Recommender Systems
推薦系統

賴旭昭 Hsu-Chao Lai

National Cheng Kung University

國立成功大學
National Cheng Kung University

About Me

- Hsu-Chao Lai 賴旭昭
- AI Scientist at NetDB Lab
- Ph.D. of NYCU CS
- Specialized in
  - Recommender systems
  - Real-time bidding
  - AI stock trading
- hclai@netdb.csie.ncku.edu.tw

Outline

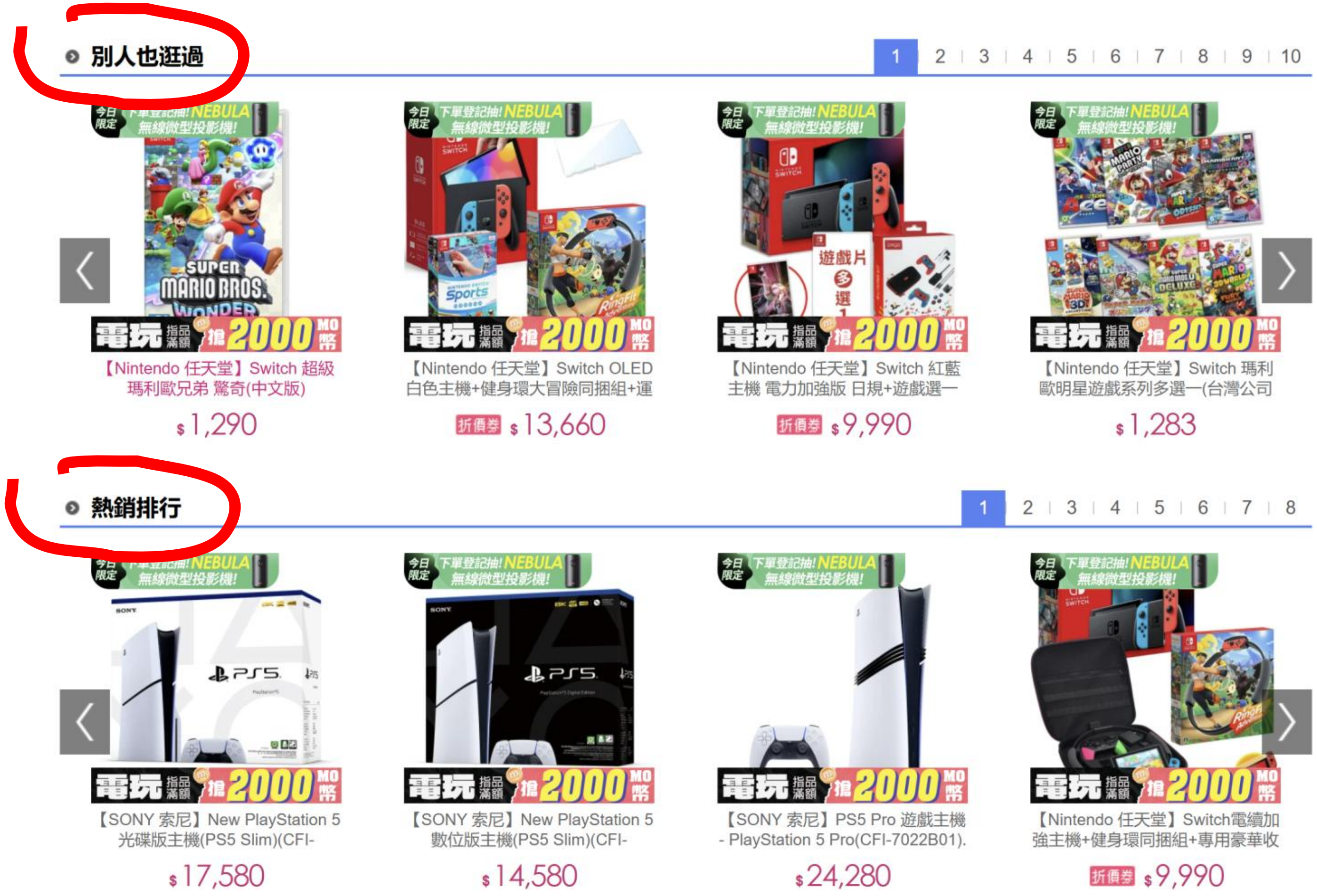
- Introduction
- Content-based Approaches
- Collaborative Filtering
- Matrix Factorization
- Ranking Objective
- Takeaways
- Appendix: Glossary of Hot Topics
- References

Background Knowledge:

- Linear regression
- Logistic regression
- Maximum log-likelihood
- Or Minimum negative log-likelihood
- Singular Value Decomposition (SVD)
- (Stochastic) gradient descent

Example

momo



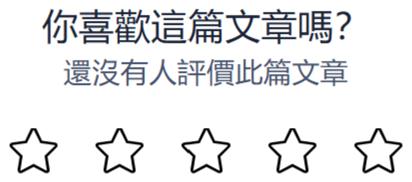

Retrieval
Recommendations
User
Item Database

Types of Recommendations

- **Non-personalized recommendations**
  - Simple aggregation
    - Top-10 popular, new-on-shelf, most advertised
- **Personalized recommendations**
  - Specially tailored to **each individual**
  - Find relevant items that are now browsing
    - Amazon, Netflix, YouTube, Facebook, news, etc
  - Customize user experience
    - Spotify, trip planning, friend, etc
  - Increase conversion rate (e.g., purchase)
    - Ads, new items, etc

Recommender Systems (RS)

- $\boldsymbol{U}$: a set of users
- $\boldsymbol{V}$: a set of items



- Optimize a **Utility function** $\boldsymbol{f}: \boldsymbol{U} \times \boldsymbol{V} \rightarrow \boldsymbol{R}$
  - $\boldsymbol{R}$: a set of user feedback
    - Ratings (1-5 stars)
    - Binary (purchased or not)
    - Polarity (like or dislike)
  - Preference, satisfaction, etc

Workflow

1. **Collect known user feedback**
   - What data can be used as user feedback?
2. **Infer unknown user feedback based on the known ones**
   - Focus on retrieving unknown feedback with high ratings
     - Low rating ones do not bring profits
3. **Evaluate the inference of unknown ones**
   - How to properly measure the performance of recommendations

(1) Known User Feedback Collection

- **Explicit Feedback**
  - Ask users to rate items
    - 1-5 stars
    - Very dissatisfied, dissatisfied, moderate, satisfied, very satisfied
  - Crowdsourcing: pay users to rate

- **Implicit Feedback**
  - User actions/engagement
    - *Purchase* implies high ratings
    - *Watched* implies high ratings
    - Watch time, visit and re-visit, etc
  - What defines low ratings?

