Course

- Overview of personalized recommendation
- Collaborative filtering
- Rating prediction
- Top-N ranking

A Data Mining Application: Recommendation

Overview

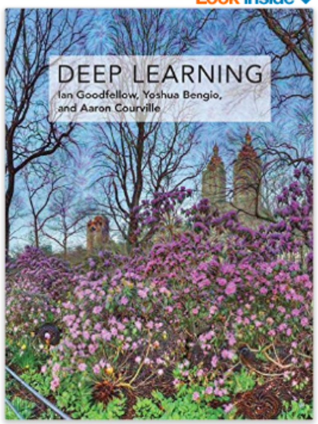
Collaborative Filtering

Rating prediction

Top-N ranking

Sincerely thank Prof. Jun Wang

Book Recommendation



Look inside

See this image

Deep Learning (Adaptive Computation and Machine Learning series) Hardcover – November 18, 2016

by Ian Goodfellow (Author), Yoshua Bengio (Author), Aaron Courville (Author)

46 customer reviews

#1 Best Seller in Artificial Intelligence & Semantics

See all formats and editions

Hardcover
$64.00

11 Used from $80.12
15 New from $64.00

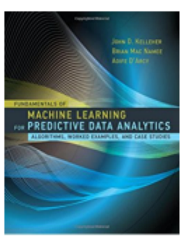
Primestudent **College student?** FREE shipping and exclusive deals Learn more ›

"Written by three experts in the field, *Deep Learning* is the only comprehensive book on the subject." -- **Elon Musk**, cochair of OpenAI; cofounder and CEO of Tesla and SpaceX

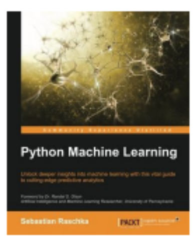
Deep learning is a form of machine learning that enables computers to learn from experience and understand the world in terms of a hierarchy of concepts. Because the computer gathers knowledge from experience, there is no need for a human computer operator to formally specify all the knowledge

Read more

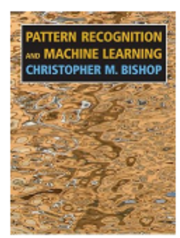
Customers Who Bought This Item Also Bought



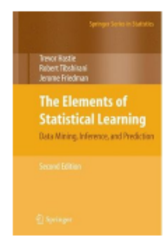
Fundamentals of Machine Learning for Predictive Data Analytics:...
› John D. Kelleher
22
Hardcover
$76.00 Prime



Python Machine Learning
› Sebastian Raschka
98
#1 Best Seller in Computer Neural Networks
Paperback
$40.49 Prime



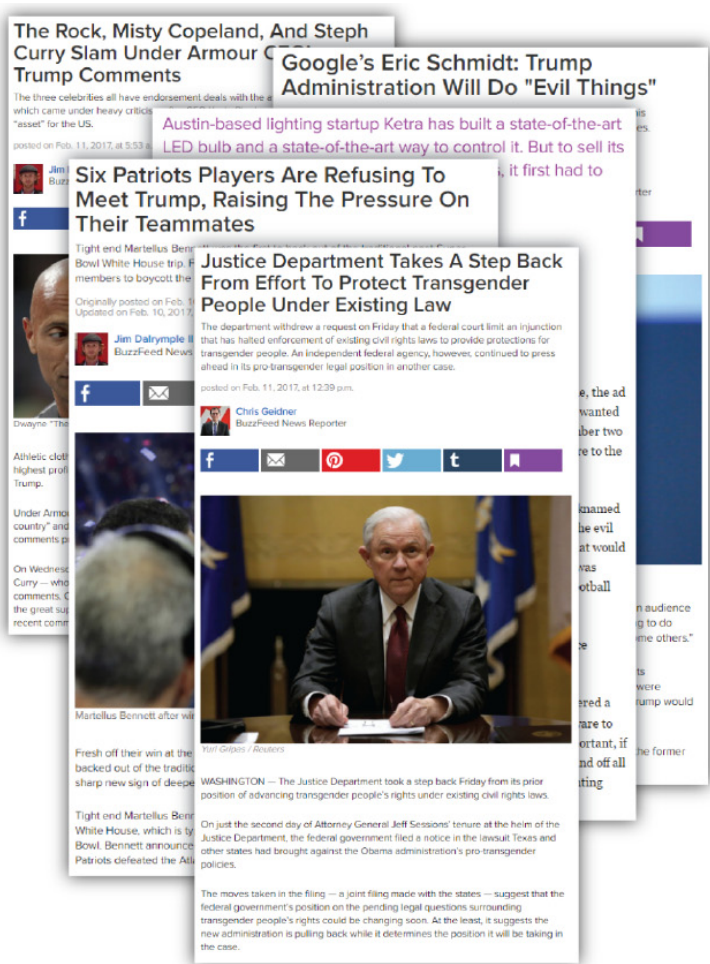
Pattern Recognition and Machine Learning (Information Science and...
› Christopher M. Bishop
132
Hardcover
$63.68 Prime



The Elements of Statistical Learning: Data Mining, Inference, and...
Trevor Hastie
92
#1 Best Seller in Bioinformatics
Hardcover
$73.18 Prime

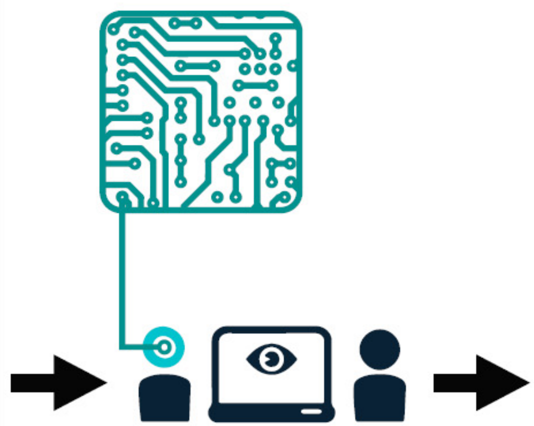
Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques...
› Aurélien Géron
Paperback
$28.56 Prime

News Feed Recommendation

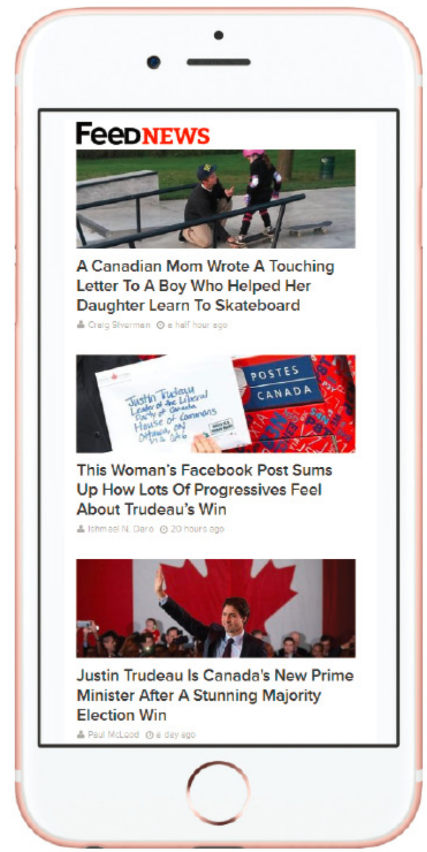


Huge numbers of candidate articles daily

Recommender System



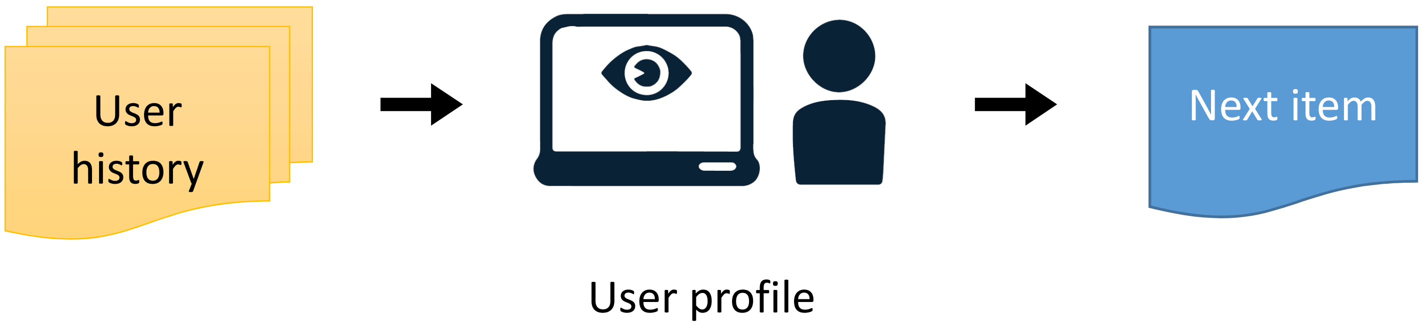
Editors manually select quality articles



Quality articles selected for news feed to end users

Personalized Recommendation

- Personalization framework



Build user profile from her history

- Ratings [amazon.com]
  - Explicit, but time-consuming
- Visits [newsfeed]
  - Implicit

Personalization Methodologies

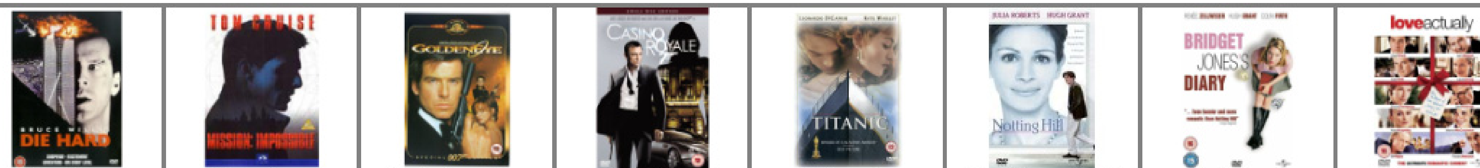
- Given the user's previous liked movies, how to recommend more movies she would like?
  - Method 1: recommend the movies that share the actors/actresses/director/genre with those the user likes
  - Method 2: recommend the movies that the users with similar interest to her like

Information Filtering

- Information filtering deals with the delivery of information that the user is likely to find interesting or useful
  - Recommender system: information filtering in the form of suggestions
  - Two approaches for information filtering
    - Content-based filtering
      - recommend the movies that share the actors/actresses/director/genre with those the user likes
    - Collaborative filtering (the focus of this lecture)
      - recommend the movies that the users with similar interest to her like

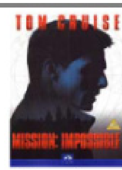
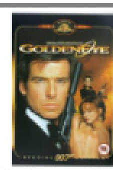
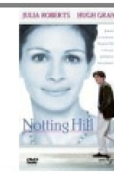
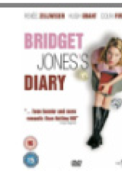
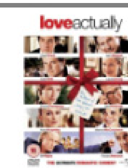
http://recommender-systems.org/information-filtering/

A (small) Rating Matrix

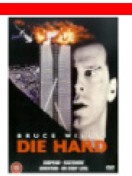
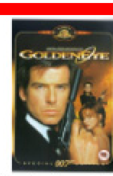
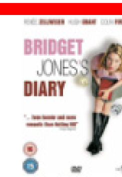
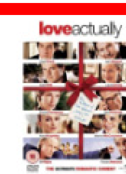


	Die Hard	Mission: Impossible	GoldenEye	Casino Royale	Titanic	Notting Hill	Bridget Jones's Diary	Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

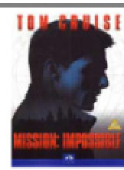
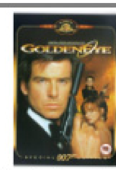
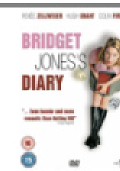
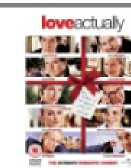
The Users

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

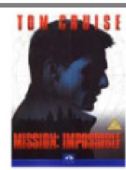
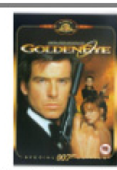
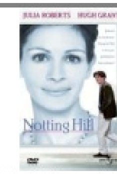
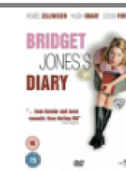
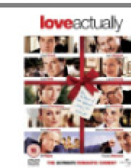
The Items

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

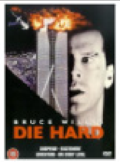
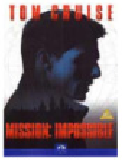
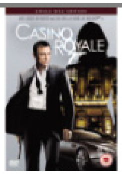
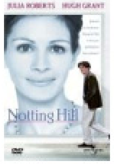
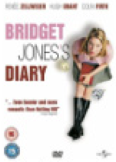
A User-Item Rating

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		
Dave		★★★★★	$\boxed{★★★★★}$	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

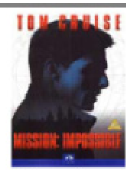
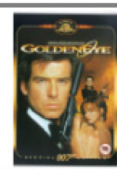
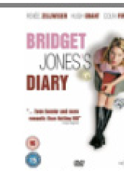
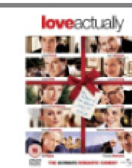
A User Profile

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

An Item Profile

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

A Null Rating Entry

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

- Recommendation on explicit data
  - Predict the null ratings

If I watched *Love Actually*, how would I rate it?

