NUS | Computing
National University of Singapore

CS5228: Knowledge Discovery and Data Mining

Lecture 8 — Recommender Systems

Course Logistics

- Deadline Reminders
  - Submission of A3: Thu Oct 24, 11.59 pm – answer please in English only :)
  - Submission of project report: Nov 14, 11.59 pm
  - Extended TEAMMATES deadline: Sat, Oct 19, 11.59pm
- Marks upload
  - A2 + midterm result soon ready

Quick Recap — Linear Models

- Basic Assumption
  - Linear relationship between x_i and dependent variable y_i

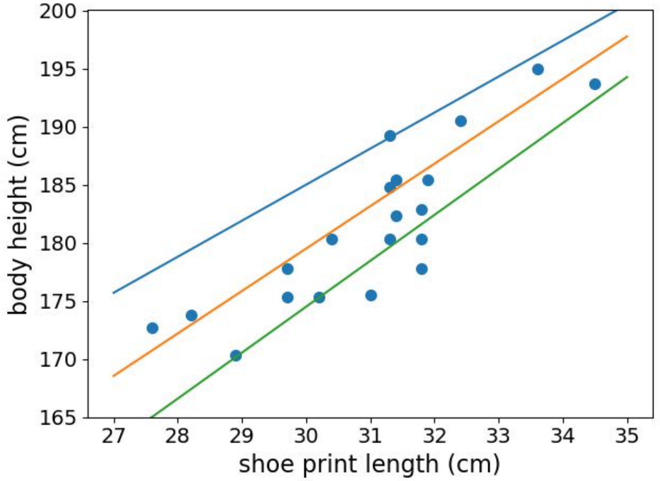
$$\hat{y}_i = h_\theta(x_i) = f(\theta_0 \underbrace{x_{i0}}_{=1} + \theta_1 x_{i1} + \theta_2 x_{i2} + \ldots + \theta_d x_{id})$$

Predicted value which is hopefully close to y_i

1, 2, …, d input features

  - Learned parameters of model: $\theta = \{\theta_0, \theta_1, \theta_2, \ldots, \theta_d\},\ \theta_i \in \mathbb{R}$

Quick Recap — Linear Regression



- Find θ that minimizes MSE loss L

$$L = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$
$$= \frac{1}{n}\|X\theta - y\|^2$$

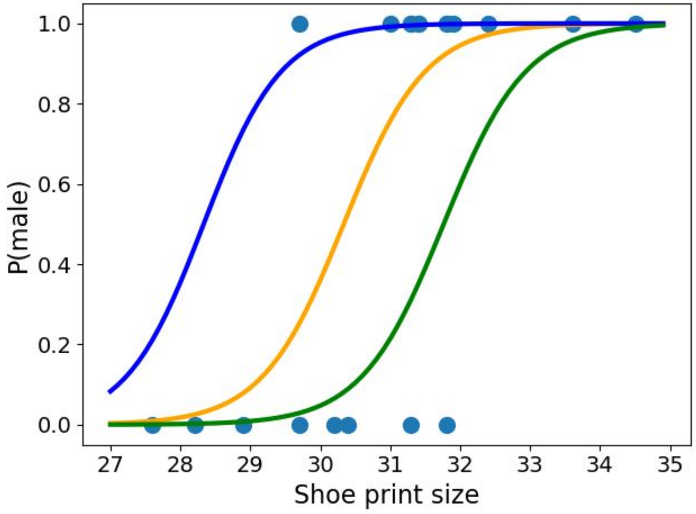
- Solve using Normal Equation

$$\frac{\partial L}{\partial \theta} = \frac{2}{n}X^T(X\theta - y) \quad \rightarrow \quad \frac{2}{n}X^T(X\theta - y) \stackrel{!}{=} \vec{0} \quad \rightarrow \quad \theta = (X^TX)^{-1}X^Ty$$

- Solve using Gradient Descent

$$\nabla_\theta L = \frac{2}{n}X^T(X\theta - y) \quad \rightarrow \quad \text{repeat:} \quad \theta \leftarrow \theta - (\eta \cdot \nabla_\theta L)$$

Quick Recap — Logistic Regression



- Regression model for classification
  - Interpret $\hat{y}$ as probability that *x* belongs to Class 1

$$\hat{y} = h_\theta(x) = f(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

  - Minimize **Cross-Entropy Loss** L

$$L = -\left[y \log \hat{y} + (1 - y) \log (1 - \hat{y})\right]$$

  - Solve using **Gradient Descent**

$$\nabla_\theta L = \frac{1}{n} X^T (h_\theta(X) - y)$$ ➜ repeat: $\theta \leftarrow \theta - (\eta \cdot \nabla_\theta L)$

Quick Recap — Linear Models

- Polynomial Linear/Logistic Regression
  - Data transformation to include polynomial terms of features
  - No change of algorithms needed

$$X^{(1)} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \quad X^{(2)} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \quad X^{(3)} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix}$$

- Regularization to avoid overfitting
  - Extend loss function to "punish" large values of θ
  - Only minor changes to Normal Equation and calculation of Gradient

$$L = \underbrace{\frac{1}{n}\|X\theta - y\|^2}_{\text{good match}} + \underbrace{\boxed{\lambda \frac{1}{n}\|\theta\|_2^2}}_{\text{no extreme } \theta \text{ values}}$$