Utility Matrix with Explicit Feedback

Movie Rating Records		
Willie	全面啟動	5
Xavier	星際效應	2
Zack	全面啟動	4
Willie	星際效應	5
Xavier	鬼滅之刃	5
Willie	奧本海默	5
Zack	奧本海默	4

Ratings	奧本海默	星際效應	全面啟動	鬼滅之刃	天氣之子
Willie	5	5	5		
Xavier		2		5	
Yvonne					
Zack	4		4		

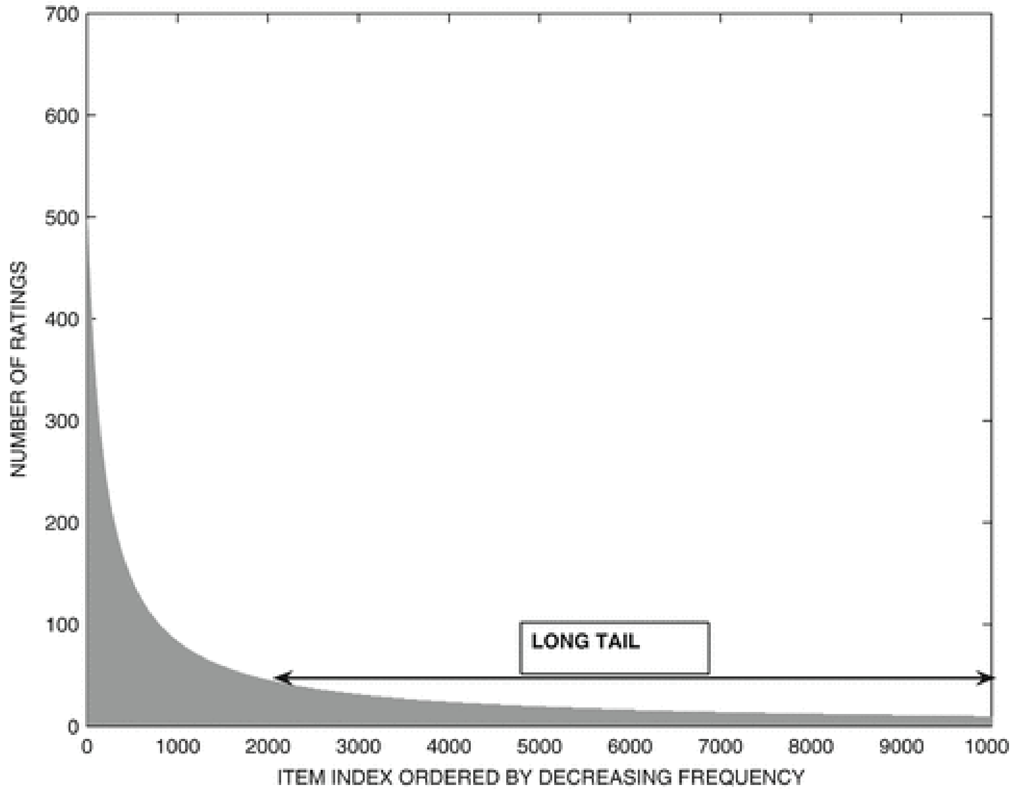
Utility Matrix with Implicit Feedback

Movie Watching Records		
Willie	全面啟動	Watched (5)
Xavier	星際效應	Watched (2)
Zack	全面啟動	Watched (4)
Yvonne	天氣之子	Watched (-)
Willie	星際效應	Watched (5)
Xavier	鬼滅之刃	Watched (5)
Yvonne	鬼滅之刃	Watched (-)
Willie	奧本海默	Watched (5)
Zack	奧本海默	Watched (4)
Zack	星際效應	Watched (-)

Watched	奧本海默	星際效應	全面啟動	鬼滅之刃	天氣之子
Willie	1	1	1		
Xavier		1		1	
Yvonne				1	1
Zack	1	1	1		

(2) Inferring Utilities

- Key challenges:
  - Utility matrix $\boldsymbol{M}$ is **sparse**
    - Most of the entries of $\boldsymbol{M}$ are left blank
    - Sparsity[1]: MoiveLens 95-99%, Netflix Prize 99%, Amazon Product Review 99.9%
  - **Cold-start** issues
    - New items or users have few records to learn
- Three classic approaches:
  - **Content-based**
  - **Collaborative Filtering**
  - **Latent-based**



[1] MovieLens | GroupLens, Netflix Prize data, and Amazon Review Datasets

(3) Evaluation

- Pointwise objectives
  - Classification: logistic function, cross-entropy, softmax, …
  - Regression: MSE, MAE, Mean Squared Logarithmic Error, …
- **Highly sensitive** to the **sparsity** issue
- Given the ratio of positive to negative data is about 1:99 in $\boldsymbol{M}$
- Always inferring “negative” yields 99% accuracy during training
- Unable to retrieve items with high ratings