K Nearest Neighbor Algorithm (KNN)

- A non-parametric method used for classification and regression

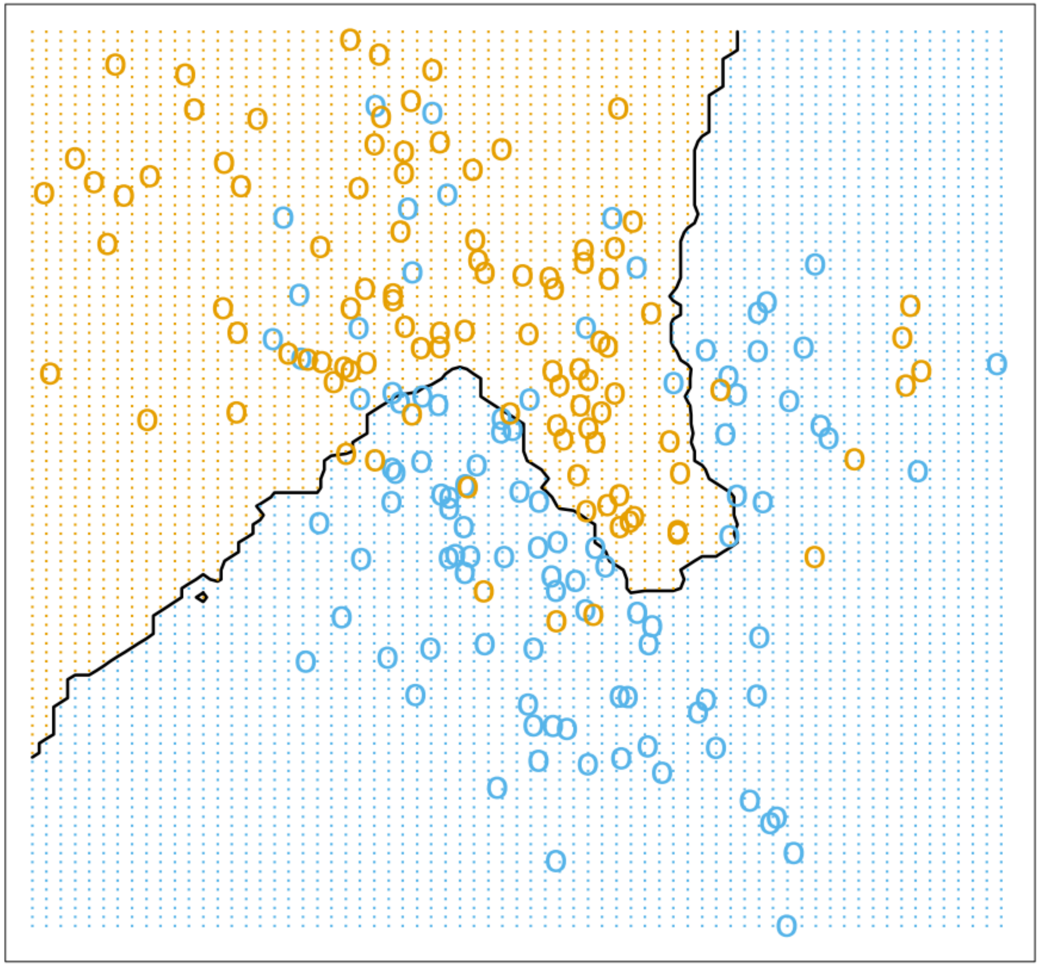
  - for each input instance *x*, find *k* closest training instances $N_k(x)$ in the feature space
  - the prediction of *x* is based on the average of labels of the *k* instances

$$\hat{y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

  - For classification problem, it is the majority voting among neighbors

kNN Example

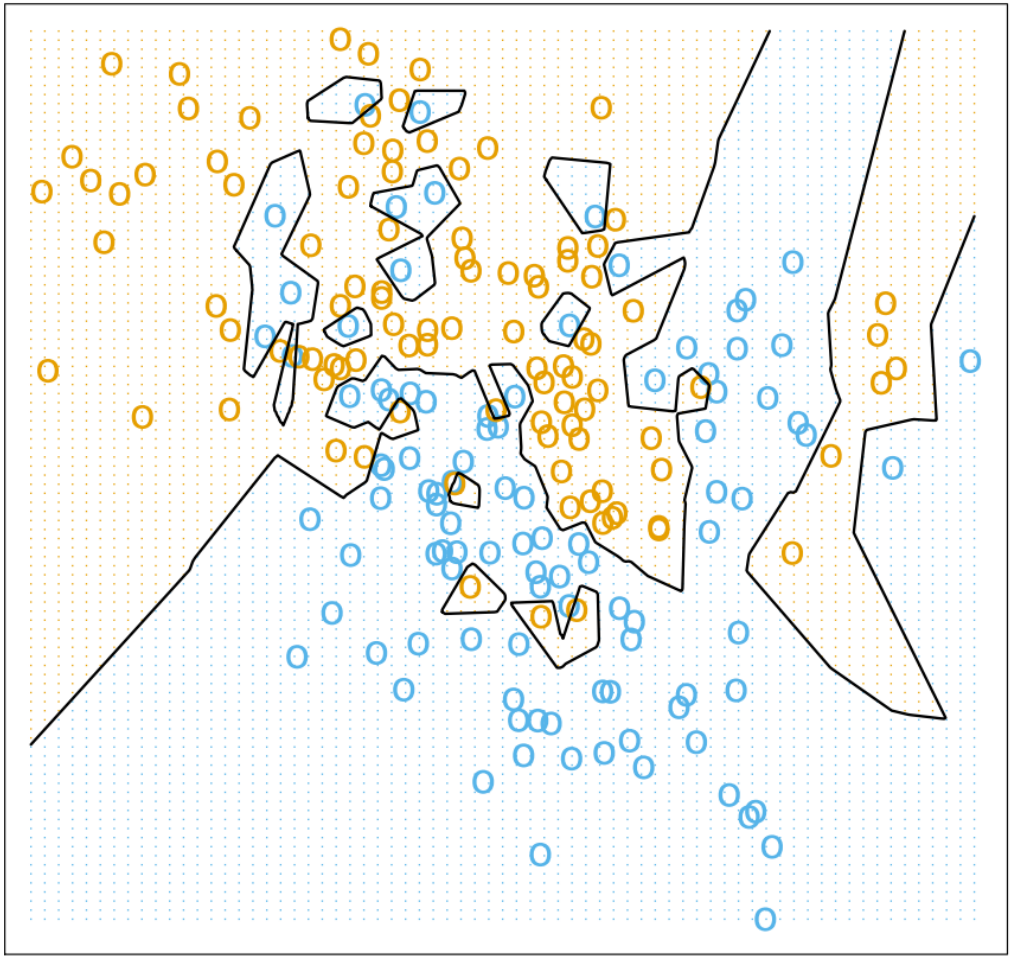
15-nearest neighbor



Jerome H. Friedman, Robert Tibshirani, and Trevor Hastie. “The Elements of Statistical Learning”. Springer 2009.

kNN Example

1-nearest neighbor



Jerome H. Friedman, Robert Tibshirani, and Trevor Hastie. “The Elements of Statistical Learning”. Springer 2009.

K Nearest Neighbor Algorithm (KNN)

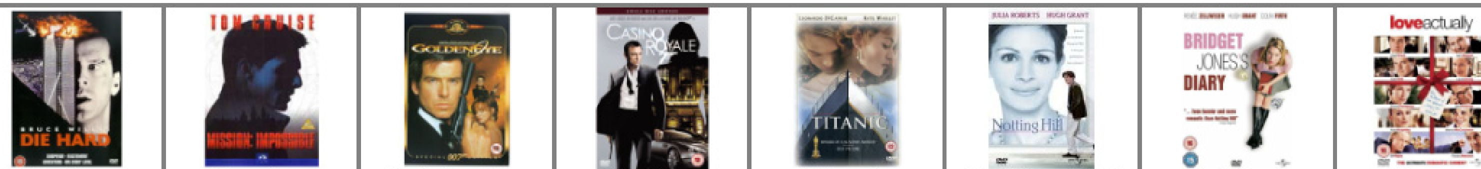
- Generalized version
  - Define similarity function $s(x, x_i)$ between the input instance x and its neighbor x_i
  - Then the prediction is based on the weighted average of the neighbor labels based on the similarities

$$\hat{y}(x) = \frac{\sum_{x_i \in N_k(x)} s(x, x_i) y_i}{\sum_{x_i \in N_k(x)} s(x, x_i)}$$

Non-Parametric kNN

- No parameter to learn
  - In fact, there are N parameters: each instance is a parameter
  - There are N/k effective parameters
    - Intuition: if the neighborhoods are non-overlapping, there would be N/k neighborhoods, each of which fits one parameter
- Hyperparameter k
  - We cannot use sum-of-squared error on the training set as a criterion for picking k, since k=1 is always the best
  - Tune k on validation set

A Null Rating Entry

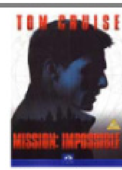
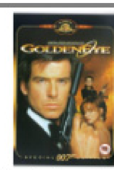
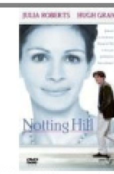
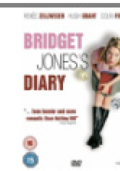
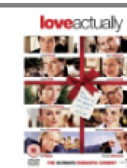


	Die Hard	Mission: Impossible	GoldenEye	Casino Royale	Titanic	Notting Hill	Bridget Jones's Diary	Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

- Recommendation on explicit data
  - Predict the null ratings

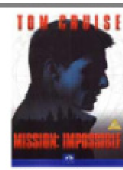
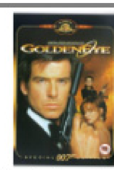
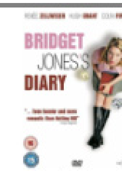
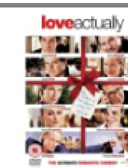
If I watched *Love Actually*, how would I rate it?

Collaborative Filtering Example

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		**?**
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

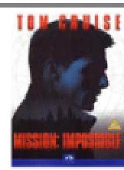
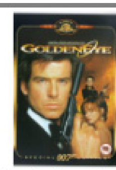
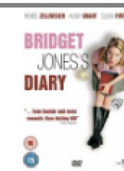
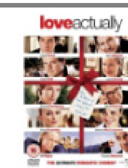
- What do you think the rating would be?

User-based kNN Solution

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		?
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

- Find similar users (neighbors) for Boris
  - Dave and George

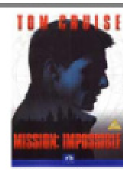
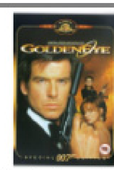
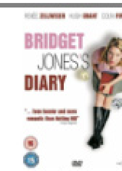
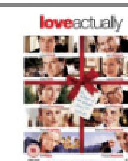
Rating Prediction

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		★⯪☆☆☆
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

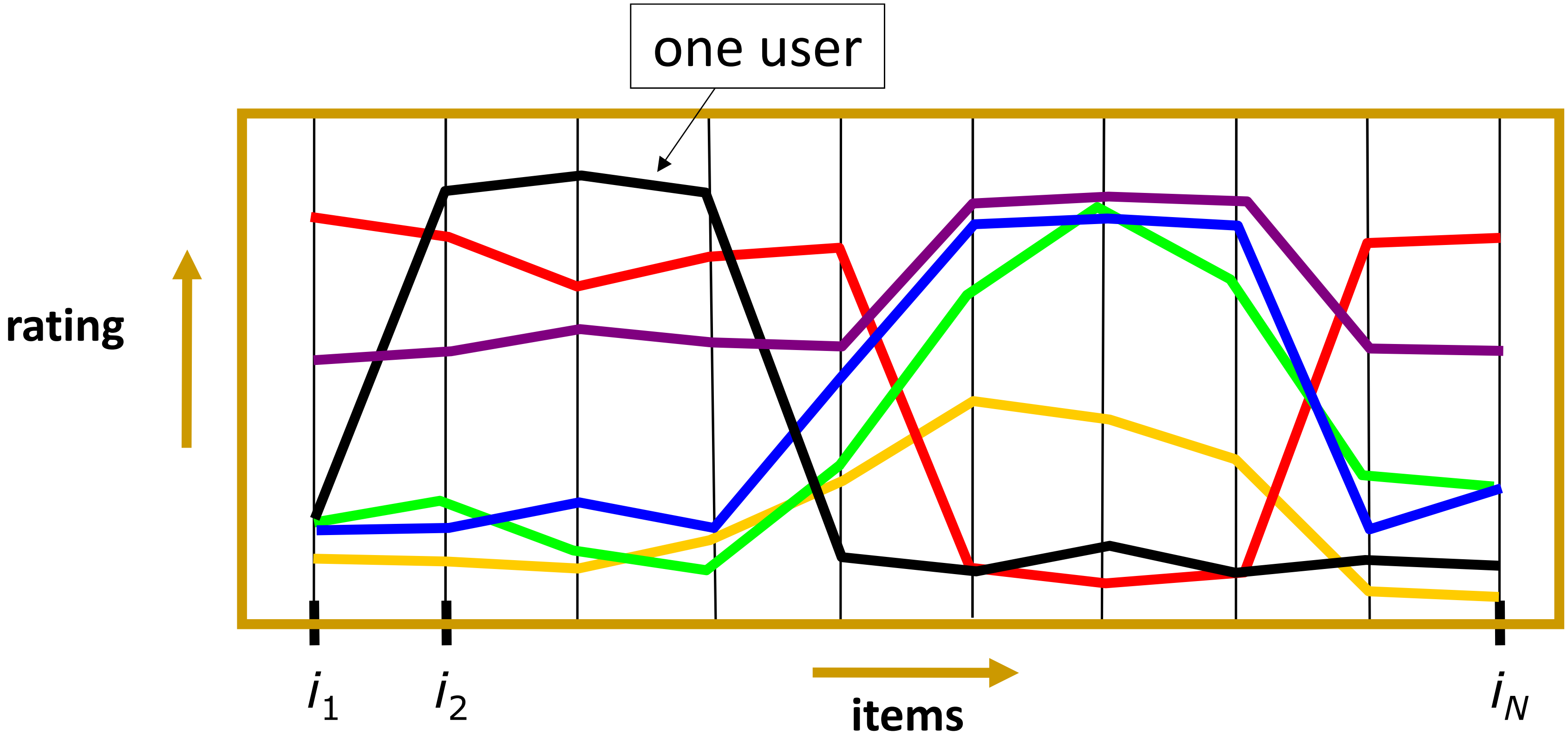
- Average Dave's and George's rating on Love Actually
  - Prediction = (1+2)/2 = 1.5