Outline

- **Overview**
  - **Motivation**
  - Naive / alternative approaches
- Content-based recommendation systems
  - Pairwise item-item similarity
  - User-item similarity
- Collaborative filtering (CF)
  - Memory-based CF
  - Model-based CF

Recommender Systems — Motivation

- In the online world: information and item overload
  - Too many items: products, songs, movies, news articles, restaurants, etc.
  - More choices require better filters ➜ recommendation engines

User perspective	Provider perspective
• Identify relevant items • Minimize effort (to find relevant items) • Maximize satisfaction • Optimize spending of money and attention	• Maximize sales / transactions • Maximize user engagement (e.g., to maximize ad revenue) • Gain competitive advantage

Editorial Recommendations

- Recommendations by "experts"
  - Expert = person with expertise about item(s)
    (e.g., movie or restaurant critic, staff writers, journalists)
  - Objective, elaborate, trustworthy, credible
    (at least in an ideal world...)
  - Writing editorial recommendations is generally paid work

A Foodie's Guide to the Best Burgers in Singapore

The best burgers in Singapore

15 Best Burgers In Singapore So Good, You Won't Stop At Just Bun

Burgers, burgers, burgers: The best to sink your teeth into whether dine-in or delivered

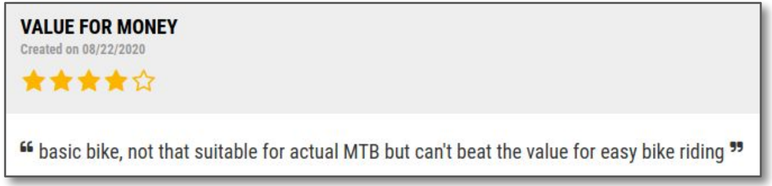
Best Burger Joints You Must Try in Singapore

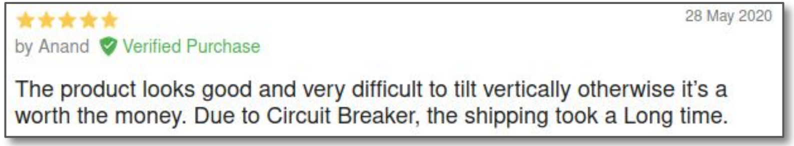
Snackdown review: The best burgers in Singapore

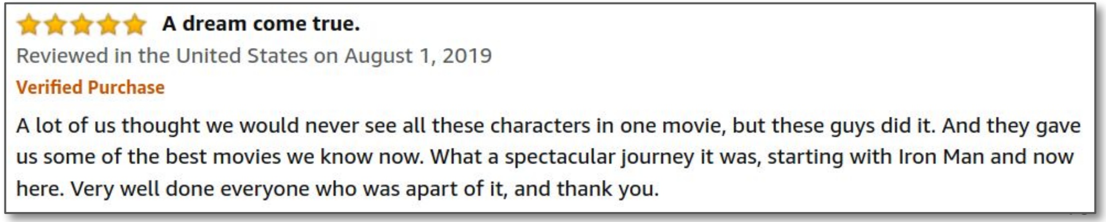
The best restaurants with burger delivery options in Singapore

Peer Recommendations

- Recommendations by normal users
  - Online word-of-mouth recommendations
    (however, users are typically strangers)
  - Common feature on shopping/booking sites
  - Typically subjective, short, biased
  - Many reviews per item create average view but represents again information overload







Manual Recommendations — Pros & Cons

- Pros
    - Semantically rich (ratings, plain text, images, videos, etc.)
    - Explainability / Interpretability
- Cons
    - Manual effort — What is the incentive for writing a review?
    - Lack of personalization

Recommendation Fraud

How merchants use Facebook to flood Amazon with fake reviews

Online reviews 'used as blackmail'

Army of fake reviewers being built to dupe buyers, drive online sales

'Why I write fake online reviews'

Spotify tests sponsorship of full-screen album recommendations

Influencer Marketing Fraud: The Shady Side of Social Media

Can We Trust Social Media Influencers?

A new study analyses the murky world of fake Amazon reviews

Buy Lazada Review - Votes - Sells

How to Get Paid to Write Reviews

Threatening a business with a bad review is ugly bullying

Amazon's Fake Review Problem Is Getting Worse

Amazon is filled with fake reviews and it's getting harder to spot them

Quick Quiz

How many Amazon reviews for electronic products are **"fake"**?
(according to a study from 2018)

A ~10%

B ~30%

C ~60%

D ~90%

Outline

- **Overview**
  - Motivation
  - **~~Naive~~ / alternative approaches**
- Content-based recommendation systems
  - Pairwise item-item similarity
  - User-item similarity
- Collaborative filtering (CF)
  - Memory-based CF
  - Model-based CF

Recommendations Simple Aggregations

- Rank items based on aggregated scores

Rotten Tomatoes Top-100 Movies

BEST OF RT

Movies with 40 or more critic reviews vie for their place in history at Rotten Tomatoes. Eligible movies are ranked based on their Adjusted Scores.