- Solution: **pairwise objectives (a.k.a. ranking)**

Outline

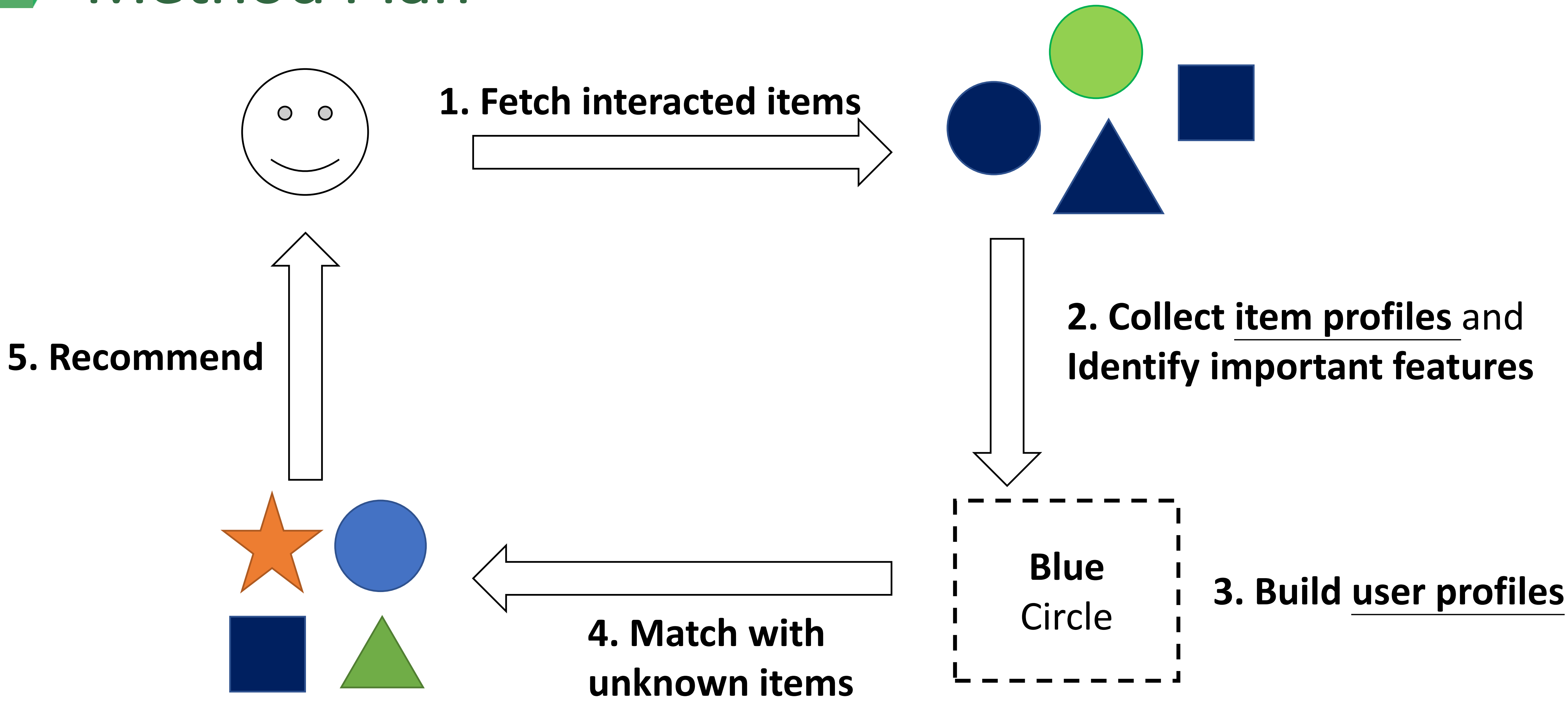
Content-based Approaches

- Core idea: **items have profiles!**
  - Movie: {genre, director, casts, series}
  - Article: {author, platform, a set of keywords}
  - E-commerce product: {function, brand, specs}
- Recommend unknown items to customer u similar to previous items highly rated by u

Watched	奧本海默	星際效應	全面啟動
Willie	1	1	1

- Recommend C. Nolan’s new movie to Willie
- Recommend new Apple products to loyal customers

Method Plan


1. Fetch interacted items
2. Collect item profiles and
Identify important features
Blue
Circle
3. Build user profiles
4. Match with
unknown items
5. Recommend

(2) Feature Extraction for Item Profiles

- A set (vector) of features *distinguishing each item*
- Structural data
  - Movie recommendation: {genre, director, casts, series}
- Unstructured data
  - **Text:** **key words**, latent topics, writing style, …
    - News, product descriptions, reviews, blogs, social media, etc
  - **Video:** latent topics, sequential storytelling, …
  - **Image:** objects, styles, color tones, …
  - *Too abstract to extract*

A Glimpse of Key Word Extraction: TF-IDF

1. Frequency $f_{x,y}$ of word x in document y
   - $\boldsymbol{TF_{x,y} = {}^{f_{x,y}}/_{MAX_{\forall i} f_{i,y}}}$
2. The **uniqueness** of x among all documents
   - $\boldsymbol{IDF_{x,y} = ln({}^{N}/_{n_x})},$
     - where $\boldsymbol{N}$ is the number of total documents, and $\boldsymbol{n_x}$ is the number of docs containing x
3. TF-IDF score: $\boldsymbol{w_{x,y} = TF_{x,y} \times IDF_{x,y}}$
4. **Document profile**: a set of words with top-*k* TF-IDF scores

Modern approach key words: text embeddings, LLM, attention, transformer, BERT, …

(3)(4) User Profiles and Recommendations

- A set (vector) of features *profiling/describing user preference*
  - Weighted average of rated item profiles

- Matching user profile $\boldsymbol{u}$ to unknown item profile $\boldsymbol{v}$
- **Cosine similarity**: $\mathrm{f(u, v)} = \cos(u, v) = \frac{u \cdot v}{\|u\| \cdot \|v\|}$

- Recommend the unknown items with *top-k cosine similarity scores*

Pros and Cons for Content-based Methods

Pros	Cons
No sparsity issue	Extensive feature engineering
No cold-start item problem	Cold-start user problem
Identify unique items or tastes of users	Poor at exploring new items
Interpretable	

Outline

- Introduction
- Content-based Approaches
- Collaborative Filtering
- Matrix Factorization
- Ranking Objective
- Takeaways
- Appendix: Glossary of Hot Topics
- References

What If Content Is Unavailable

- And sometimes engineering features is just too expensive

Watched	奧本海默	星際效應	全面啟動	鬼滅之刃	天氣之子
Willie	5	5	5		
Xavier				5	
Yvonne					
Zack	4		4		

What If Content Is Unavailable

Given records in blue

Watched	奧本海默	星際效應	全面啟動	鬼滅之刃	天氣之子
Willie	5	5	5		
Xavier		?		5	
Yvonne		?			
Zack	4	?	4		

Who is likely to watch this movie? Why?