Collaborative Filtering for Recommendation

- Basic user-based kNN algorithm
  - For each target user for recommendation
    1. Find similar users
    2. Based on similar users, recommend new items

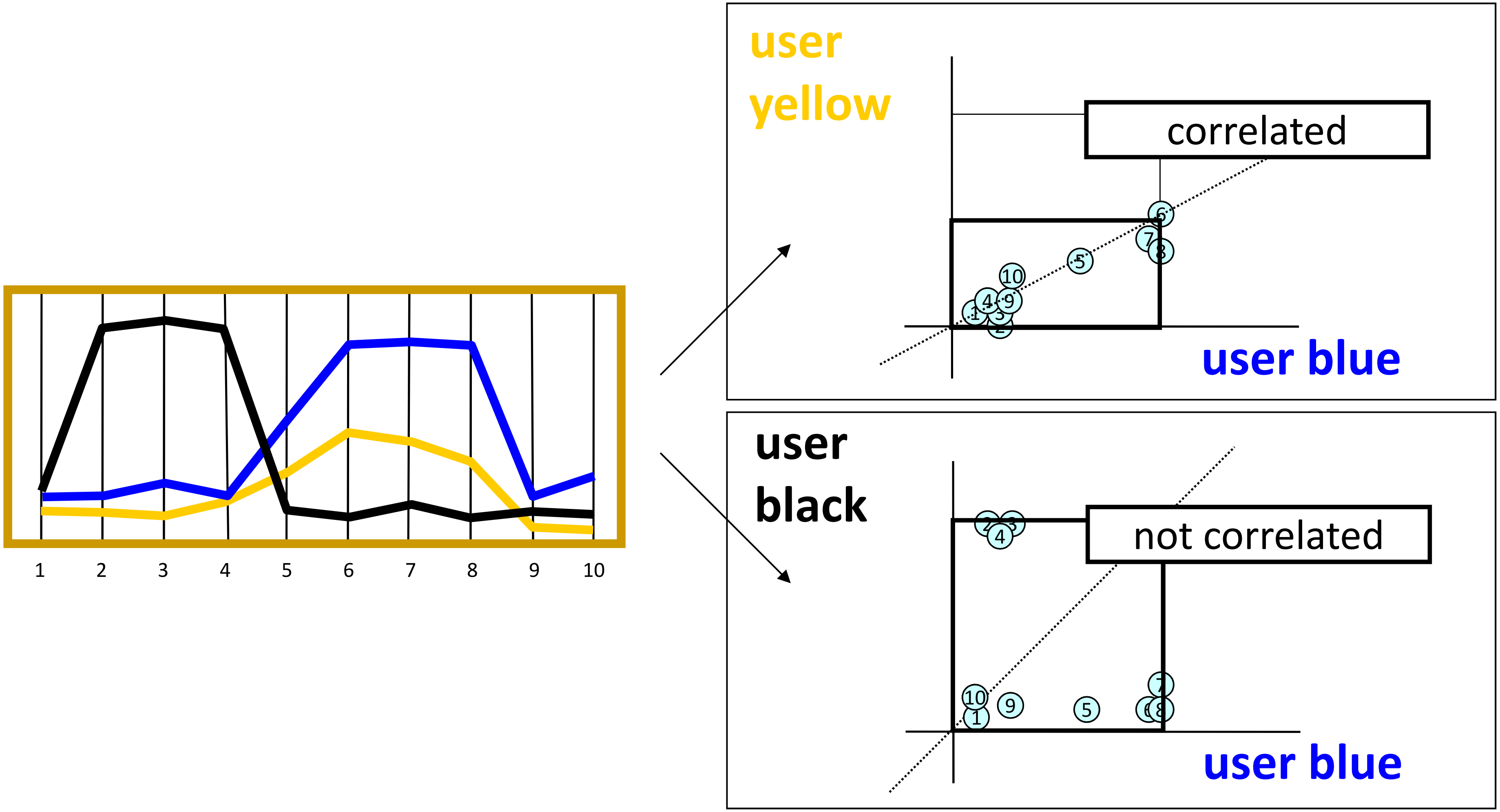
	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		★★☆☆☆
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

Similarity between Users

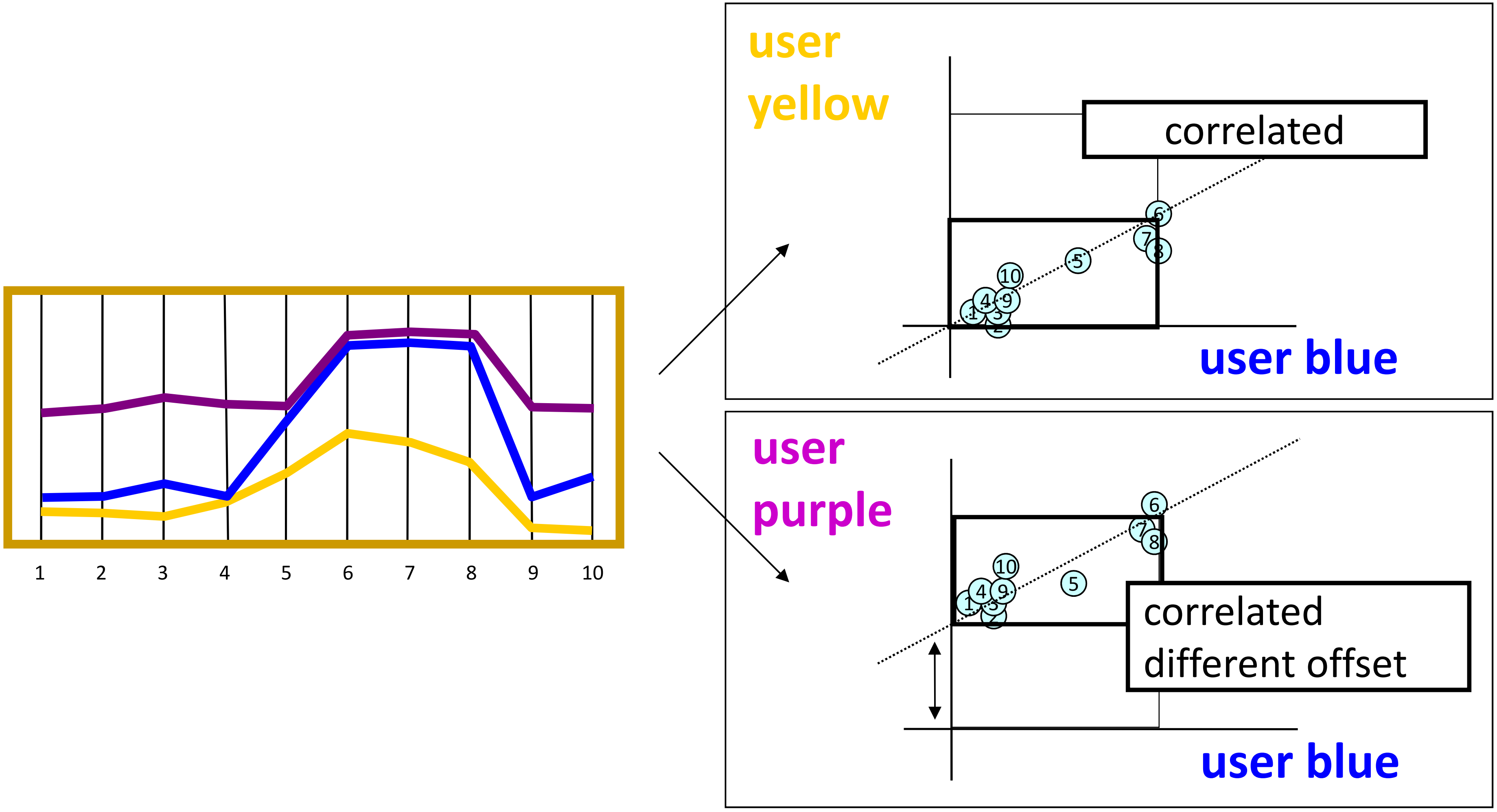


- Each user's profile can be directly built as a vector based on her item ratings

Similarity between Users



Similarity between Users



Similarity Measures (Users)

- Similarity measures between two users *a* and *b*
  - Cosine (angle)

$$s_u^{\cos}(u_a, u_b) = \frac{u_a^\top u_b}{\|u_a\|\|u_b\|} = \frac{\sum_m x_{a,m} x_{b,m}}{\sqrt{\sum_m x_{a,m}^2 \sum_m x_{b,m}^2}}$$

  - Pearson Correlation

$$s_u^{\mathrm{corr}}(u_a, u_b) = \frac{\sum_m (x_{a,m} - \bar{x}_a)(x_{b,m} - \bar{x}_b)}{\sqrt{\sum_m (x_{a,m} - \bar{x}_a)^2 \sum_m (x_{b,m} - \bar{x}_b)^2}}$$

User-based kNN Rating Prediction

- Predicting the rating from target user *t* to item *m*

Weights according to similarity with query user

predict according to rating of similar user (corrected for average rating of that user)

$$\hat{x}_{t,m} = \bar{x}_t + \frac{\sum_{u_a \in N_u(u_t)} s_u(u_t, u_a)(x_{a,m} - \bar{x}_a)}{\sum_{u_a \in N_u(u_t)} s_u(u_t, u_a)}$$

correct for average rating query user

$$\bar{u}_t = \frac{1}{|I(u_t)|} \sum_{m \in I(u_t)} x_{t,m}$$

normalize such that ratings stay between 0 and *R*

Item-based kNN Solution

- Recommendation based on item similarity

query item

similar items based on profiles

	i_1	i_2	i_3	i_4	i_5	…	i_M
u_1	2	0	3	2	5	…	1
u_2	0	4	0	0	0	…	5
u_3	0	2	0	0	0	…	4
u_4	1	0	4	2	4	…	2
…	…	…	…	…	…	…	…
u_K	2	0	4	?	4	…	1

target user

predicted rating: 4

Item-based kNN Solution

- For each unrated items *m* of the target user *t*
  - Find similar items {*a*}
  - Based on the set of similar items {*a*}
    - Predict the rating of the item *m*

query item

similar items based on profiles

	i_1	i_2	i_3	i_4	i_5	…	i_M
u_1	2	0	3	2	5	…	1
u_2	0	4	0	0	0	…	5
u_3	0	2	0	0	0	…	4
u_4	1	0	4	2	4	…	2
…	…	…	…	…	…	…	…
u_K	2	0	$\boxed{4}$	?	$\boxed{4}$	…	1

target user

predicted rating: 4

Similarity Measures (Items)

- Similarity measures between two items *a* and *b*
  - Cosine (angle)

$$s_i^{\cos}(i_a, i_b) = \frac{i_a^\top i_b}{\|i_a\| \|i_b\|} = \frac{\sum_u x_{u,a} x_{u,b}}{\sqrt{\sum_u x_{u,a}^2 \sum_u x_{u,b}^2}}$$

  - Adjusted Cosine

$$s_i^{\text{adcos}}(i_a, i_b) = \frac{\sum_u (x_{u,a} - \bar{x}_u)(x_{u,b} - \bar{x}_u)}{\sqrt{\sum_u (x_{u,a} - \bar{x}_u)^2 \sum_u (x_{u,b} - \bar{x}_u)^2}}$$

  - Pearson Correlation

$$s_i^{\text{corr}}(i_a, i_b) = \frac{\sum_u (x_{u,a} - \bar{x}_a)(x_{u,b} - \bar{x}_b)}{\sqrt{\sum_u (x_{u,a} - \bar{x}_a)^2 \sum_u (x_{u,b} - \bar{x}_b)^2}}$$

Item-based kNN Rating Prediction

- Get top-*k* neighbor items that the target user *t* rated

Rank position

$$N_i(u_t, i_a) = \{i_b | r_i(i_a, i_b) < K^*, x_{t,b} \neq 0\}$$

Choose *K** such that $|N_i(u_t, i_a)| = k$

- Predict ratings for item *a* that the target user *t* did not rate

$$\hat{x}_{t,a} = \frac{\sum_{i_b \in N_i(u_t, i_a)} s_i(i_a, i_b) x_{t,b}}{\sum_{i_b \in N_i(u_t, i_a)} s_i(i_a, i_b)}$$

Don't need to correct for users average rating since query user itself is used to do predictions

Empirical Study

- Movielens dataset from grouplens
    - http://www.grouplens.org/node/73
    - Users visit Movielens
        - rate and receive recommendations for movies
    - Dataset (ML-100k)
        - 100k ratings from 1 to 5
        - 943 users, 1682 movies (rated by at least one user)
        - Sparsity level

$$1 - \frac{\#\text{non-zero entries}}{\text{total entries}} = 1 - \frac{10^5}{943 \times 1682} = 93.69\%$$

Experiment Setup

- Split data in training (*x*%) and test set ((100-*x*)%)
  - Can be repeated *T* times and results averaged