Rank	Rating	Title	No. of Reviews
1.	96%	Black Panther (2018)	516
2.	94%	Avengers: Endgame (2019)	531
3.	93%	Us (2019)	536
4.	97%	Toy Story 4 (2019)	445
5.	99%	Lady Bird (2017)	394
6.	100%	Citizen Kane (1941)	94
7.	97%	Mission: Impossible - Fallout (2018)	430
8.	98%	The Wizard of Oz (1939)	120
9.	96%	The Irishman (2019)	441
10.	96%	BlacKkKlansman (2018)	438

IMDB Top-250 Movies

Top Rated Movies

Top 250 as rated by IMDb Users

SHARE

Rank & Title	IMDb Rating	Your Rating
1. The Shawshank Redemption (1994)	9.2	
2. The Godfather (1972)	9.1	
3. The Godfather: Part II (1974)	9.0	
4. The Dark Knight (2008)	9.0	
5. 12 Angry Men (1957)	8.9	
6. Schindler's List (1993)	8.9	
7. The Lord of the Rings: The Return of the King (2003)	8.9	
8. Pulp Fiction (1994)	8.8	
9. The Good, the Bad and the Ugly (1966)	8.8	
10. The Lord of the Rings: The Fellowship of the Ring (2001)	8.8	

Simple Aggregations — Pros & Cons

- Pros
  - Relatively easy to compute (typically weighted aggregated based on different factors)
  - Typically good/safe recommendations (particularly for new/unknown users)

- Cons
  - Requires sufficient number of ratings per items
  - High risk of popularity bias; lack of diversity
    ("rich get richer" effects, "few get richer" effects)
  - Lack of personalization

Personalized Recommendations

- Users have different preferences that define the relevance of items
  - Preferences = interests, tastes, likings, needs, wants, desires, etc.
  - Relevant items = items that match users' preferences best

- Basic setup
  - Set of users $U = \{u_1, u_2, \dots, u_n\}$
  - Set of items $V = \{v_1, v_2, \dots, v_m\}$
  - Rating matrix R with $|U|$ rows and $|V|$ columns
  - Matrix element R_{uv}: u's rating of v (e.g., 1-5 stars, binary 0/1)

items
~~movies~~

users

	2	5		1			
		4		5	5		
	3.7?	4	2				
2		3			2		3
					4		
5		5				2	
						4	3
3				1			
		2		2			
		5		1			4

Personalized Recommendations — Core Tasks

Collect ratings R_{uv}

Infer missing values $\widehat{R}_{uv}$

- In practice, mainly interested in high values
- Algorithmic component of recommender systems
- Wide range of existing approaches

Evaluation: How good are the recommendations?

- Compare $\widehat{R}_{uv}$ with true R_{uv} from test set

Collecting Ratings

- Explicit
  - Ask/invite/encourage users to rate items
  - 1 Pay users to rate items (e.g., crowdsourcing)

- Implicit — derive ratings from users' behavior, e.g.:
  - Product bought
  - Video watched
  - Article read
  - Link clicked
  - ...

➜ High ratings

(But how to get low ratings?)

Key challenge: Rating matrix R is in practice very sparse!

Evaluation

Training

	2	5		1			
		4		5	5		
		4	2				
2		3			2		3
					4		
5		5				2	
						4	3
3				1			
		2		2			
		5		1			4

Test

- Split R into training and test set
- Performance metrics
  - Root Mean Squared Error (for numerical ratings)

$$\sqrt{\frac{1}{|S|}\sum_{(u,v)\in S}\left(\widehat{R}_{uv} - R_{uv}\right)^2}$$

Set of (u, v) pairs in test set

  - Precision, Recall, F1 score, etc.
    (TP, TN, FP, FN for binary ratings or binary recommendation after converting numerical ratings)
  - Precision@k, Recall@k
    (precision and recall w.r.t. to the top-k highest predicted ratings)
  - Compare rankings induced by $\widehat{R}_{uv}$ and R_{uv} with $(u, v) \in S$
    (also consider the order of the top-k highest ratings)

Kendall
Spearman
ranking corr coeff.

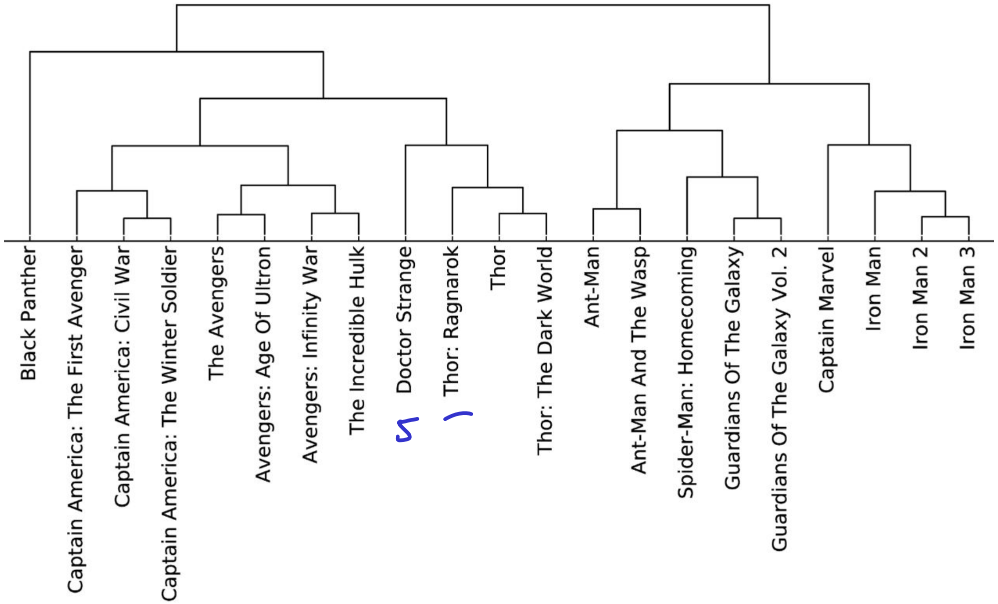
Recommendations Using Association Rules

- User preferences & likings
  - Items: movies, songs, books, etc.
  - Transaction: viewing/listening/reading history
- Interesting rules (movies):
  - Viewer who watched movies {a, b} also watched movies {x, y}
  - Example: {Jaws} ➜ {It}
- Limitations
  - Basic AR algorithm ignores ratings
  - Popularity bias: user with very unique tastes likely to get subpar recommendations