Collaborative Filtering (CF)

- **[Assumption] Similar users** (based on known) tend to have **similar behavior/ratings** (on unknown)

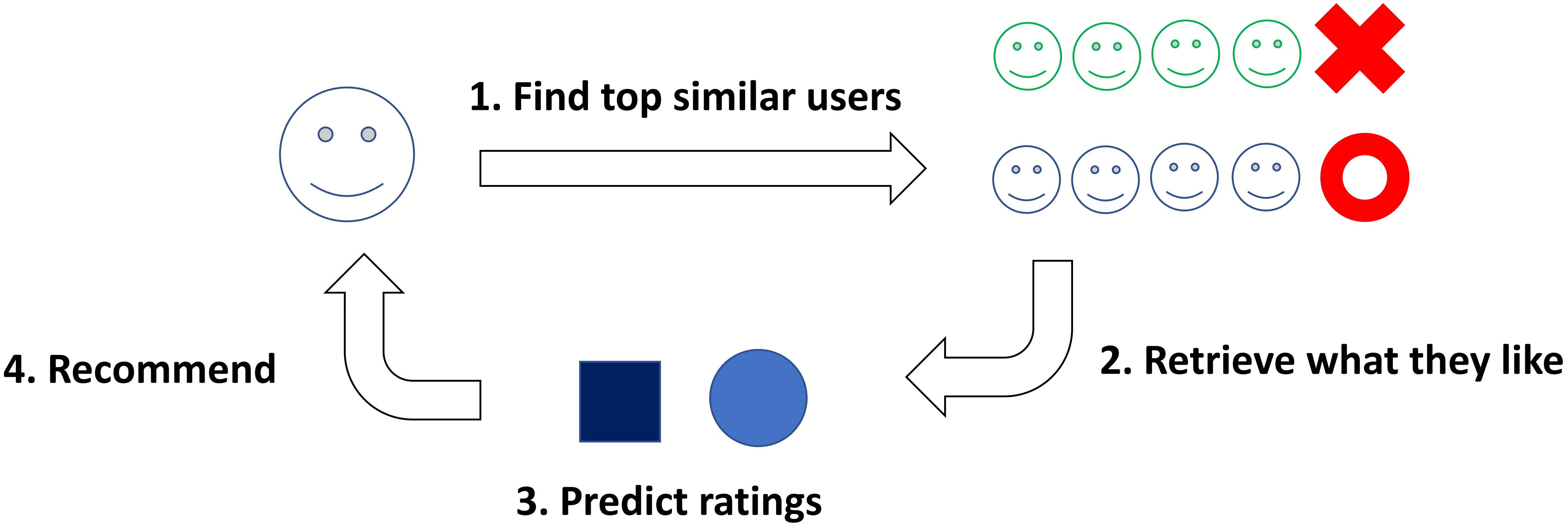
$\Rightarrow$ **User-user Collaborative Filtering**

$\Rightarrow$ **Item-item Collaborative Filtering**

- No need of content
- Treat **behavior** as features

- **Similarity measurement** is the key question of CF!

Method Plan


1. Find top similar users
2. Retrieve what they like
3. Predict ratings
4. Recommend

(1) Choices of Similarity Function

- Cosine Similarity (numerical, binary)
- Euclidean distance (numerical)
- Jaccard similarity (binary)
- Pearson correlation (numerical)

- Choose function based on data types
- Be careful of the limitations and cons

Jaccard Similarity

u_i: user
I_i: interacted items of u_i

- For users u_i and u_j, $\boldsymbol{Jacc(i,j)} = \frac{|\boldsymbol{I_i} \cap \boldsymbol{I_j}|}{|\boldsymbol{I_i} \cup \boldsymbol{I_j}|} \in [0,1]$
  - 0 means no overlap, while 1 means totally the same
- $I_i = \{a, b, c, d, e\}$ and $I_j = \{a, b, c, x, y\}$
- $Jacc(i,j) = 3/7$

- Cons 1: when <u>item space is large</u>, jaccard is easily diluted to 0
  - Every pair of user unions exactly 100,000,000 items
- Cons 2: unable to handle numerical values and negative records
- And there are more ...

Pearson Correlation

$\boldsymbol{i}, \boldsymbol{j}$: user identifier
$\boldsymbol{I_i}$: interacted items of user i
$\boldsymbol{R_{i,v}}$: ratings for item v of user i
$\overline{\boldsymbol{R_i}}$: average ratings of user i

- $\boldsymbol{P_{i,j}} = \frac{\sum_{v \in I_i \cap I_j} (R_{i,v} - \overline{R_i})(R_{j,v} - \overline{R_j})}{\sqrt{\sum_{v \in I_i \cap I_j} (R_{i,v} - \overline{R_i})^2} \sqrt{\sum_{v \in I_i \cap I_j} (R_{j,v} - \overline{R_j})^2}}$

- Recall cosine similarity $\cos(u, v) = \frac{u \cdot v}{\|u\| \cdot \|v\|}$
- **Normalized variation** of cosine similarity

- Cons: inaccurate when intersection set is small
  - Every pair of user shares exactly one interacted item

(2)(3) Rating Predictions

- Let $\boldsymbol{S_i^U}$ be the set of top-*k* similar users of user $\boldsymbol{i}$
- Let $r_{i,v}$ be the ratings of item $\boldsymbol{v}$ for user $\boldsymbol{i}$

- **Average score**: $r_{i,v} = \frac{1}{k}\sum_{j\in \boldsymbol{S_i^U}} r_{j,v}$
- **Similarity-weighted score**: $r_{i,v} = \frac{\sum_{j\in \boldsymbol{S_i^U}} sim(i,j)\times r_{j,v}}{\sum_{j\in \boldsymbol{S_i^U}} sim(i,j)}$
- **kNN-classifier:** output the majority class/rating of $\boldsymbol{S_i^U}$

Item-item CF

GUESS: In practice, which is better?
User-user CF or item-item CF?

- Similar items may attract similar user interactions
- Apply same tricks as user-user CF

- $r_{u,i} = b_{u,i} + \frac{\sum_{j \in \boldsymbol{S}_i^I} sim(i,j) \times (r_{u,j} - b_{u,j})}{\sum_{j \in \boldsymbol{S}_i^I} sim(i,j)}$

- $\boldsymbol{S}_i^{\mathrm{I}}$ is the set of top-*N* similar items
- $sim(i,j)$ here is the item-item similarities

- $b_{u,i} = \mu + b_u + b_i$ is the baseline estimation of $r_{u,i}$
- μ is the overall mean item ratings
- b_u is the rating deviation (bias) of user u => (avg. ratings of u)- μ
- b_i is the rating deviation (bias) of item i

Pros and Cons for Collaborative Filtering

Pros	Cons
No need of feature engineering	Cold-start item/user problem
	Sparsity issue
	Tends to overlook unpopular items

Is it possible to combine content-based and CF to enjoy both benefits? How?