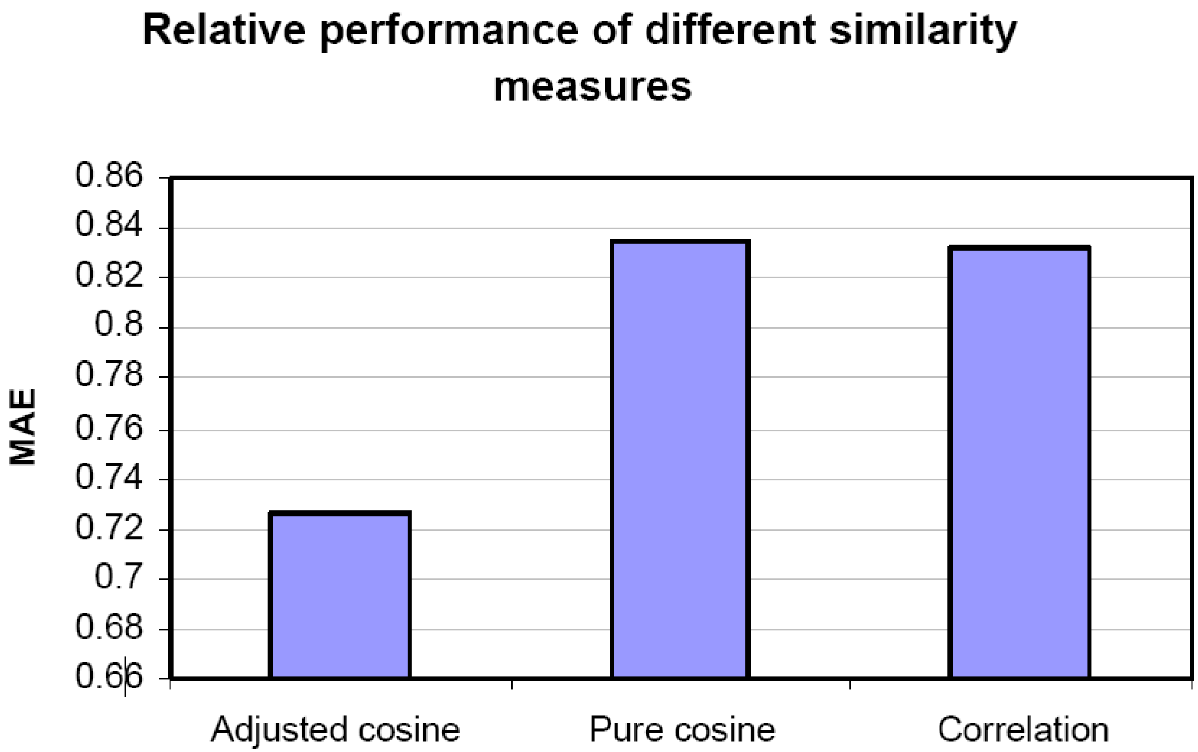
- Evaluation metrics
  - Mean-Absolute Error (MAE)

$$\mathrm{MAE} = \frac{1}{|D_{\mathrm{test}}|} \sum_{(u,i,r) \in D_{\mathrm{test}}} |r - \hat{r}_{u,i}|$$

  - Root Mean Squared Error (RMSE)

$$\mathrm{RMSE} = \sqrt{\frac{1}{|D_{\mathrm{test}}|} \sum_{(u,i,r) \in D_{\mathrm{test}}} (r - \hat{r}_{u,i})^2}$$

Impact of Similarity Measures

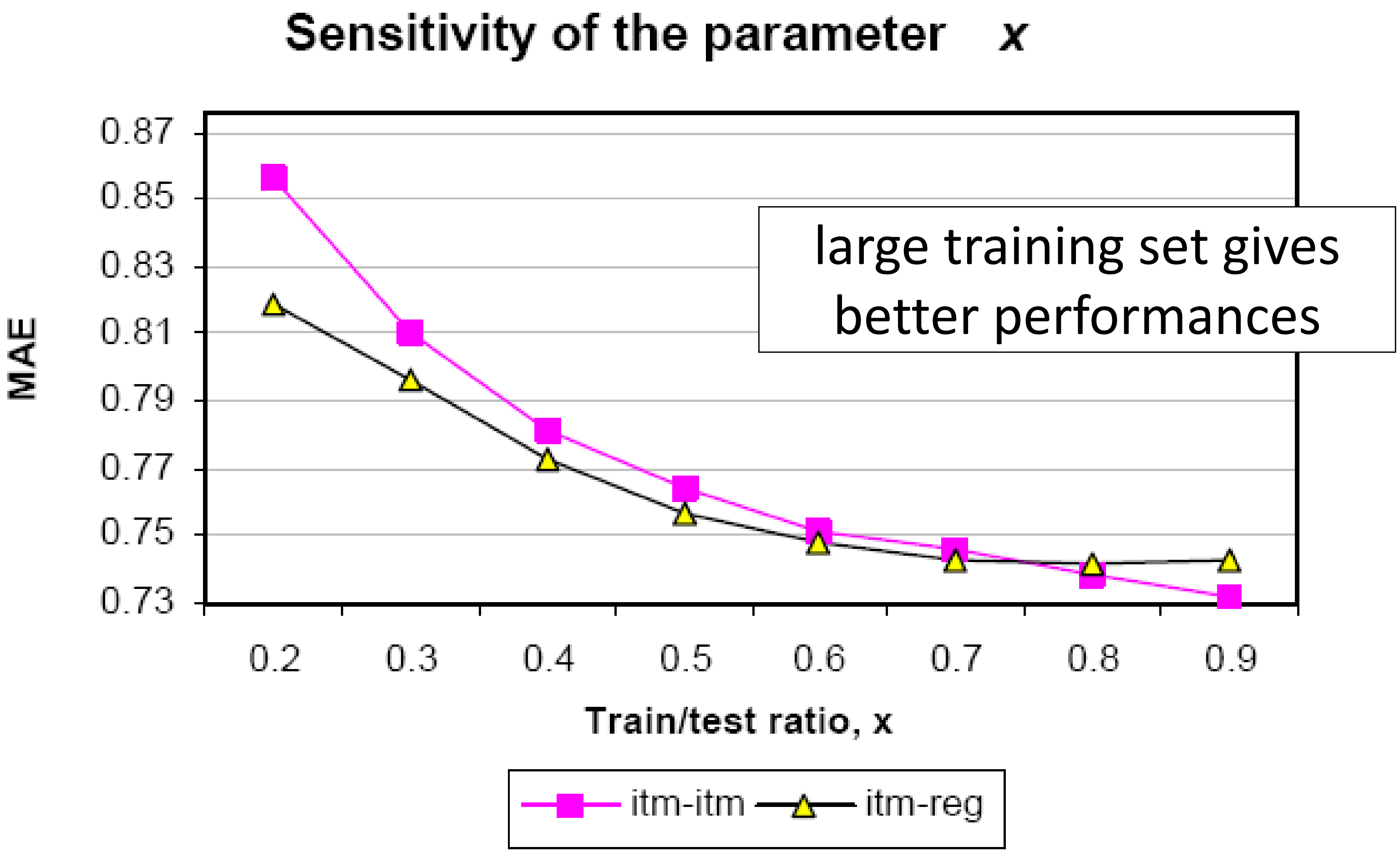


$$s_i^{\text{adcos}}(i_a, i_b) = \frac{\sum_u (x_{u,a} - \bar{x}_u)(x_{u,b} - \bar{x}_u)}{\sqrt{\sum_u (x_{u,a} - \bar{x}_u)^2 \sum_u (x_{u,b} - \bar{x}_u)^2}}$$

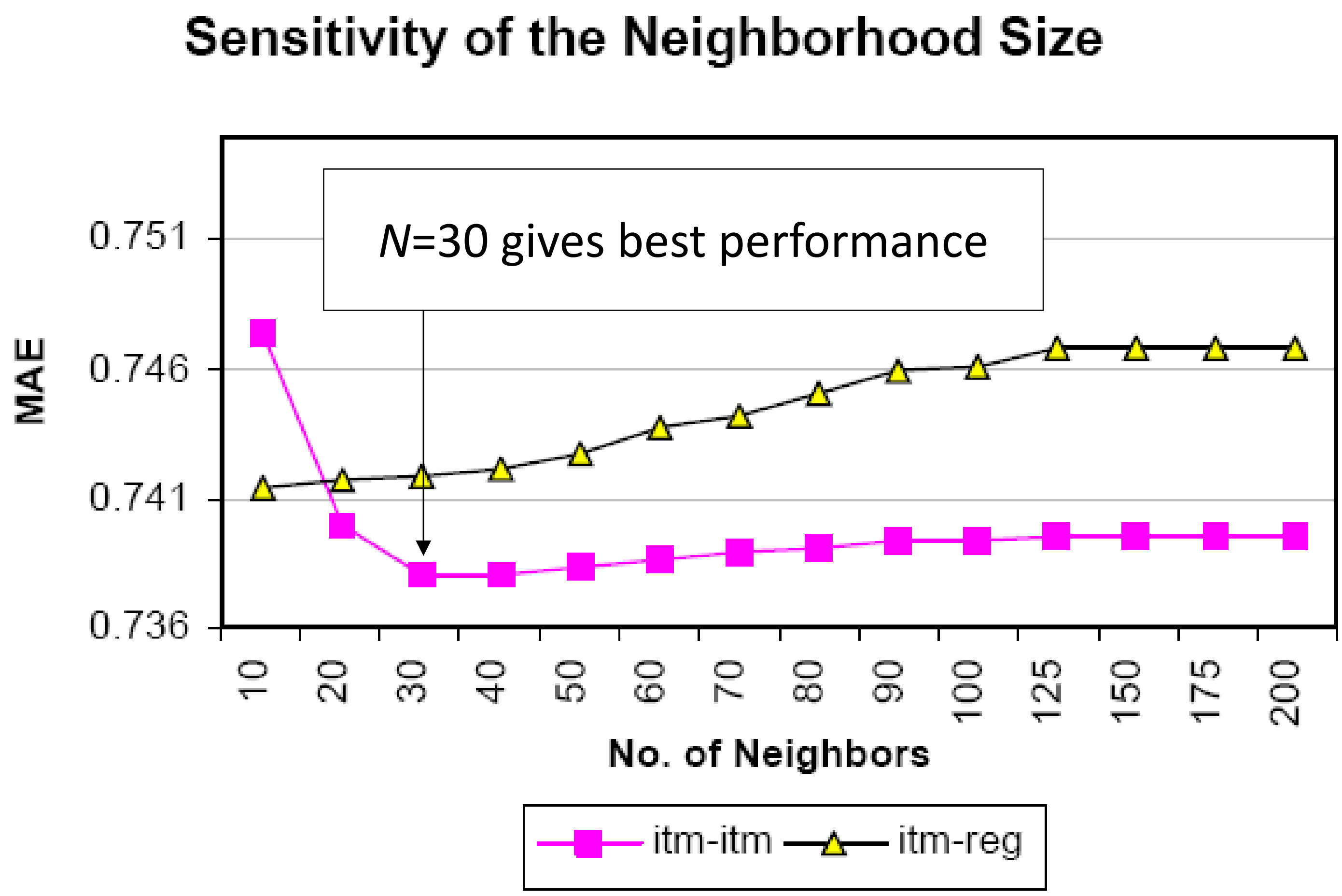
$$s_i^{\text{corr}}(i_a, i_b) = \frac{\sum_u (x_{u,a} - \bar{x}_a)(x_{u,b} - \bar{x}_b)}{\sqrt{\sum_u (x_{u,a} - \bar{x}_a)^2 \sum_u (x_{u,b} - \bar{x}_b)^2}}$$

$$s_i^{\cos}(i_a, i_b) = \frac{i_a^\top i_b}{\|i_a\| \|i_b\|} = \frac{\sum_u x_{u,a} x_{u,b}}{\sqrt{\sum_u x_{u,a}^2 \sum_u x_{u,b}^2}}$$