TID	Items
1	Jaws, Halloween, Scream, It
2	Alien, Jaws, Scream, It
3	Tenet, Inception, Interstellar
4	Jaws, Halloween, It
5	Alien, Tenet Jaws, It
...	

Recommendations Using Clustering

Example: Hierarchical clustering of MCU movies



- Approach
  - Cluster movies based on "useful" features (genre, director, writer, length, ...)
  - Recommend movies from clusters with movies a user has rated highly
- Limitations
  - Find good feature in practice very difficult (we come back to that)
  - "Unsystematic:" no well-defined process to pick recommendations

Source: [How the Avengers assemble: Ecological modelling of effective cast sizes for movies]

Recommendations Using Regression (or Classification)

- Example approach: Linear Regression
  - Independent variable: movie features
  - Dependent variable: user rating

➜ Build a linear regression model for each users

- Limitations
  - Requires good features for each item
  - Cold-start problem: requires a lot of user ratings to build a good model

Rated movies of an individual user

original	powerful	absorbing	comical	romantic	...	Rating
0.20	0.95	0.80	0.00	0.10	...	4.5
0.80	0.25	0.50	0.50	0.75	...	4.0
0.05	0.2	0.20	0.95	0.90	...	2.0
0.60	0.20	0.80	0.00	0.85	..	3.5
0.90	0.95	0.90	0.10	0.40	..	5.0
0.45	0.50	0.20	0.60	0.30	..	2.0
0.10	0.40	0.55	0.90	0.30	..	2.5
0.75	0.50	0.50	0.40	0.40	..	3.5
0.80	0.80	0.85	0.10	0.10	...	4.5
...	...	...	...	...	...	...

Outline

- Overview
  - Motivation
  - Naive / alternative approaches
- **Content-based recommendation systems**
  - **Pairwise item-item similarity**
  - User-item similarity
- Collaborative filtering (CF)
  - Memory-based CF
  - Model-based CF

Content-Based Recommender System

- Intuition
  - Recommend item *v* to user *u* that are similar to *v* and *u* has rated highly
  - Examples: movies of the same genre, songs from the same artist, articles about the same topic, products with similar features, etc.

- Basic requirement: **item profiles** = feature vector for each item, e.g.:
  - Movie: genre, director, writer, cast, length, year, …
  - Product: type, brand, price, weight, color, ...
  - Article: set of (important) words / tf-idf vector / ...

Running Example

- MovieLens dataset
  - Items: Movies of different genres
  - Features: 20 genres (incl. "uncategorized")
  - Ratings: 1-5 stars (incl. half stars)

	comedy	action	romance	drama	fantasy	thriller	...
Clueless	1	0	1	0	0	0	...
Heat	0	1	0	0	0	1	...
Bad Boys	1	1	0	1	0	1	...
Leon	0	1	0	1	0	1	...
Alice	1	0	1	1	1	0	...
Jarhead	0	1	0	1	0	0	...
Rocky	0	0	0	1	0	0	...
Big	1	0	1	1	1	0	...
Krull	0	1	0	0	1	0	...
...	...	...	...	...	...	...	...

- Numbers for "Small" dataset
  - 610 users, 9,742 movies
  - ~100k ratings ➜ sparsity: ~1.7%

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	...
u_1	3.0					4.5			...
u_2	3.5	5.0						2.0	...
u_3				3.0		5.0			...
u_4		4.5	2.0				2.0		...
u_5	3.5		2.5						...
u_6					3.0			3.0	...
u_7					3.5	3.0			...
u_8			2.0				3.0	3.0	...
u_9	5.0				4.5				...
...	...	...	...	...	...	...			...

Source: MovieLens Dataset

Simple Approach — Pairwise Item Similarity

- Pairwise item similarity *sim*(*x*,*y*)
  - *x, y* — feature vectors of movies
  - Common metric: cosine similarity

$$sim(x, y) = \cos(\theta) = \frac{x \cdot y}{\|x\| \|y\|}$$

sim(*Heat, Heat*) = 1.0

sim(*Heat, Clueless*) = 0.0

sim(*Heat, Bad Boys*) = 0.77

sim(*Heat, Jarhead*) = 0.33

sim(*Heat, Alice*) = 0.0

- Limitation: Requires reference item, e.g.:
  - Movie(s) the user was most recently watching
  - Movie(s) the user has rated the highest
  - Movie(s) the user is currently browsing

Outline

- Overview
  - Motivation
  - Naive / alternative approaches
- **Content-based recommendation systems**
  - Pairwise item-item similarity
  - **User-item similarity**
- Collaborative filtering (CF)
  - Memory-based CF
  - Model-based CF

User-Item Similarities

- Needed: **user profiles** = feature vector for each user
  - Requirement: same shape as item profiles to calculate similarities
  - Approach: user profile = "some aggregate" over item profiles rated by the user

Profiles of movies watched/rated/purchased/… by user *u*

User u's profile

	m_1	m_2	m_3	...		?	User u's profile
comedy	1	1	1	1			?
action		1			1		?
romance			1	1	1		?
drama			1				?
fantasy	1						?
thriller	1		1				?
horror							?
adventure	1	1			1		?
animation			1	1	1		?