Outline

- Introduction
- Content-based Approaches
- Collaborative Filtering
- Matrix Factorization
- Ranking Objective
- Takeaways
- Appendix: Glossary of Hot Topics
- References

Review CF

- $r_{u,i} = b_{u,i} + \frac{\sum_{j \in S_i^I} sim(i,j) \times (r_{u,j} - b_{u,j})}{\sum_{j \in S_i^I} sim(i,j)}$

- S_i^I is the set of top-*N* similar items
- $sim(i,j)$ here is the item-item similarities
- $b_{u,i} = \mu + b_u + b_i$ is the baseline estimation of $r_{u,i}$
- μ is the overall mean item ratings
- b_u is the rating deviation of user u
- b_i is the rating deviation of item i

- Is similarity-weighted average flexible?
  - Strong and **artificial** assumption restricts learning
  - May overlook **interdependencies** between users
- How to design **a similar but data-driven approach**?
  - Hint: it's a weighted average formulation

Weighted Sum Approach

- Use **weighted sum** rather than weighted average!

$$\widehat{r_{u,i}} = b_{u,i} + \sum_{j \in \boldsymbol{S^{I}_{u,i}}} \boldsymbol{w_{i,j}}(r_{u,j} - b_{u,j})$$

- $\boldsymbol{S^{I}_{u,i}}$: set of items rated by user u and similar to item i
- $\boldsymbol{w_{i,j}}$: weights between similar item pairs

- How to obtain $\boldsymbol{w_{i,j}}$?
- Regression! **Minimize SSE**: $\sum_{(\boldsymbol{u},\boldsymbol{i}) \in \boldsymbol{R}} (\widehat{\boldsymbol{r_{u,i}}} - \boldsymbol{r_{u,i}})^2$

$\boldsymbol{R}$ contains every rated pair of user u and item i

Optimization

- Objective function SSE:

$$L(\boldsymbol{w}) = \sum_{(u,i)\in R}(\underbrace{[b_{u,i} + \sum_{j\in S^I_{u,i}} w_{i,j}(r_{u,j} - b_{u,j})]}_{\text{Predicted rating}} - \underbrace{r_{u,i}}_{\text{True rating}})^2$$

- Gradient Descent!
- $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \boldsymbol{\nabla}_{\boldsymbol{w}} \boldsymbol{L}$

- Still treat users and items **independently**.
  - $f(u,v) = f(u) + f(v)$

Modeling User-item Interaction

- Linear formulation: $\widehat{r_{u,i}} = \mu + b_u + b_i$

- μ is the overall mean item ratings
- b_u is the user bias
- b_i is the item bias

- Second-order: $\widehat{r_{u,i}} = \mu + b_u + b_i + \sum_{u,i}(w_u x_u) \times (w_i x_i)$
  - Optimizing w_u and w_u is affected by $x_u x_i$ <= **interaction!**

- x_u, x_i are user and item features
- w_u, w_i are weights

- Expensive features
- High dimensionality of features might overwhelm learning
- Few samples to learn well due to sparsity

Matrix Factorization (MF)

- Reduce to: $\widehat{r_{u,i}} = \mu + b_u + b_i + \sum_{u,i} p_u q_i$

p_u, q_i are some **easy-to-get**, and **low-rank** features of user and item

- If we ignore μ, b_u, b_i and rewrite to matrix form: $\boldsymbol{R} \approx \boldsymbol{P}\boldsymbol{Q}^{\boldsymbol{T}}$
    - $\boldsymbol{P} \in \mathrm{R}^{|U| \times m}$ compose latent features of users
    - $\boldsymbol{Q} \in \mathrm{R}^{|V| \times m}$ compose latent features of items

$\boldsymbol{R}$ is the utility matrix

$\boldsymbol{R}$

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

$\approx$

$\boldsymbol{P}$

8.0	-1.5	-0.1
7.2	2.9	1.6
10.	-1.7	2.4
8.5	-1.0	-1.6
10.	2.3	-1.3
8.5	-0.7	-1.0

$\boldsymbol{Q}^{\boldsymbol{T}}$

.28	.24	.21	.35	.33	.31	.35	.36	.34	.33
-.35	.15	-.12	.06	-.25	.34	-.34	.70	-.16	-.11
-.03	.26	.53	.00	-.22	-.61	-.29	.17	.31	.08

6 users vs 10 items, and $m = 3$

Is MF What We Want?

$\boldsymbol{R}$

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

$\approx$

8.0	-1.5	-0.1
7.2	2.9	1.6
10.	-1.7	2.4
8.5	-1.0	-1.6
10.	2.3	-1.3
8.5	-0.7	-1.0

$\boldsymbol{P}$

.28	.24	.21	.35	.33	.31	.35	.36	.34	.33
-.35	.15	-.12	.06	-.25	.34	-.34	.70	-.16	-.11
-.03	.26	.53	.00	-.22	-.61	-.29	.17	.31	.08

$\boldsymbol{Q}^T$

- Easy-to-get data? **Yes!**
  - Need only the utility matrix
  - No additional feature engineering
- Low-rank? **Yes!**
  - On-demand size m
- Wait, what is this exactly?

Latent Factor (a.k.a. Embedding)

User

8.0	-1.5	-0.1
7.2	2.9	1.6
10.	-1.7	2.4
8.5	-1.0	-1.6
10.	2.3	-1.3
8.5	-0.7	-1.0

$\boldsymbol{P}$

Latent user factor
Or user embedding

Item

.28	.24	.21	.35	.33	.31	.35	.36	.34	.33
-.35	.15	-.12	.06	-.25	.34	-.34	.70	-.16	-.11
-.03	.26	.53	.00	-.22	-.61	-.29	.17	.31	.08