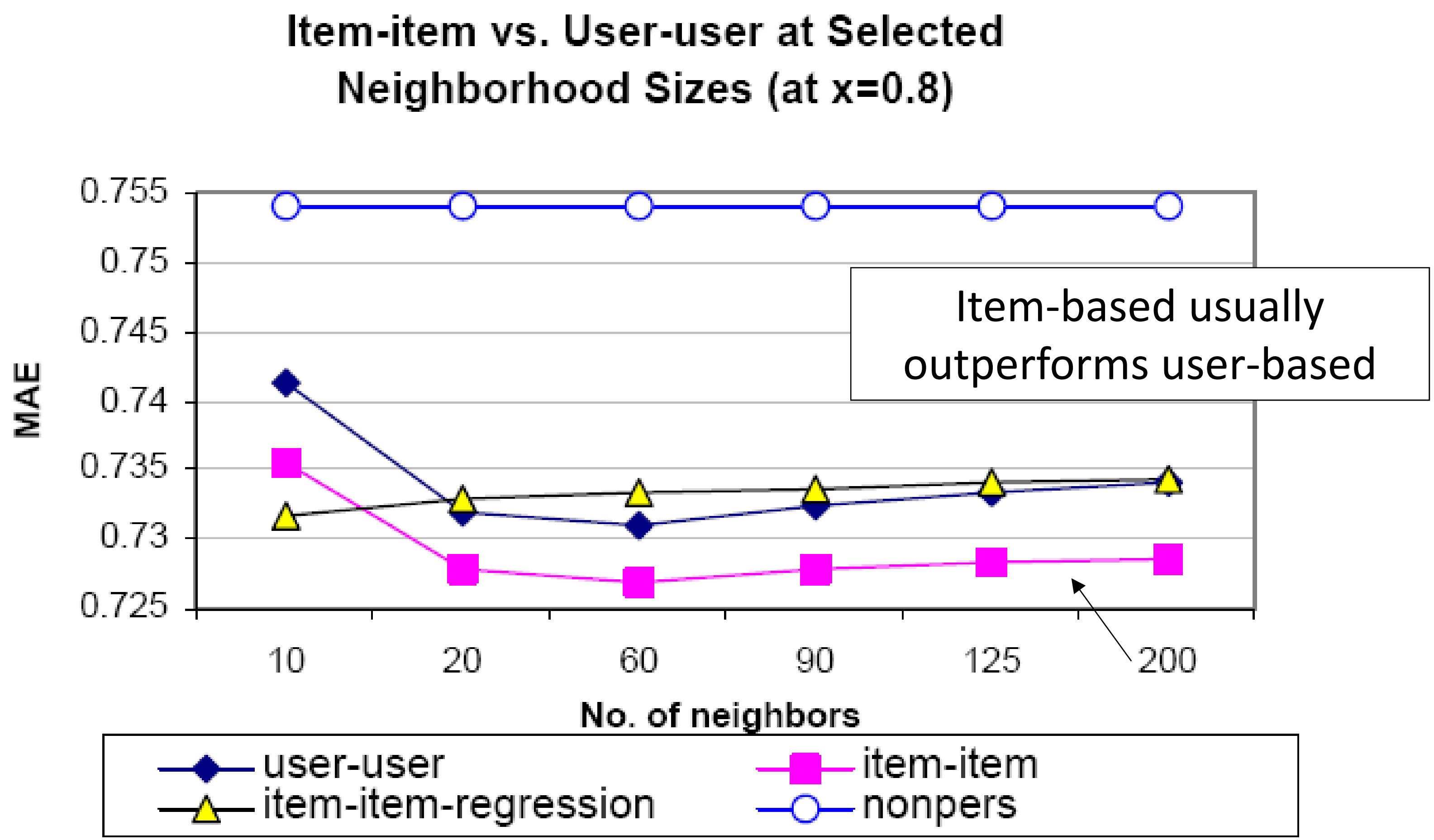
Sensitivity of Train/Test Ratio



Sensitivity Neighborhood Size *k*



Item-based vs. User-based



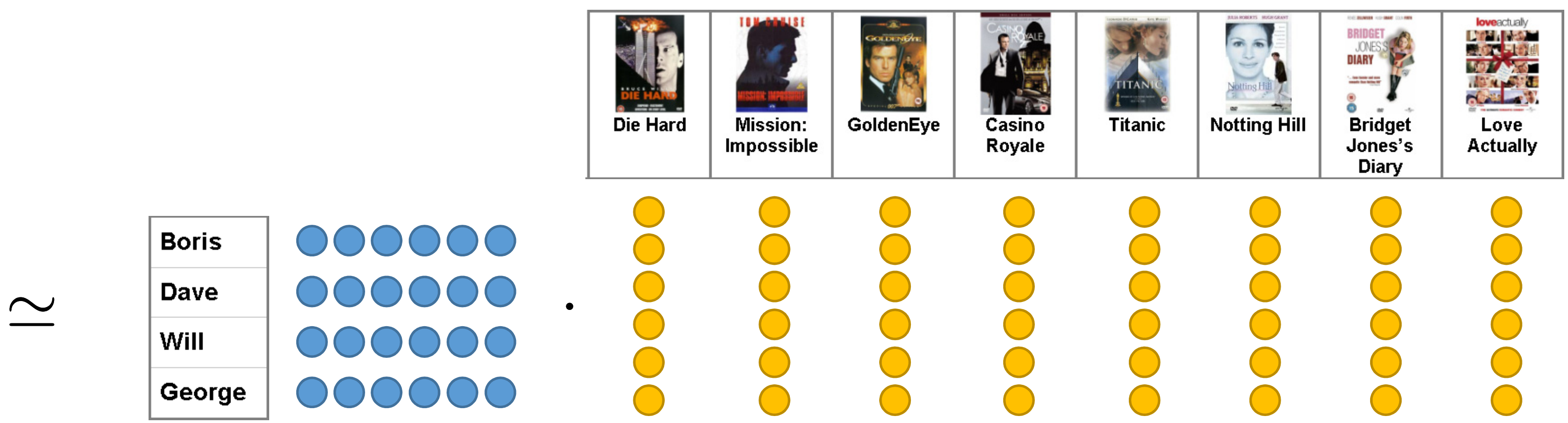
- Item-item similarity is usually more stable and objective

kNN based Methods Summary

- Straightforward and highly explainable
- No parameter learning
  - Only one hyperparameter *k* to tune
  - Cannot get improved by learning
- Efficiency could be a serious problem
  - When the user/item numbers are large
  - When there are a huge number of user-item ratings
- We may need a parametric and learnable model

Matrix Factorization Techniques

	Die Hard	Mission: Impossible	GoldenEye	Casino Royale	Titanic	Notting Hill	Bridget Jones's Diary	Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		★★☆☆☆
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆



Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." *Computer* 42.8 (2009).

Matrix Factorization Techniques

Items

Users												
	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	

$$\hat{r}_{u,i} = p_u^\top q_i$$

$\simeq$

Users

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

$\cdot$

Items

				i							
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

Basic MF Model

- Prediction of user *u*'s rating on item *i*

$$\hat{r}_{u,i} = p_u^\top q_i$$ ← Bilinear model

- Loss function

$$\mathcal{L}(u, i, r_{u,i}) = \frac{1}{2}(r_{u,i} - p_u^\top q_i)^2$$

- Training objective

$$\min_{P,Q} \sum_{r_{u,i} \in D} \frac{1}{2}(r_{u,i} - p_u^\top q_i)^2 + \frac{\lambda}{2}(\|p_u\|^2 + \|q_i\|^2)$$

- Gradients

$$\frac{\partial \mathcal{L}(u, i, r_{u,i})}{\partial p_u} = (p_u^\top q_i - r_{u,i})q_i + \lambda p_u$$

$$\frac{\partial \mathcal{L}(u, i, r_{u,i})}{\partial q_i} = (p_u^\top q_i - r_{u,i})p_u + \lambda q_i$$

MF with Biases

- Prediction of user *u*'s rating on item *i*

$$\hat{r}_{u,i} = \mu + b_u + b_i + p_u^\top q_i$$

Global bias | User bias | Item bias | User-item Interaction

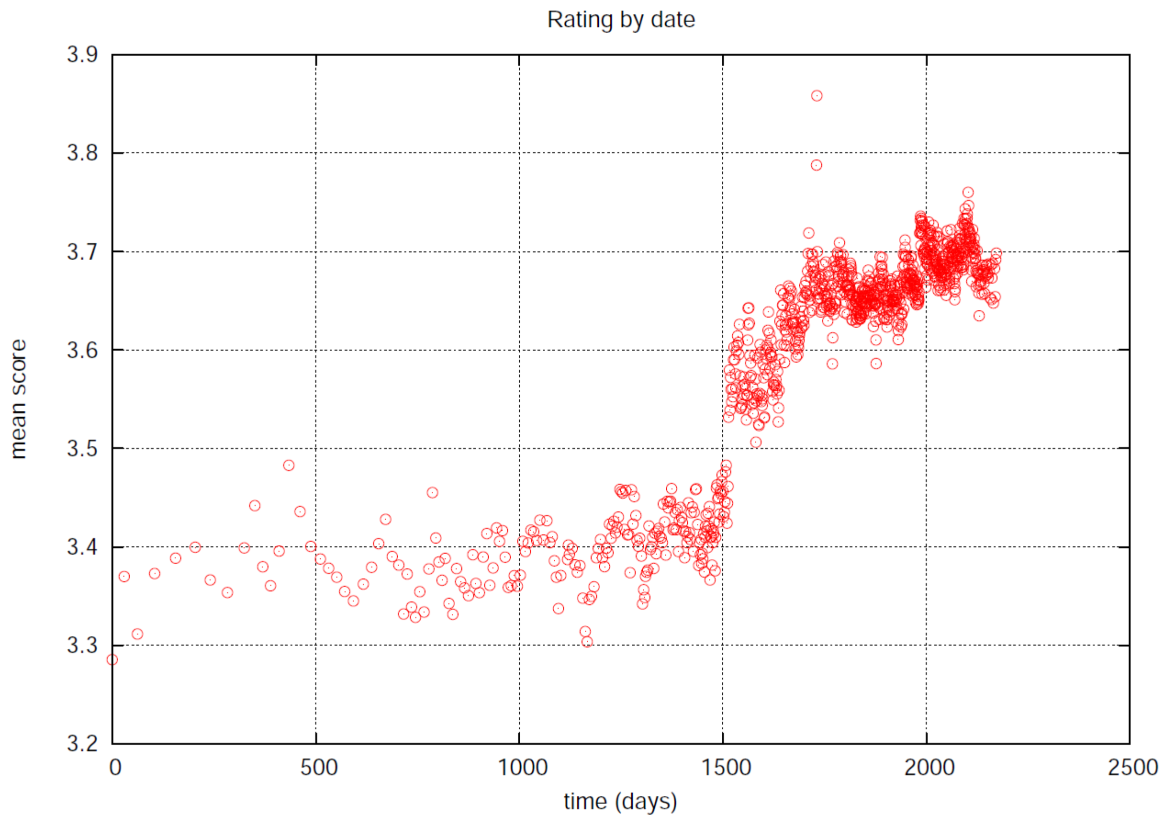
- Training objective

$$\min_{P,Q} \sum_{r_{u,i} \in D} \frac{1}{2}\Big(r_{u,i} - (\mu + b_u + b_i + p_u^\top q_i)\Big)^2 + \frac{\lambda}{2}(\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2)$$

- Gradient update

$$\begin{aligned}
\delta &= r_{u,i} - (\mu + b_u + b_i + p_u^\top q_i) \\
\mu &\leftarrow \mu + \eta\delta \\
b_u &\leftarrow (1 - \eta\lambda) b_u + \eta\delta \\
b_i &\leftarrow (1 - \eta\lambda) b_i + \eta\delta \\
p_u &\leftarrow (1 - \eta\lambda) p_u + \eta\delta q_i \\
q_i &\leftarrow (1 - \eta\lambda) q_i + \eta\delta p_u
\end{aligned}$$

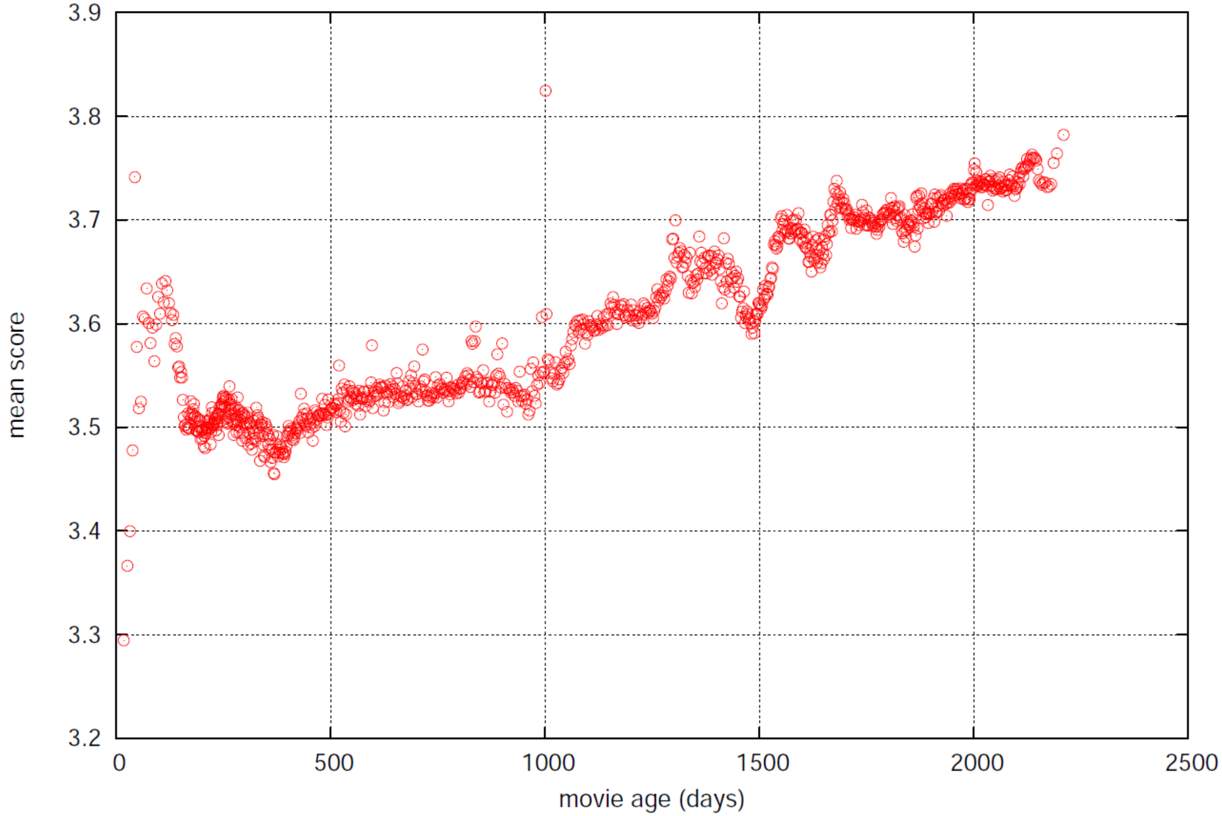
Temporal Dynamics



- A sudden rise in the average movie rating begging around 1500 days (early 2004) into the dataset

Koren, Yehuda. "Collaborative filtering with temporal dynamics." *KDD 2009*.

Temporal Dynamics



- People tend to give higher ratings as movies become older

Koren, Yehuda. "Collaborative filtering with temporal dynamics." *KDD 2009*.