User-Item Similarities — Binary Utility Matrix

- $R_{uv} \in \{0, 1\}$ — for example, $R_{uv} = 1$ if
  - User u bought movie v
  - User u watched movie v

  Implicit rating that *u* liked *v* (no explicit ratings available here; but also no implicit dislikes!)

- Simple Average

	m_1		m_2								User u's profile
comedy	1		1		1		1				4/5 = **0.8**
action			1						1		2/5 = **0.4**
romance					1		1		1		3/5 = **0.6**
drama					1						1/5 = **0.2**
fantasy	1	+		+		+		+		⇨	1/5 = **0.2**
thriller	1				1						2/5 = **0.4**
horror											0/5 = **0.0**
adventure	1		1						1		3/5 = **0.6**
animation					1		1		1		3/5 = **0.6**

#movies = 5

User-Item Similarities — Real-Valued Utility Matrix

- $R_{uv} \in \mathbb{R}$ — for example, $R_{uv} \in \{1.0, 1.5, 2.0, 2.5, \ldots, 5.0\}$ star rating
  - Explicit rating of user *u* for movie *v*
  - Important: semantic interpretation — ratings express both likes and dislikes (despite all ratings positive)
  - Use rating as weights for features for a weighted aggregation

comedy	1	1	1	1	
action		1			1
romance			1	1	1
drama			1		
fantasy	1				
thriller	1		1		
horror					
adventure	1	1			1
animation			1	1	1
u's ratings	**1.5**	**2.0**	**4.5**	**5.0**	**4.0**

Intuition

- The user likes romantic and animated movies
- The user dislikes fantasy and adventure movies

User-Item Similarities — Real-Valued Utility Matrix

- Step 1: Normalize ratings
  - Subtract average user rating from each movie rating
  - Converts ratings into positive (liked) and negative (disliked) scale
  - Distinguishes "generous" users (mostly rate highly and a 3.0 is a low rating) from more "grumpy" users (mostly rate low and a 3.0 is a high rating)

$1.5 - 3.4 = -1.9$

comedy	1	1	1	1	
action		1			1
romance			1	1	1
drama			1		
fantasy	1				
thriller	1		1		
horror					
adventure	1	1			1
animation			1	1	1
u's normalized ratings 1.5 →	**-1.9**	**-1.4**	**1.1**	**1.6**	**0.6**

Average user rating

$$\frac{1.5 + 2.0 + 4.5 + 5.0 + 4.0}{5} = \mathbf{3.4}$$

User-Item Similarities — Real-Valued Utility Matrix

- Step 2: Calculate weighted features for user profile
  - The weights are the normalized weights

...	...	...	...	...	...
adventure	1	1			1
animation			1	1	1
u's normalized ratings	-1.9	-1.4	1.1	1.6	0.6

$$w_{adventure} = \frac{1 \cdot (-1.9) + 1 \cdot (-1.4) + 1 \cdot 0.6}{3} = -0.9$$

$$w_{animation} = \frac{1 \cdot 1.1 + 1 \cdot 1.6 + 1 \cdot 0.6}{3} = 1.1$$

4/5

User u's profile

-0.15	comedy
-0.40	action
1.10	romance
1.10	drama
-1.90	fantasy
-0.40	thriller
0.00	horror
-0.90	adventure
1.10	animation

Quick Quiz

Given the example calculation above what should we **ensure** to get good profiles?

$$w_{adventure} = \frac{\overbrace{1 \cdot (-1.9) + 1 \cdot (-1.4) + 1 \cdot 0.6}^{\text{as many as possible}}}{3} = -0.9$$

A A user has given mostly high ratings to different movies

B Enough movies of the same genre have been rated

C A user has rated enough movies of the same genre

D A user's ratings for the same genre need to be diverse

User-Item Similarities

- Pairwise item similarity *sim*(*u*,*v*)
  - *u* — user profile; v — item profile
  - Suitable metric: cosine similarity (note that user and item profiles can have different magnitudes)

➜ Recommend items v_i to user *u* with max. similarities $sim(u,v_i)$

- Practical considerations
  - Top *k* most similar items always the same ➜ add some randomization for diversity
    (the set of top *k* most similar items might only change over time if the user rates more items)
  - Top *k* most similar items might include items the user has already rated ➜ remove those items
    (in practice, recommending known items not uncommon — e.g., YouTube recommendations)
  - More sophisticated ways to aggregate item profiles to user profiles conceivable
    (for example: ignore underrepresented features, e.g., if a user rated only one comedy movie)

Content-Based Recommender System — Pros & Cons

- Pros
  - Recommendations for user *u* do not depend on other users
    (this also allows for good recommendations for users with very unique tastes)
  - Recommendations can also include new or unpopular items (i.e., with no or very view ratings)
  - Good explainability (features that had most effect on the high similarity)