$\boldsymbol{Q^T}$

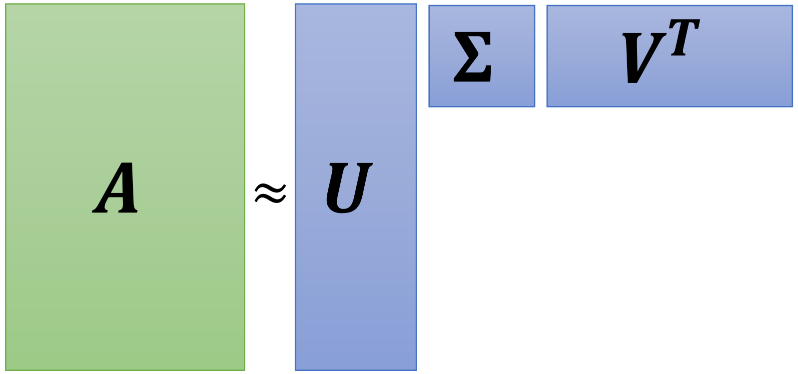
Latent item factor
Or item embedding

- Latent factor is the **hidden feature** that captures underlying patterns in the behavioral data
  - Abstractive concepts of directors, genres, and popularity
  - Data-driven
- Interpretability ... Meh ...
- Optimization? Hint: linear algebra

First Approach: SVD

- Matrix factorization $\boldsymbol{R} \approx \boldsymbol{P}\boldsymbol{Q}^{\boldsymbol{T}}$
- **Singular Value Decomposition** (SVD): $\boldsymbol{A} \approx \boldsymbol{U\Sigma V^T}$
  - SVD always yields minimal reconstruction loss: $\sum_{i,j}(A_{i,j} - [U\Sigma V^T]_{i,j})^2$
  - Wait… isn't it **SSE**!? JACKPOT!

- **Sadly, SVD isn't designed for missing values**
- Need other approaches to optimize MF



Second Approach: Gradient Descent (GD)

- Revisit objective function:

$$\min_{\mathbf{P},\mathbf{Q}} L^{MF} = \sum_{(u,i)\in R} (r_{u,i} - \widehat{r_{u,i}})^2 = \sum_{(u,i)\in R} (r_{u,i} - p_u \cdot q_i)^2$$

- To prevent overfitting, we add L2 regularization:

$$\min_{\mathbf{P},\mathbf{Q}} L^{MF} = \sum_{(u,i)\in R} (r_{u,i} - p_u \cdot q_i)^2 + \lambda_1 \sum_u \|p_u\|^2 + \lambda_2 \sum_i \|q_i\|^2$$

- Initialize $\mathbf{P}, \mathbf{Q}$ with small non-zero random values
- Do gradient descent:
  - $\boldsymbol{P} \leftarrow \boldsymbol{P} - \eta \nabla_{\boldsymbol{P}} \boldsymbol{L}^{\boldsymbol{MF}}$
  - $\boldsymbol{Q} \leftarrow \boldsymbol{Q} - \eta \nabla_{\boldsymbol{Q}} \boldsymbol{L}^{\boldsymbol{MF}}$

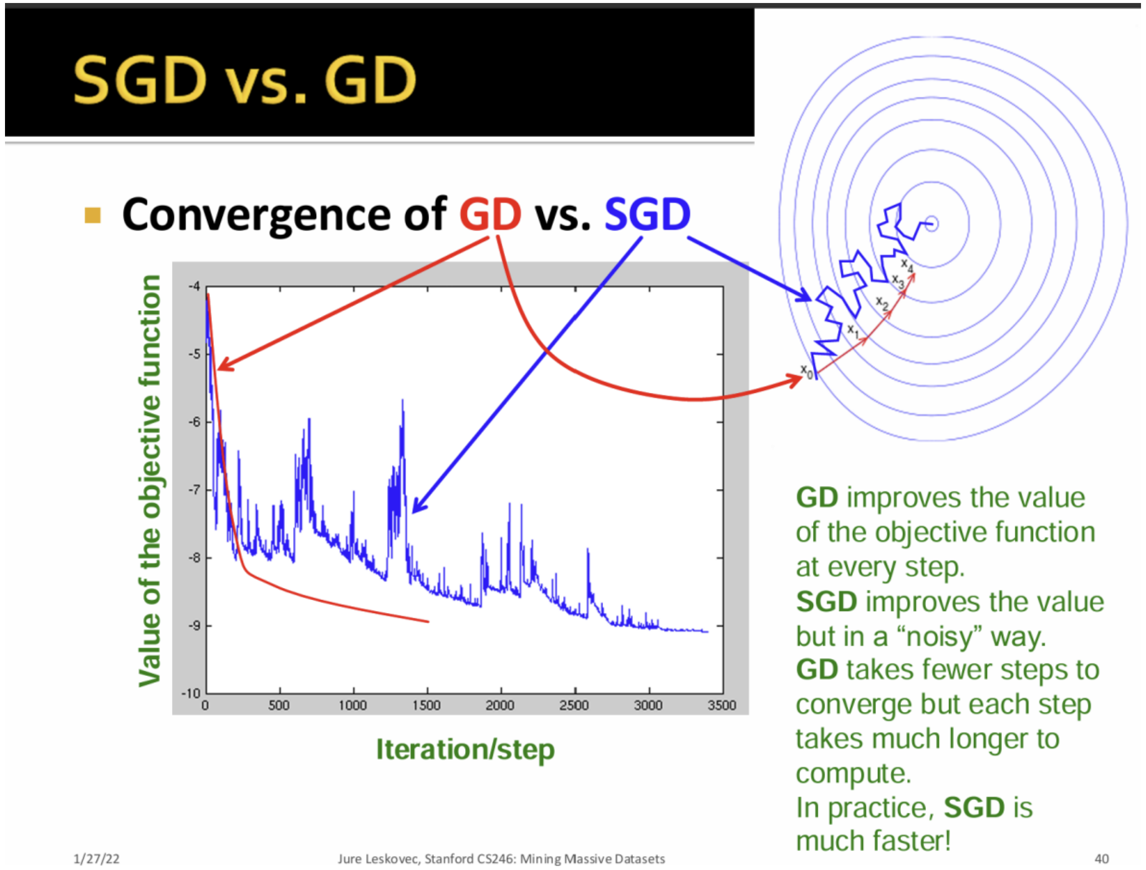
Stochastic Gradient Descent (SGD)

- Matrix operation was **slow for CPU**
  (but fast for GPU)

- **GD**: $\boldsymbol{P} \leftarrow \boldsymbol{P} - \eta \sum_{\boldsymbol{r}_{u,i}} \nabla_{\boldsymbol{P}} \boldsymbol{P}(\boldsymbol{r}_{u,i})$
- **SGD**: $\boldsymbol{P} \leftarrow \boldsymbol{P} - \eta \nabla_{\boldsymbol{P}} \boldsymbol{P}(\boldsymbol{r}_{u,i})$