Multiple sources of temporal dynamics

- Item-side effects
  - Product perception and popularity are constantly changing
  - Seasonal patterns influence items’ popularity

- User-side effects
  - Customers ever redefine their taste
  - Transient, short-term bias
  - Drifting rating scale
  - Change of rater within household

Addressing temporal dynamics

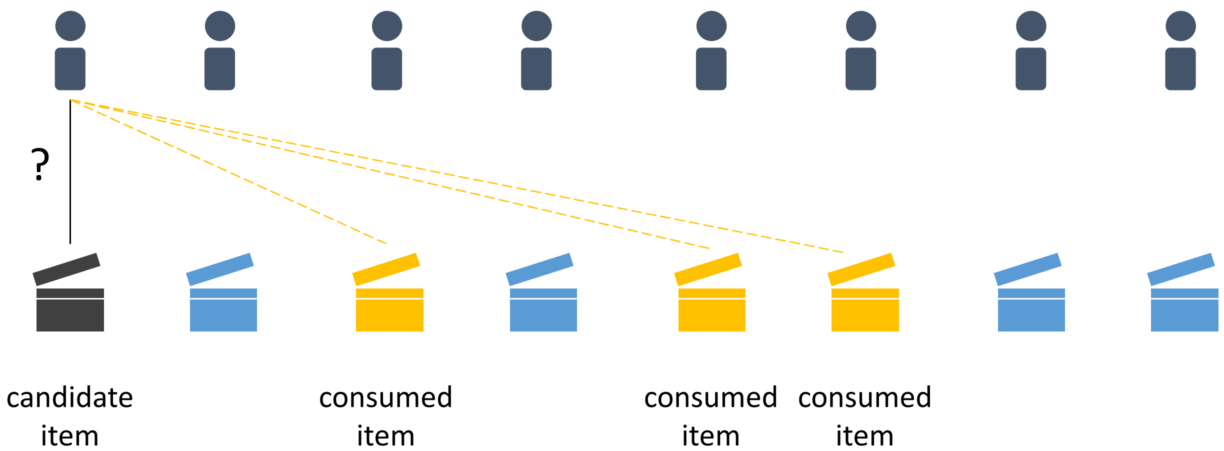
- Factor model conveniently allows separately treating different aspects
- We observe changes in:
  - Rating scale of individual users $b_u(t)$
  - Popularity of individual items $b_i(t)$
  - User preferences $p_u(t)$

$$r_{u,i}(t) = \mu + b_u(t) + b_i(t) + p_u(t)^\top q_i$$

- Design guidelines
  - Items show slower temporal changes
  - Users exhibit frequent and sudden changes
  - Factors $p_u(t)$ are expensive to model
  - Gain flexibility by heavily parameterizing the functions

Neighborhood (Similarity)-based MF

- Assumption: user's previous consumed items reflect her taste
- Derive unknown ratings from those of "similar" items (item-item variant)



Neighborhood based MF modeling: SVD++

$$\hat{r}_{u,i} = b_{u,i} + q_i^\top \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right)$$

need to estimate rating of user u for item i

bias

item vector

user vector

Set of user previously rated items

item vector for neighborhood similarity

- Each item has two latent vectors
  - The standard item vector q_i
  - The vector y_i when it is used for estimation the similarity between the candidate item and the target user

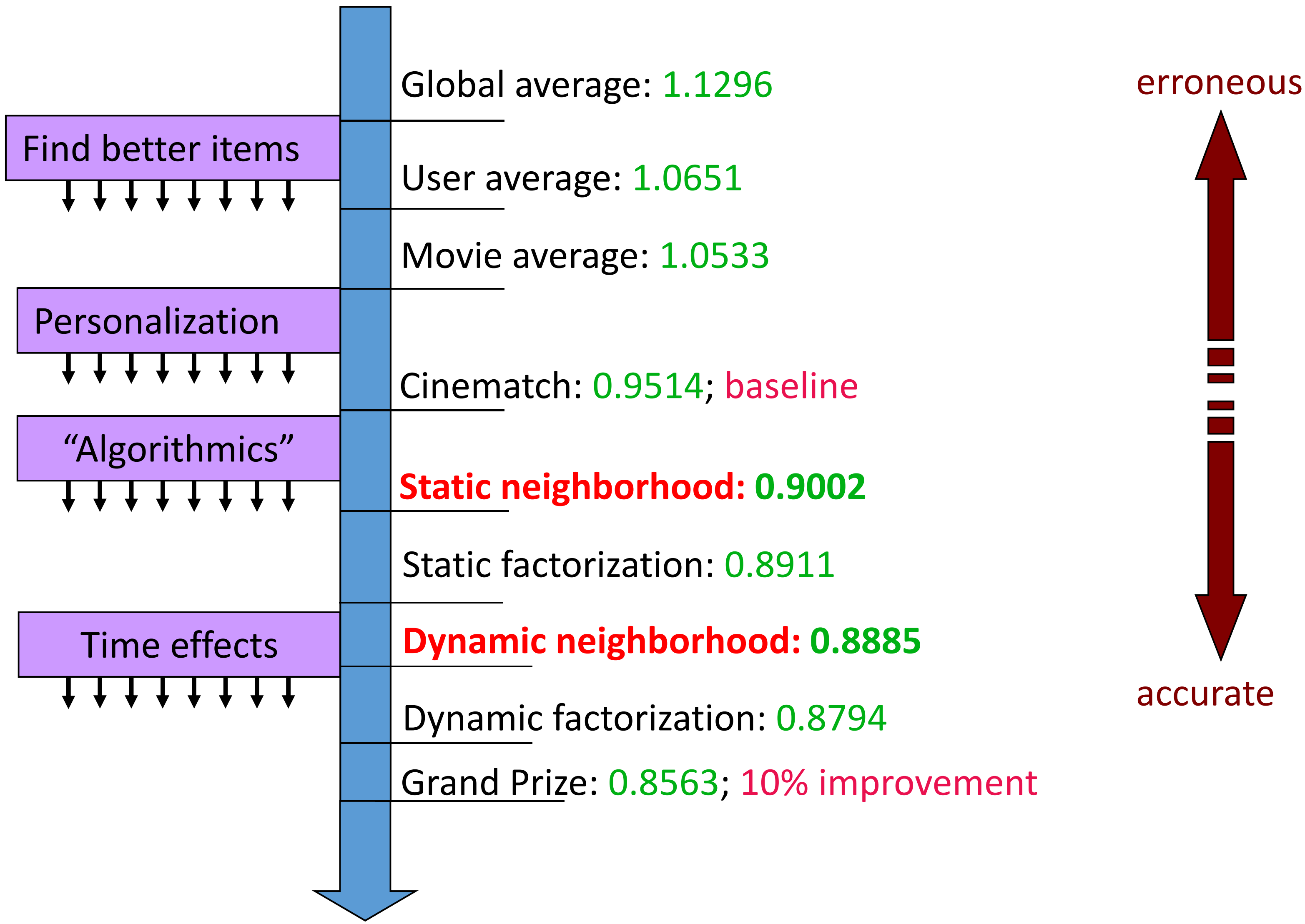
Koren, Yehuda. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." *KDD*, 2008.

Netflix Prize

- An open competition for the best collaborative filtering algorithm for movies
  - Began on October 2, 2006.
  - A million-dollar challenge to improve the accuracy (RMSE) of the Netflix recommendation algorithm by 10%

- Netflix provided
  - Training data: 100,480,507 ratings:
  - 480,189 users x 17,770 movies.
  - Format: <user, movie, date, rating>

- Two popular approaches:
  - Matrix factorization
  - Neighborhood



Temporal neighborhood model delivers same relative RMSE improvement (0.0117) as temporal factor model (!)

2009
NETFLIX
DATE: 09.21.09
PAY TO THE ORDER OF: BellKor's Pragmatic Chaos
$1,000,000.00
AMOUNT: ONE MILLION 00/100
FOR: The Netflix Prize
Reed Hastings
The Netflix Prize

Feature-based Matrix Factorization

$$\hat{y} = \mu + \left(\sum_j b_j^{(g)}\gamma_j + \sum_j b_j^{(u)}\alpha_j + \sum_j b_j^{(i)}\beta_j\right) + \left(\sum_j p_j\alpha_j\right)^\top \left(\sum_j q_j\beta_j\right)$$

- Regard all information as features
  - User id and item id
  - Time, item category, user demographics etc.
- User and item features are with latent factors

T. Chen et al. Feature-based matrix factorization. arXiv:1109.2271
http://svdfeature.apexlab.org/wiki/images/7/76/APEX-TR-2011-07-11.pdf

Open source: http://svdfeature.apexlab.org/wiki/Main_Page

Factorization Machine

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$$

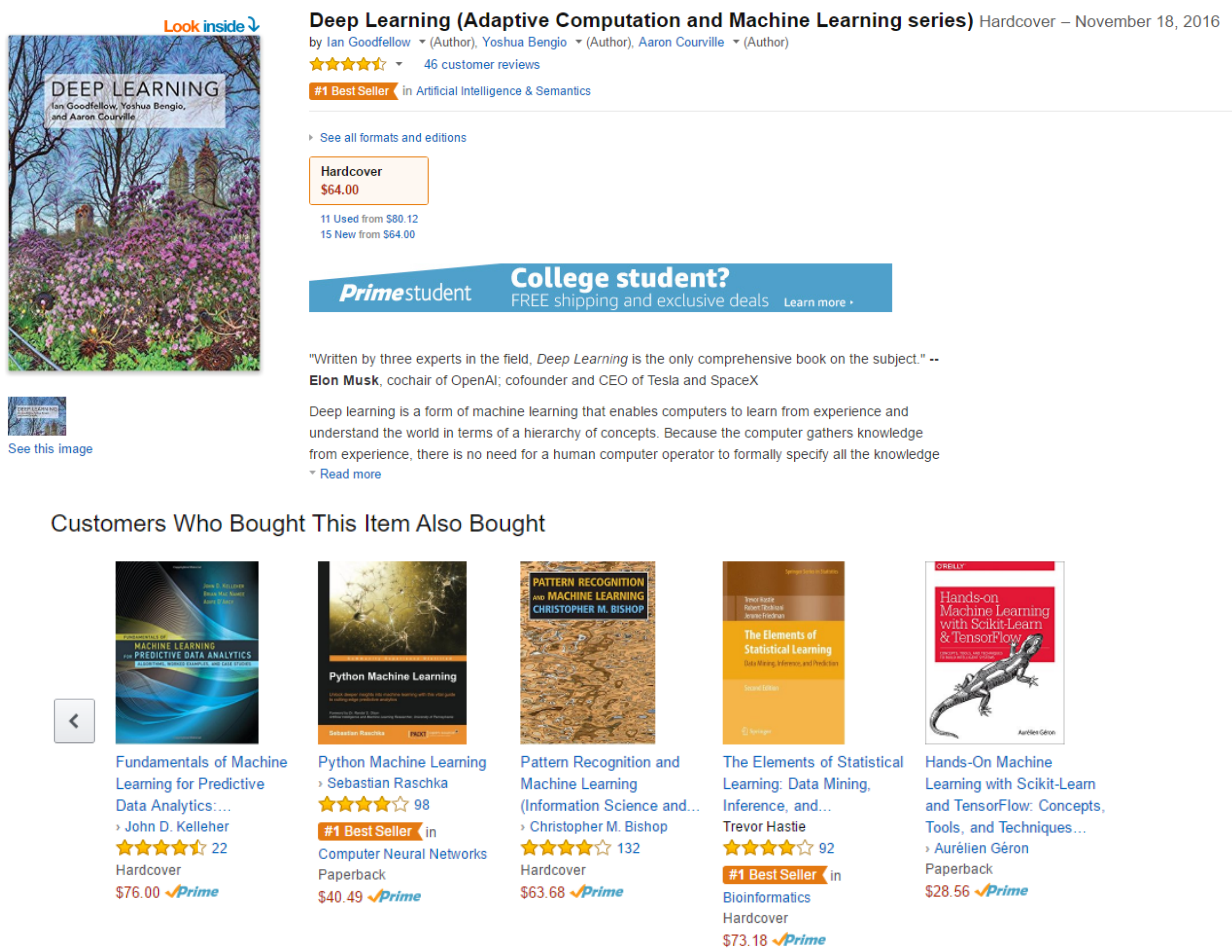
- One-hot encoding for each discrete (categorical) field
- One real-value feature for each continuous field
- All features are with latent factors
- A more general regression model

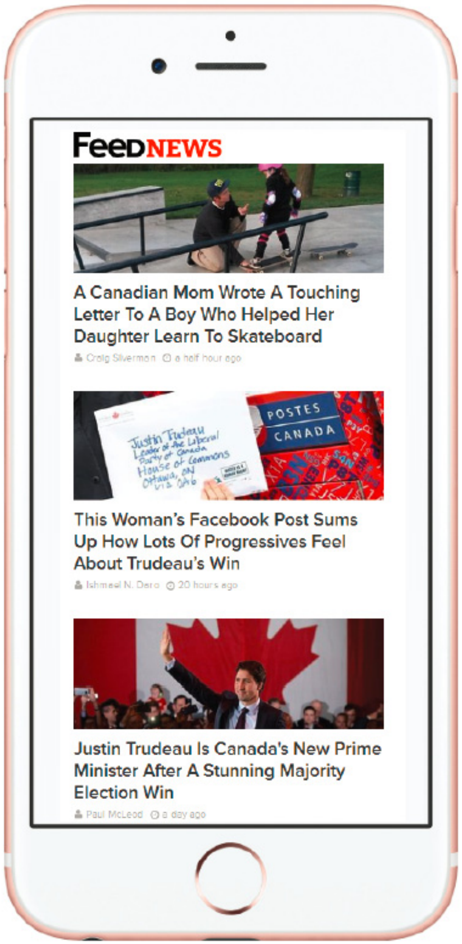
Steffen Rendle. Factorization Machines. ICDM 2010
http://www.ismll.uni-hildesheim.de/pub/pdfs/Rendle2010FM.pdf
Open source: http://www.libfm.org/