- Cons
  - Cold-start problem: How to build a profile for new users?
    (naive approach: recommend generally popular items to new users)
  - Finding good features (and values!) for items a non-trivial tasks
    (Question: Are genres a good feature set to represent movies?)
  - Overspecialization: By default, no recommendations outside a user's profile
    (in practice: add some randomization into the recommendation process)

Outline

- Overview
  - Motivation
  - Naive / alternative approaches
- Content-based recommendation systems
  - Pairwise item-item similarity
  - User-item similarity
- **Collaborative filtering (CF)**
  - **Memory-based CF**
  - Model-based CF

Collaborative Filtering

- Idea: Utilize the opinions of others
  - Recommend items that other user with similar tastes/preferences/needs have liked
  - Does not require item or user-specific features
- Two perspectives
  - User-based — two users are similar if they rated the same items similarly
  - Item-based — two items are similar if they are equally rated by users

User-Based CF — Calculating Similarities

- Example: movie ratings
  - How much might Bob like the movie "Heat"?

	Clueless	Heat	Jarhead	Big	Rocky
Alice	2	4	5	0	1
Bob	1	???	4	0	2
Claire	1	0	4	3	0
Dave	5	1	2	0	5
Erin	1	5	3	0	3

- Intuitions given the dataset
  - Alice and Bob have similar tastes, so Bob might rate *"Heat"* similar to Alice
  - Claire and Bob have similar tastes, but Claire has not rated *"Heat"*
  - Dave and Bob have very different tastes, so Dave's opinion about *"Heat"* shouldn't matter
    (if anything, it should an indicator that Bob will like "Heat"; usually not relevant in practice)

➜ How can we capture and quantify these intuitions?

User-Based CF — Calculating Similarities

- Represent all users by their rating vectors v
  - Rows of rating matrix

$$r_A = (2,4,5,0,1)^T$$
$$r_B = (1,0,4,0,2)^T$$
$$r_C = (1,0,4,3,0)^T$$
$$r_D = (5,1,2,0,5)^T$$
$$r_E = (1,5,3,0,3)^T$$

Using cosine similarity

$$sim(r_A, r_B) = \boxed{0.77}$$
$$sim(r_D, r_B) = \boxed{0.67}$$

Too similar to capture our intuition

	Clueless	Heat	Jarhead	Big	Rocky
Alice	2	4	5	0	1
Bob	1	???	4	0	2
Claire	1	0	4	3	0
Dave	5	1	2	0	5
Erin	1	5	3	0	3

Problem

- Missing values (0) are treated as negative
- All ratings are positive values

➜ No explicit notion of dissimilarity
(only less or more similar)

User-Based CF — Calculating Similarities

$$Alic = \frac{2+4+5+1}{4} = \frac{12}{4} = 3$$

- Idea: Normalize rating vectors
  - Mean-centering — subtract row mean from each rating vector
  - Missing values (0) now represent the average rating
  - Bad ratings (i.e., below average) now represented by negative values

	Clueless	Heat	Jarhead	Big	Rocky
Alice	2-3 = **-1**	4-3 = **1**	5-3 = **2**	**0**	1-3 = **-2**
Bob	1-2.33 = **-1.33**	???	4-2.33 = **1.67**	**0**	2-2.33 = **-0.33**
Claire	1-2.67 = **-1.67**	**0**	4-2.67 = **1.33**	3-2.67 = **0.33**	**0**
Dave	5-3.25 = **1.75**	1-3.25 = **-2.25**	2-3.25 = **-1.25**	**0**	5-3.25 = **1.75**
Erin	1-3 = **-2**	5-3 = **2**	3-3 = **0**	**0**	3-3 = **0**

$$sim(r_A, r_B) = 0.78$$

$$sim(r_D, r_B) = -0.65$$

Cosine similarity between mean-centered vectors ➜ **Pearson Correlation Coefficient**

User-Based CF — Predicting Ratings

- $\widehat{R}_{uv}$ = weighted average of ratings from similar users
  - N — set of *k* users most similar to *u* who have already rated item *v*

$$\widehat{R}_{uv} = \frac{\sum_{w \in N} sim(u,w) \cdot R_{wv}}{\sum_{w \in N} sim(u,w)}$$

- For the example
  - With k = 2

$$\widehat{R}_{Bob,Heat} = \frac{\overbrace{0.78 \cdot 4}^{\text{Alice}} + \overbrace{0.44 \cdot 5}^{\text{Erin}}}{0.78 + 0.44} = 4.3$$

	Clueless	Heat	Jarhead	Big	Rocky
Alice	2	4	5	0	1
Bob	1	???	4	0	2
Claire	1	0	4	3	0
Dave	5	1	2	0	5
Erin	1	5	3	0	3

Item-Based CF

- Analog to user-based approach
  - For an item *v*, find the most similar items
    (2 items are similar, if their ratings across all users are similar)
  - $\widehat{R}_{uv}$ = weighted average of ratings of similar items
  - M — set of *k* items most similar to *v* who have already been rated by u

$$\widehat{R}_{uv} = \frac{\sum_{i \in M} sim(i, v) \cdot R_{ui}}{\sum_{i \in M} sim(i, v)}$$

Note: Recall that in content-based recommender systems, measuring the similarity between items relied on item profiles / feature vectors. In case of item-item CF, the rating vector of an item represents its profile.