- **Evaluate on all ratings** vs.
- **Evaluate on one rating**

- **Fast convergence**



Credit: Jure Leskovec, Stanford CS246

Add Biases Back

- Overall objective function

Goodness of fit

$$\min_{\mathbf{P},\mathbf{Q},\mathbf{b}} L^{MF} = \sum_{(u,i)\in R}[r_{u,i} - (\mu + b_u + b_i + p_u \cdot q_i)]^2$$

$$+\lambda_1 \sum_u ||p_u||^2 + \lambda_2 \sum_i ||q_i||^2 + \lambda_3 \sum_u ||b_u||^2 + \lambda_4 \sum_i ||b_i||^2$$

Regularization

- Learn p_u, q_i, b_u, b_i with SGD
- Select λ with the validation set

- Further reading: Alternating Least Square (ALS)

Pros and Cons for Matrix Factorization

Pros	Cons
No need of feature engineering	Poor interpretability
Model user-item interactions	Additional hyperparameters λ, m
Capture hidden relationships	Cold-start user and item problem
Data-driven	Sparsity issue

Outline

- Introduction
- Content-based Approaches
- Collaborative Filtering
- Matrix Factorization
- Ranking Objective
- Takeaways
- Appendix: Glossary of Hot Topics
- References

Pointwise Recommendations

- Classification, regression, etc
  - “Predicting whether a user will rate a movie 5 stars.”
  - “Predicting the rating a user will rate.”
  - **Use rated data only** => abandon huge amount of unrated data
  - **Implicit feedback** => severe imbalance; unknown or dislike?
- **Rethink: is rating that important in recommendations?**
  - Given movie prediction={A:4.6, B:4.7, C:4.5}, recommend which one?
- **Feasible to implicit feedback**
- **Controllable imbalance**

Relative Preference

- Another mindset: **Which one is better? Item $\boldsymbol{i}$ or $\boldsymbol{j}$?**
- We care about their **relative preference**, not their ratings
  - $i > j$ or $i < j$?
  - *Generalize* definition of preference <- **implicit feedback**

Purchase > add-to-cart
Click > seen
Finish > skip
Interacted > not interacted

- **Given user $\boldsymbol{u}$ likes item $\boldsymbol{i}$ better than $\boldsymbol{j}$**
- **For $\boldsymbol{u}$, maximize the probability of ranking item $\boldsymbol{i}$ over $\boldsymbol{j}$**
  - Pointwise: $\min \sum (\widehat{\boldsymbol{r}_{u,i}} - \boldsymbol{r}_{u,i})^2$
  - **Pairwise:** $\boldsymbol{\max \sum P\left(\widehat{r_{u,i}} > \widehat{r_{u,j}} \;\middle|\; r_{u,i} > r_{u,j}\right)}$

Pairwise Relation Dataset

- Define $i >_u j$, where u interacted with i but not j
- A training database O collects triplets (u, i, j) $s.t.$ $i >_u j$
  - Negative sample ratio? Let's say we collect just enough to train for now

	i_1	i_2	i_3	i_4
u_1	1	1		
u_2		1		1
u_3				1
u_4	1			1

$$O = \begin{Bmatrix} (u_1, i_1, i_3) \\ (u_1, i_2, i_3) \\ (u_1, i_1, i_4) \\ \vdots \end{Bmatrix}$$

Bayesian Personalized Ranking (BPR)

- Let Θ denote be learnable model parameter, we maximize

$$\underbrace{p(\Theta|i >_u j)}_{\max \sum P(\widehat{r_{u,i}} > \widehat{r_{u,j}} \mid r_{u,i} > r_{u,j})} \propto \underbrace{p(i >_u j|\Theta)}_{\text{Likelihood!}} \times \underbrace{p(\Theta)}_{\text{Follow Gaussian Distribution}}$$

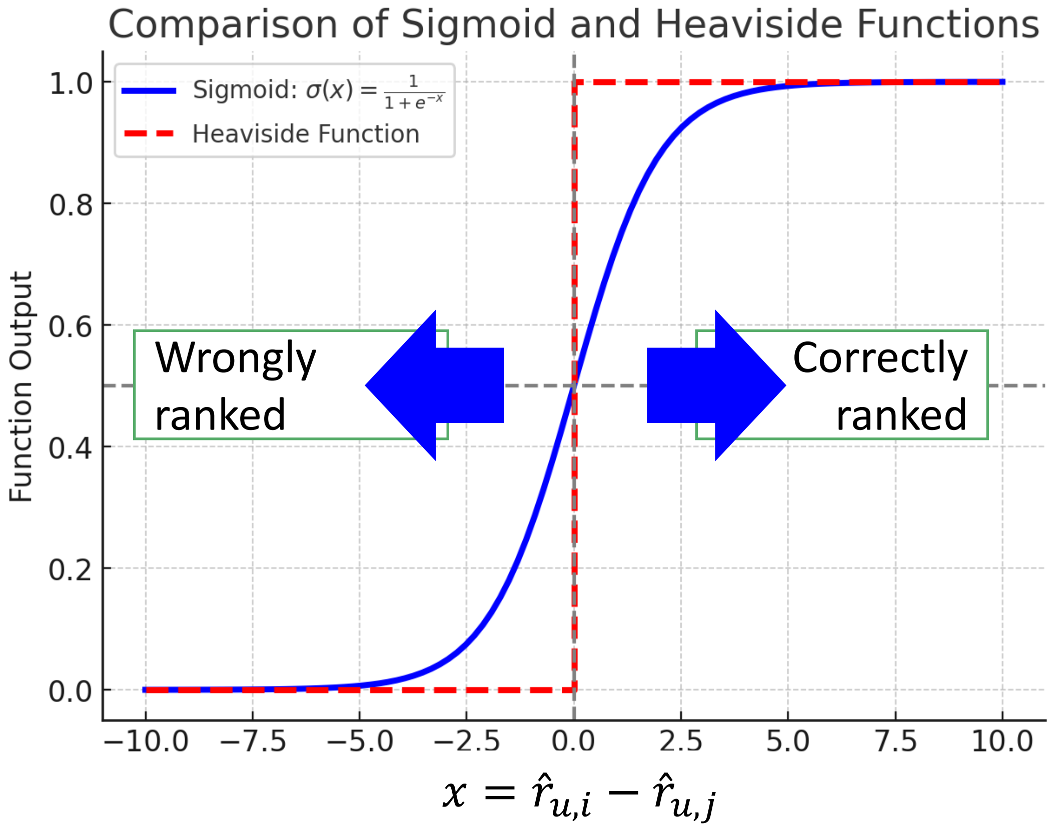
- Recall **sigmoid function** (**maximum likelihood** for *logistic regression*)