Beyond Rating Prediction

LambdaRank CF

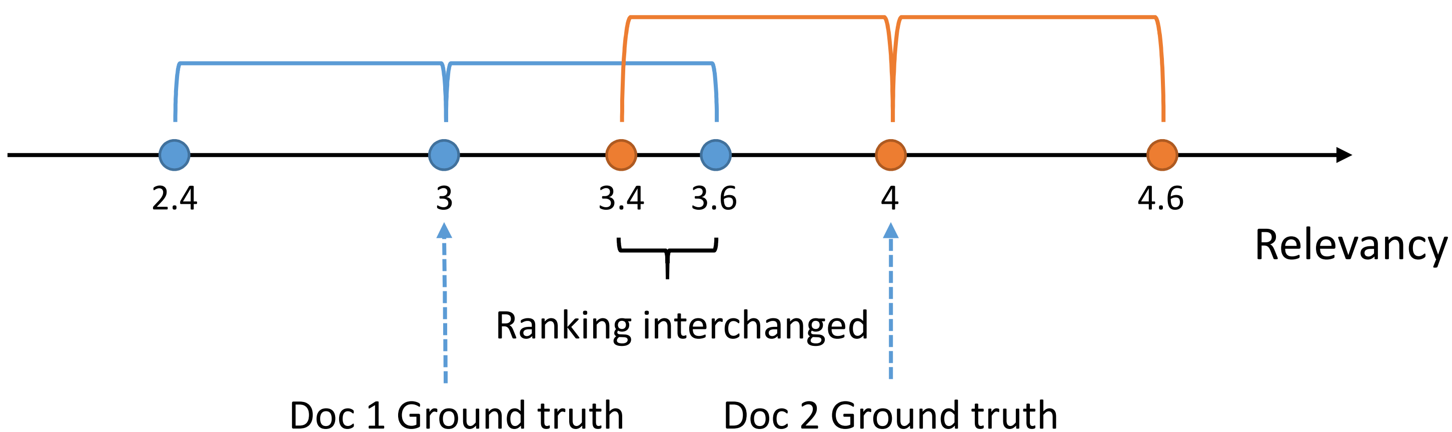
Recommendation is always rendered by ranking





Rating Prediction vs. Ranking

- Rating prediction may not be a good objective for top-N recommendation (i.e. item ranking)



- Same RMSE/MAE might lead to different rankings

Learning to Rank in Collaborative Filtering

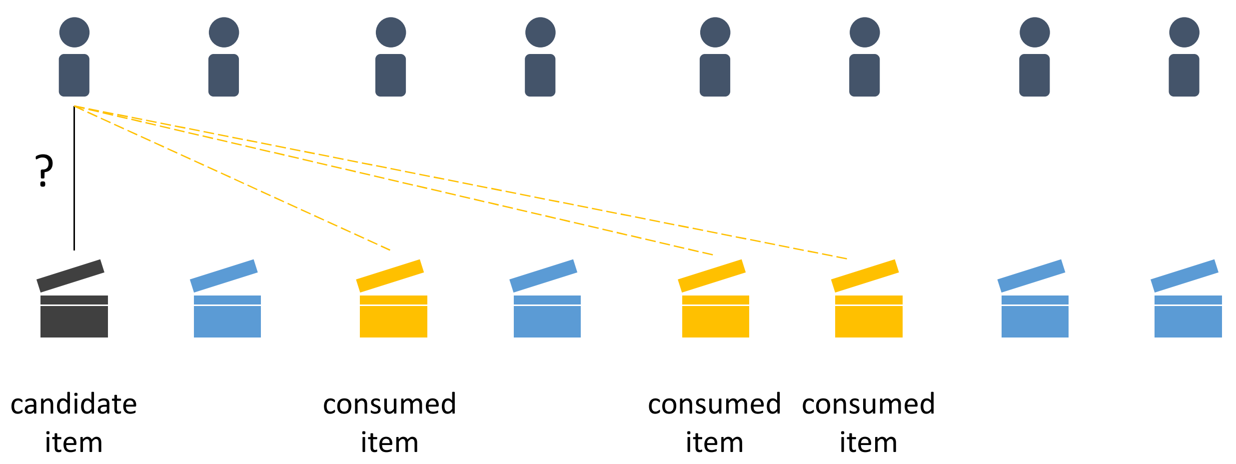
- Previous work on rating prediction can be regarded as pointwise approaches in CF
  - MF, FM, kNN, MF with temporal dynamics and neighborhood information etc.

- Pairwise approaches in CF
  - Bayesian personalized ranking (BPR)

- Listwise approaches in CF
  - LambdaRank CF, LambdaFM

Implicit Feedback Data

- No explicit preference, e.g. rating, shown in the user-item interaction
  - Only clicks, share, comments etc.



Bayesian Personalized Ranking (BPR)

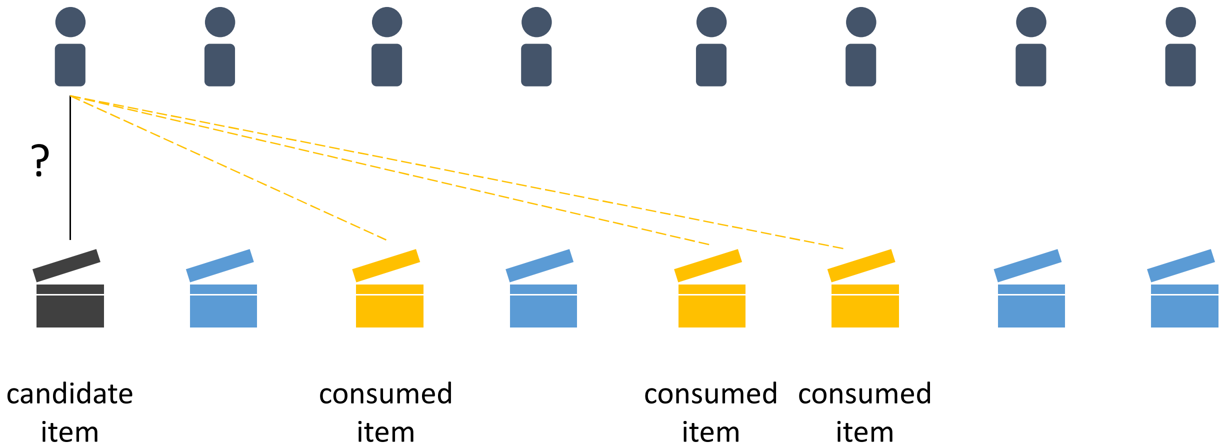
- Basic latent factor model (MF) for scoring

$$\hat{r}_{u,i} = \mu + b_u + b_i + p_u^\top q_i$$

- The (implicit feedback) training data for each user *u*

$$D_u = \{\langle i, j \rangle_u | i \in I_u \wedge j \in I \backslash I_u\}$$

consumed item   unconsumed item



Rendle, Steffen, et al. "BPR: Bayesian personalized ranking from implicit feedback." *UAI*, 2009.

Bayesian Personalized Ranking (BPR)

- Loss function on the ranking prediction of $<i,j>_u$

$$\mathcal{L}(\langle i,j\rangle_u) = \underbrace{z_u}_{\text{Normalizer}} \cdot \underbrace{\frac{1}{1+\exp(\hat{r}_{u,i}-\hat{r}_{u,j})}}_{\text{Inverse logistic loss}}$$

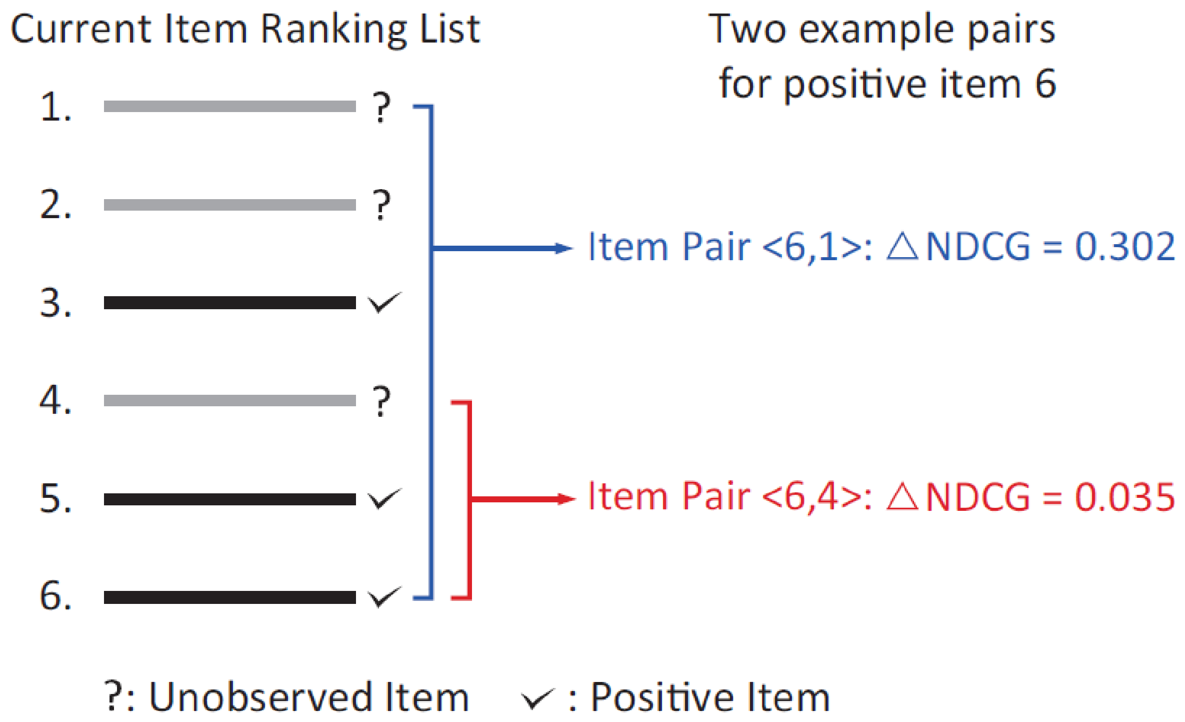
- Gradient

$$\begin{aligned}\frac{\partial \mathcal{L}(\langle i,j\rangle_u)}{\partial \theta} &= \frac{\partial \mathcal{L}(\langle i,j\rangle_u)}{\partial(\hat{r}_{u,i}-\hat{r}_{u,j})}\frac{\partial(\hat{r}_{u,i}-\hat{r}_{u,j})}{\partial \theta} \\ &\equiv \lambda_{i,j}\left(\frac{\partial \hat{r}_{u,i}}{\partial \theta}-\frac{\partial \hat{r}_{u,j}}{\partial \theta}\right)\end{aligned}$$

Rendle, Steffen, et al. "BPR: Bayesian personalized ranking from implicit feedback." *UAI*, 2009.

LambdaRank CF

- Use the idea of LambdaRank to optimize ranking performance in recommendation tasks



Zhang, Weinan, et al. "Optimizing top-n collaborative filtering via dynamic negative item sampling." *SIGIR*, 2013.

Recommendation vs. Web Search

- Difference between them
  - Recommender system should rank all the items
    - Usually more than 10k
  - Search engine only ranks a small subset of retrieved documents
    - Usually fewer than 1k
- For each training iteration, LambdaRank needs the model to rank all the items to get $\Delta\text{NDCG}_{i,j}$, super large complexity

Zhang, Weinan, et al. "Optimizing top-n collaborative filtering via dynamic negative item sampling." *SIGIR*, 2013.

LambdaRank CF Solution

- Idea: to generate the item pairs with the probability proportional to their lambda

$$\frac{\partial \mathcal{L}(\langle i,j \rangle_u)}{\partial \theta} = f(\lambda_{i,j}, \zeta_u)\Big(\frac{\partial \hat{r}_{u,i}}{\partial \theta} - \frac{\partial \hat{r}_{u,j}}{\partial \theta}\Big)$$

$$f(\lambda_{i,j}, \zeta_u) \equiv \lambda_{i,j} \Delta NDCG_{i,j}$$

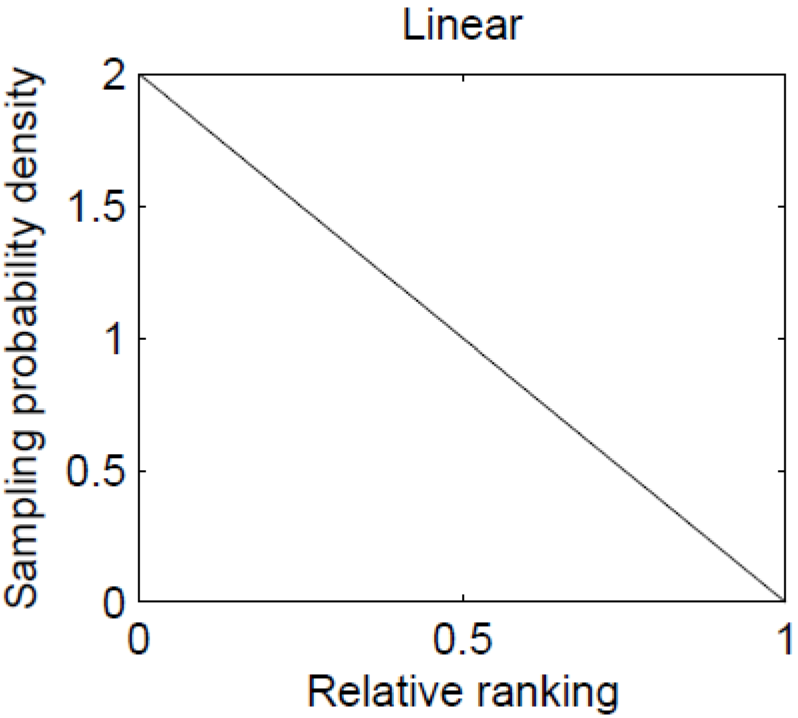
$$p_j \propto f(\lambda_{i,j}, \zeta_u)/\lambda_{i,j}$$

- $x_i \in [0, 1]$ is the relative ranking position
  - 0 means ranking at top, 1 means ranking at tail

Zhang, Weinan, et al. "Optimizing top-n collaborative filtering via dynamic negative item sampling." *SIGIR*, 2013.