Item-Based CF — Example

Calculate mean-centered rating vectors
(here: columns of rating matrix)

$$v_C = (0, -1, -1, 3, -1)^T$$

$$v_H = (0.67, 0, 0, -2.33, 1.67)^T$$

$$v_J = (1.4, 0.4, 0.4, -1.6, -0.6)^T$$

$$v_B = (0, 0, 0, 0, 0)^T$$ ← Big

$$v_R = (-1.75, -0.75, 0, 2.25, 0.25)^T$$

	Clueless	Heat	Jarhead	Big	Rocky
Alice	2	4	5	0	1
Bob	1	???	4	0	2
Claire	1	0	4	3	0
Dave	5	1	2	0	5
Erin	1	5	3	0	3

Calculate distances between "Heat" and all other movies Bob has already rated

Heat

$$sim(v_H, v_C) = -0.85$$

$$sim(v_H, v_J) = 0.55$$

$$sim(v_H, v_R) = -0.69$$

Even for k=2, there is only 1 movie with a positive Pearson correlation coefficient

$$\widehat{R}_{uv} = \frac{\overbrace{0.55 \cdot 4}^{\text{Jarhead}}}{0.55} = 4$$

Collaborative Filtering — User-Based vs. Item-Based

- In theory: user-based and item-based are dual approaches

- In practice: item-based typically outperform user-based
  - Items are "simpler" than users
  - Items can be more easily described
  - User can have very varied tastes

➜ Item-item similarity typically more meaningful

Outline

- Overview
  - Motivation
  - Naive / alternative approaches
- Content-based recommendation systems
  - Pairwise item-item similarity
  - User-item similarity
- **Collaborative filtering (CF)**
  - Memory-based CF
  - **Model-based CF**

Model-Based Collaborative Filtering

- Latent factor models
  - Latent representation: *k*-dimensional vector for each user *u* and item *v*
  - Learn latent representations from the data
    (in contrast to content-based systems where feature vectors are constructed)
  - Estimate unknown ratings $\widehat{R}_{uv} = w_u^T h_v$

Non-negative

- Approach: Matrix Factorization
  - Put all user vectors into a matrix *W*
  - Put all item vector into a matrix *H*

➜ Find *W*, *H* such that *R* = *WH*

or at least approximately

Matrix Factorization — Basic Setup

- Given: ratings matrix R
  - m — number of users |W|
  - n — number of items |H|

$$\underset{m \times n}{R} = \underset{m \times k}{W} \times \underset{k \times n}{H}$$

- Hyperparameter k
  - Size of latent representations

$k=100$

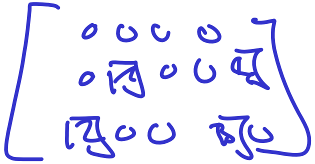
Finding Matrices W, H

ground truth

estimate

- Minimize loss function $$L = \sum_{R_{uv}>0} e_{uv} = \sum_{R_{uv}>0} (R_{uv} - \widehat{R}_{uv})^2 = \sum_{R_{uv}>0} (R_{uv} - w_u^T h_v)^2$$

  ➜ with regularization: $$L = \sum_{R_{uv}} (R_{uv} - w_u^T h_v)^2 + \lambda(\|w_u\|^2 + \|h_v\|^2)$$



- Using Gradient Descent

Calculate gradients

$$\frac{\partial e_{uv}}{\partial w_u} = -2(R_{uv} - w_u^T h_v)h_v + 2\lambda w_u$$

$$\frac{\partial e_{uv}}{\partial h_v} = -2(R_{uv} - w_u^T h_v)w_u + 2\lambda h_v$$

Update rules

$$w_u \leftarrow w_u - \eta \frac{\partial e_{uv}}{\partial w_u}$$

$$h_v \leftarrow h_v - \eta \frac{\partial e_{uv}}{\partial h_v}$$

Finding Matrices W, H — Algorithm

Input : rating matrix $R^{(m \times n)}$, latent vector size k, #iterations T

Initialization : $W^{(m \times k)}, H^{(k \times n)}$ with values 0..1

for 1 **to** T

for all u, v **with** $R_{uv} > 0$

$$w_u \leftarrow w_u + \eta[2(R_{uv} - w_u^T h_v)h_v - 2\lambda w_u]$$

$$h_v \leftarrow h_v + \eta[2(R_{uv} - w_u^T h_v)w_u - 2\lambda h_h]$$

return W, H

Finding Matrices W, H — Example

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
u_1	**4**	0	0	**5**	**1**	0	0
u_2	**5**	**5**	**4**	0	0	0	0
u_3	0	0	0	**2**	**4**	**5**	0
u_4	0	**3**	0	0	0	0	**3**

Calculate W, H

$k = 100$
$\lambda = 0.1$
#iterations = 10k

$W \cdot H$

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
u_1	**3.9**	3.5	3.4	**4.8**	**1**	2.4	3.1
u_2	**4.9**	**4.8**	**4**	3.7	3.2	5.3	4.4
u_3	3.4	3.3	3.6	**2**	**3.8**	**4.9**	3.9
u_4	3.1	**2.9**	3.2	2.5	2.3	3.8	**3**