Different Sampling Methods

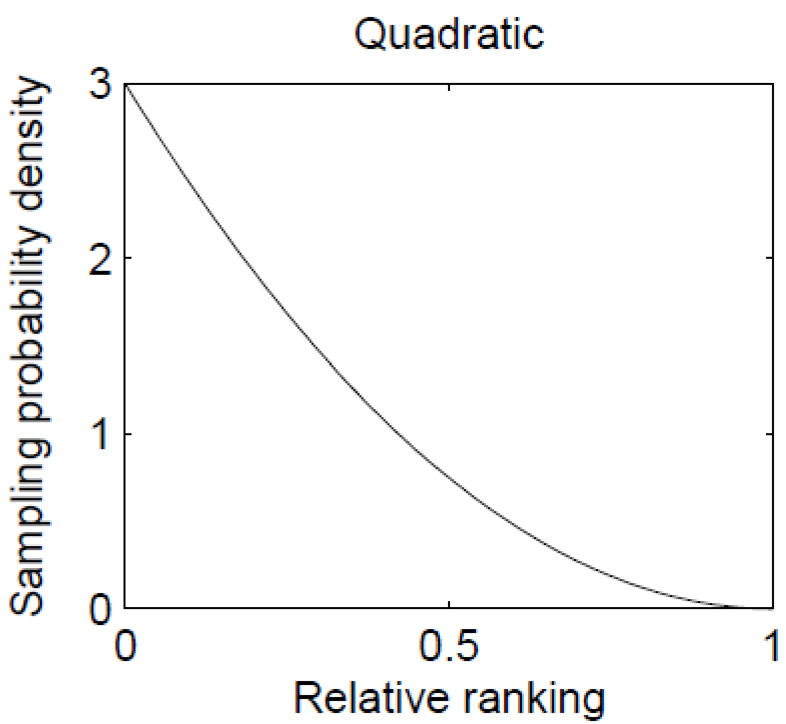
- For each positive item, find 2 candidate items, then choose the one with higher prediction score as the negative item.



Zhang, Weinan, et al. "Optimizing top-n collaborative filtering via dynamic negative item sampling." *SIGIR*, 2013.

Different Sampling Methods

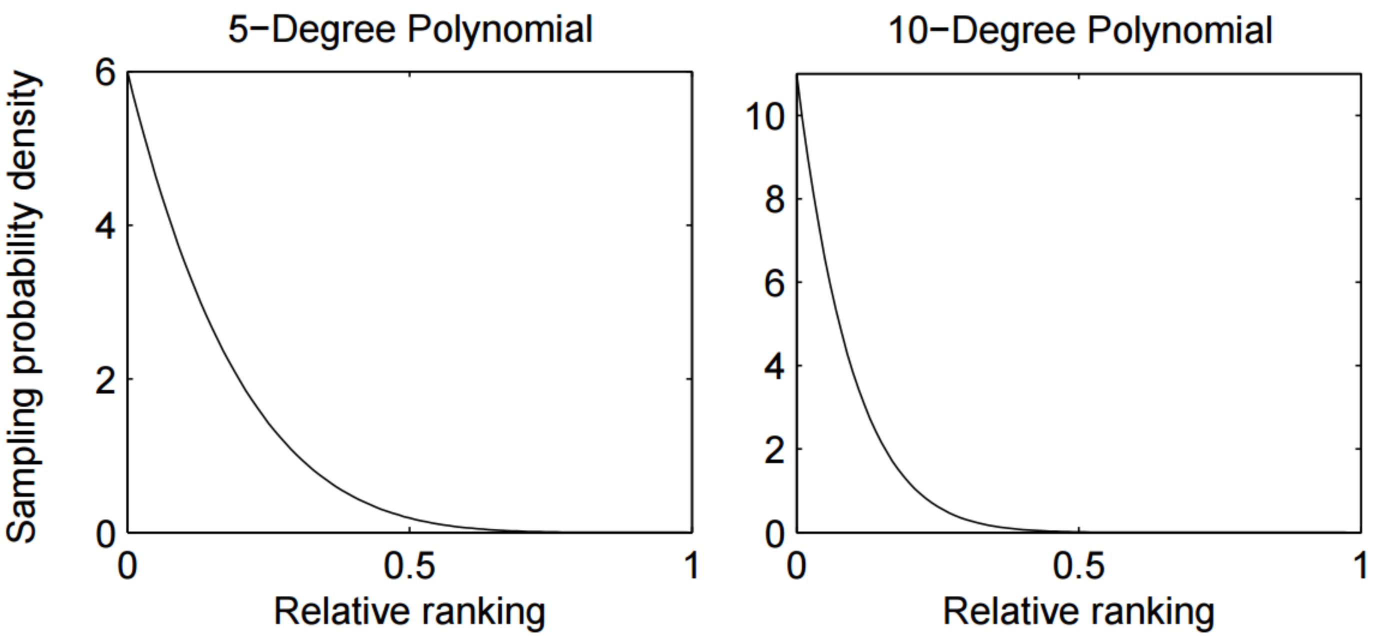
- For each positive item, find **3** candidate items, then choose the one with the **highest** prediction score as the negative item.



Zhang, Weinan, et al. "Optimizing top-n collaborative filtering via dynamic negative item sampling." *SIGIR*, 2013.

Different Sampling Methods

- For each positive item, find ***k*** candidate items, then choose the one with the **highest** prediction score as the negative item.



Zhang, Weinan, et al. "Optimizing top-n collaborative filtering via dynamic negative item sampling." *SIGIR*, 2013.

Experiments on Top-N Recommendation

- Top-N recommendation on 3 datasets

Dataset	Netflix	Yahoo! Music	Last.fm
Users	480,189	1,000,990	992
Items	17,770	624,961	961,417
Ratings	100,480,507	262,810,175	19,150,868

- Performance (DNS is our LambdaCF algorithm)

Netflix

	P@5	P@10	NDCG@5	NDCG@10	MAP
BPR	0.3826	0.3272	0.2052	0.2017	0.1403
DNS	0.4708	0.4012	0.2906	0.2887	0.2036
Impv.	23.1%*	22.6%*	41.6%*	43.1%*	45.1%*

Yahoo! Music

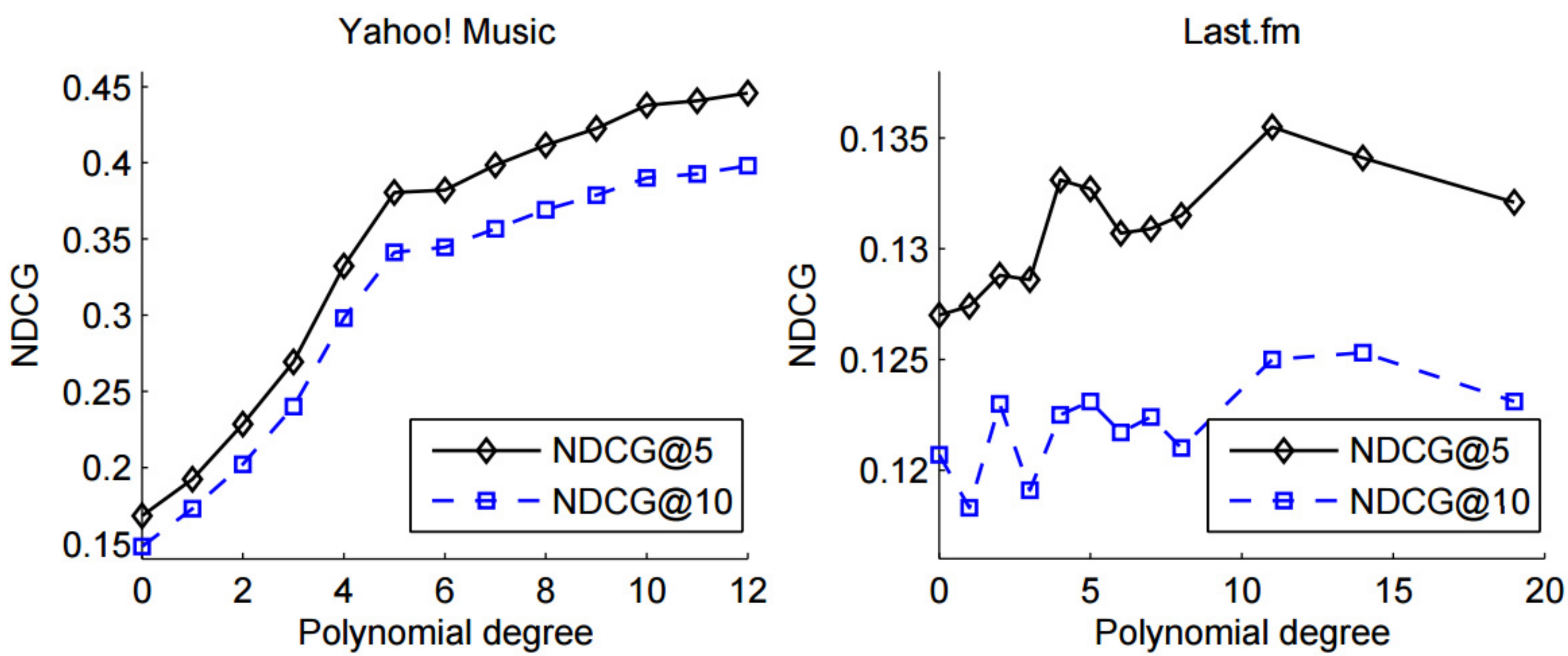
	P@5	P@10	NDCG@5	NDCG@10	MAP
BPR	0.1588	0.1359	0.1683	0.1481	0.0615
DNS	0.4243	0.3671	0.4458	0.3981	0.1644
Impv.	167.2%*	170.1%*	164.9%*	168.8%*	167.3%*

Last.fm

	P@5	P@10	NDCG@5	NDCG@10	MAP
BPR	0.1231	0.1168	0.1270	0.1207	0.0221
DNS	0.1323	0.1202	0.1355	0.1250	0.0223
Impv.	7.5%*	2.9%	6.7%*	3.6%	0.9%

Zhang, Weinan, et al. "Optimizing top-n collaborative filtering via dynamic negative item sampling." *SIGIR*, 2013.

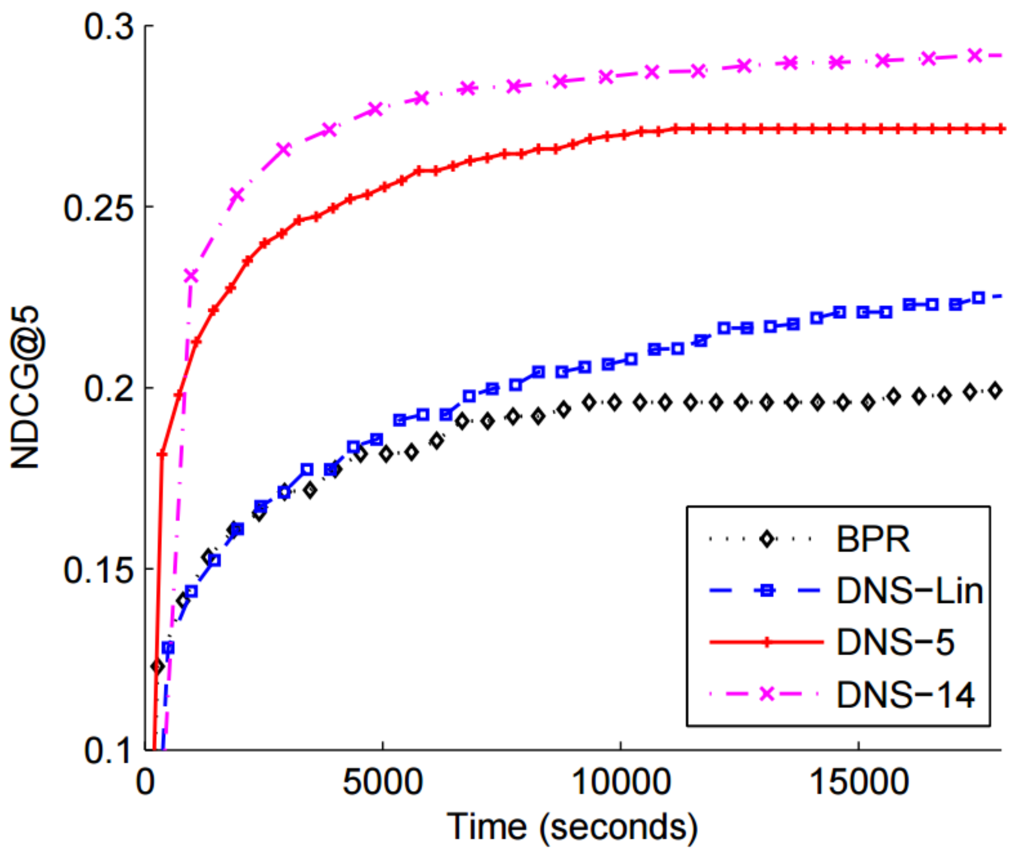
More Empirical Results



- NDCG performance against polynomial degrees on Yahoo! Music and Last.fm datasets

Zhang, Weinan, et al. "Optimizing top-n collaborative filtering via dynamic negative item sampling." *SIGIR*, 2013.

More Empirical Results



Performance convergence against training time on Netflix.

Zhang, Weinan, et al. "Optimizing top-n collaborative filtering via dynamic negative item sampling." *SIGIR*, 2013.