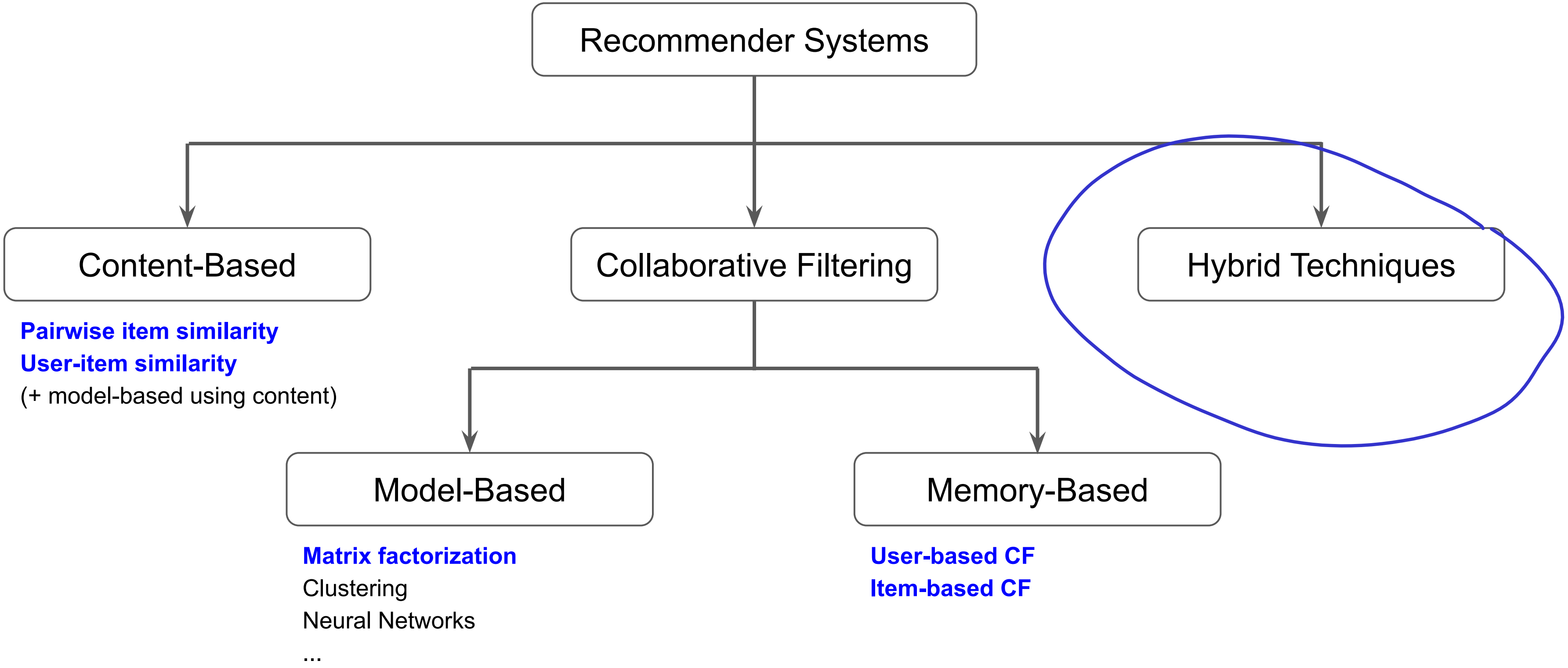
- Effects of regularization
  - Increase λ: worse fit of known ratings, "smoother" values for all ratings
  - Decrease λ: better fit of known ratings, more "extreme" values of unknown ratings

Collaborative Filtering — Pros & Cons

- Pros
  - No need to find and create good features
    (such as genres for movies)
  - Intuitive approach

- Cons
  - Similarity calculations rely on sufficient number of ratings
  - Cold-start problem in case of new users or items
  - Popularity bias: user with very unique tastes likely to get subpar recommendations
    (because even the k most similar users will not be truly very similar)
  - Naive implementation very expensive: Finding k most similar users/items $\in O(|R|)$
    (optimization techniques needed: e.g. clustering of users/items to limit search space)

Recommender Systems — Summary



Quick Quiz

What does **not** affect the recommendations made by Collaborative Filtering?

A A new movie is added

B A new user is added

C A user rates a movie

D A movie's details get updated

Quick Quiz

What does generally affect the recommendations made by Collaborative Filtering the **least**?

arguably

A An old user has rated an old movie

B An old user has rated a new movie

C A new user has rated an old movie

D A new user has rated a new movie

The Dangers of (Over-)Personalization

- 2 infamous side-effects
  (particularly when recommending news or social media posts)
  - **Filter bubbles**
  - **Echo chambers**

- Core problems
  - No incentive for service providers to ensure (sufficient) diversity
  - Users do not know what content is shown and why (or why not!)

Why is TikTok creating filter bubbles based on your race?

How social media filter bubbles and algorithms influence the election

When Algorithms Decide Whose Voices Will Be Heard

Social Media Giants Support Racial Justice. Their Products Undermine It.

Facebook reportedly ignored its own research showing algorithms divided users

Outline

- Overview
  - Motivation
  - Naive / alternative approaches
- Content-based recommendation systems
  - Pairwise item-item similarity
  - User-item similarity
- Collaborative filtering (CF)
  - Memory-based CF
  - Model-based CF

Summary

- Recommender systems
  - More specifically: personalized recommender systems
  - Integral component of many online platforms
  - User: find relevant items + providers: present relevant items
  - BUT: risk of over-personalized recommendations

- Implementing recommender systems
  - Wide range of data mining techniques applicable
  - No "one-size-fits-all" solutions
  - In practice, hybrid approaches most successful

Solutions to Quick Quizzes

- Slide 13: C
- Slide 34: C
- Slide 54: D
- Slide 55: A