$$p(i >_u j|\Theta) = \sigma\left(\hat{r}_{u,i,j}(\Theta)\right) = \sigma(\hat{r}_{u,i,j})$$

$$\boxed{\sigma(x) = \frac{1}{1 + e^{-x}}}$$

BPR Objective

- Maximum log-likelihood function
- $BPR-Opt = \prod_{(u,i,j)\in O} p(\Theta|i >_u j)$
- $= \ln \prod_{(u,i,j)\in O} p(i >_u j|\Theta)p(\Theta)$
- $= \sum_{u,i,j} \ln \sigma\left(\hat{r}_{u,i,j}\right) + \ln p(\Theta)$
- $= \sum_{u,i,j} \ln \sigma\left(\hat{r}_{u,i,j}\right) - \frac{\lambda}{2}\|\Theta\|^2$



- $\hat{r}_{u,i,j}$ should <u>quantify</u> $i >_u j$
  - $\hat{r}_{u,i,j} = \hat{r}_{u,i} - \hat{r}_{u,j} = \begin{cases} > 0 & if\ ranked\ correctly \\ \leq 0 & else \end{cases}$

SGD

- $\Theta \leftarrow \Theta - \eta \nabla_\Theta$
- $\nabla_\Theta = \frac{\partial \text{ BPR}-Opt}{\partial \Theta} = \cdots = \sum_{u,i,j} \frac{e^{-(\hat{r}_{u,i}-\hat{r}_{u,j})}}{1+e^{-(\hat{r}_{u,i}-\hat{r}_{u,j})}} \cdot \frac{\partial}{\partial \Theta}(\hat{r}_{u,i} - \hat{r}_{u,j}) - \lambda\Theta$

- Fix $\hat{r}_{u,j}$ and learn $\hat{r}_{u,i}$, and vice versa

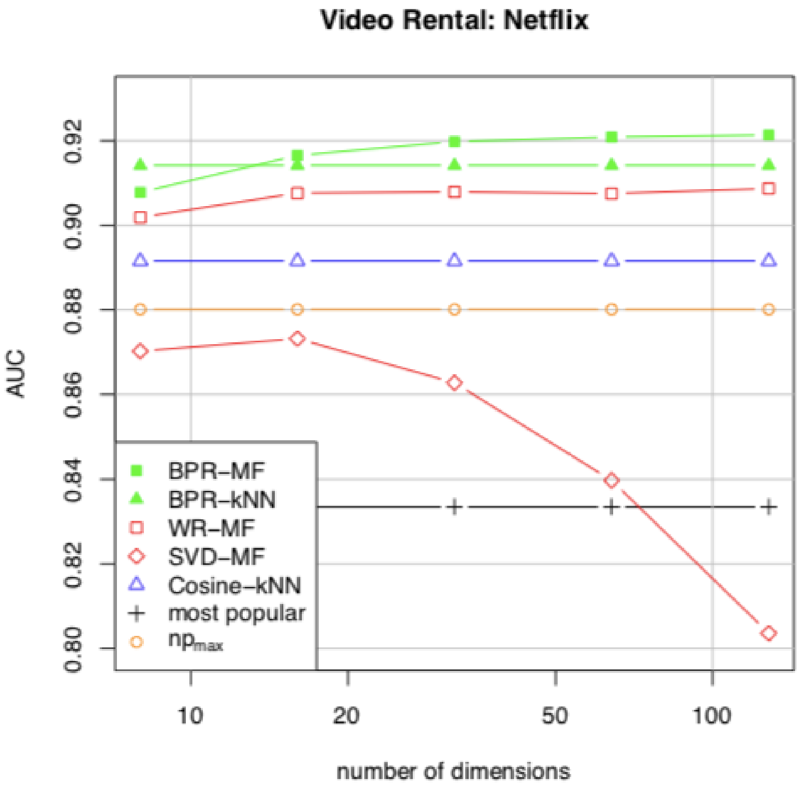
- MF-based formulation: $\hat{r}_{u,i} = \mu + b_u + b_i + p_u \cdot q_i$
- What is different for MF now?

BPR-MF vs. Vanilla MF

	BPR-MF	Vanilla MF (SVD)
Objective	Pairwise	Pointwise
Latent Factor	Explain relative orders	Explain ratings
Gradient of User	$f_u(x_i - x_j)$ based on feature deviation	$f_u(x_i)$ based on item feature
Gradient of Positive Item	$f_i(x_u)$	$f_i(x_u)$
Gradient of Negative Item	$f_j(-x_u)$ shift away from user feature	**None**

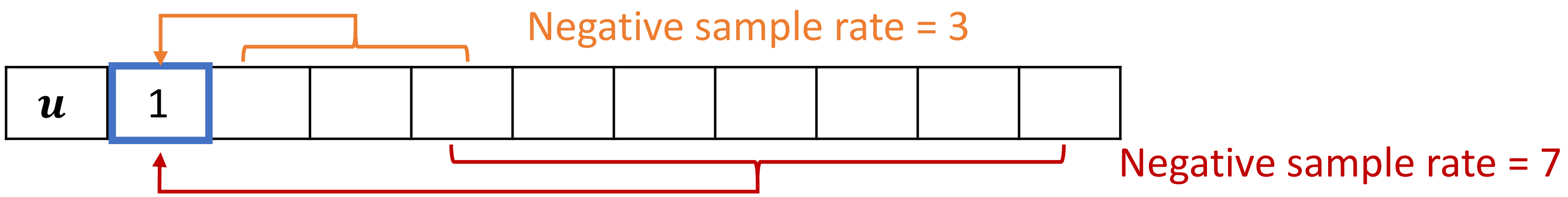
Performance Evaluation (AUC)

- AUC is the probability that a randomly chosen positive item is **ranked higher** than a randomly chosen negative item



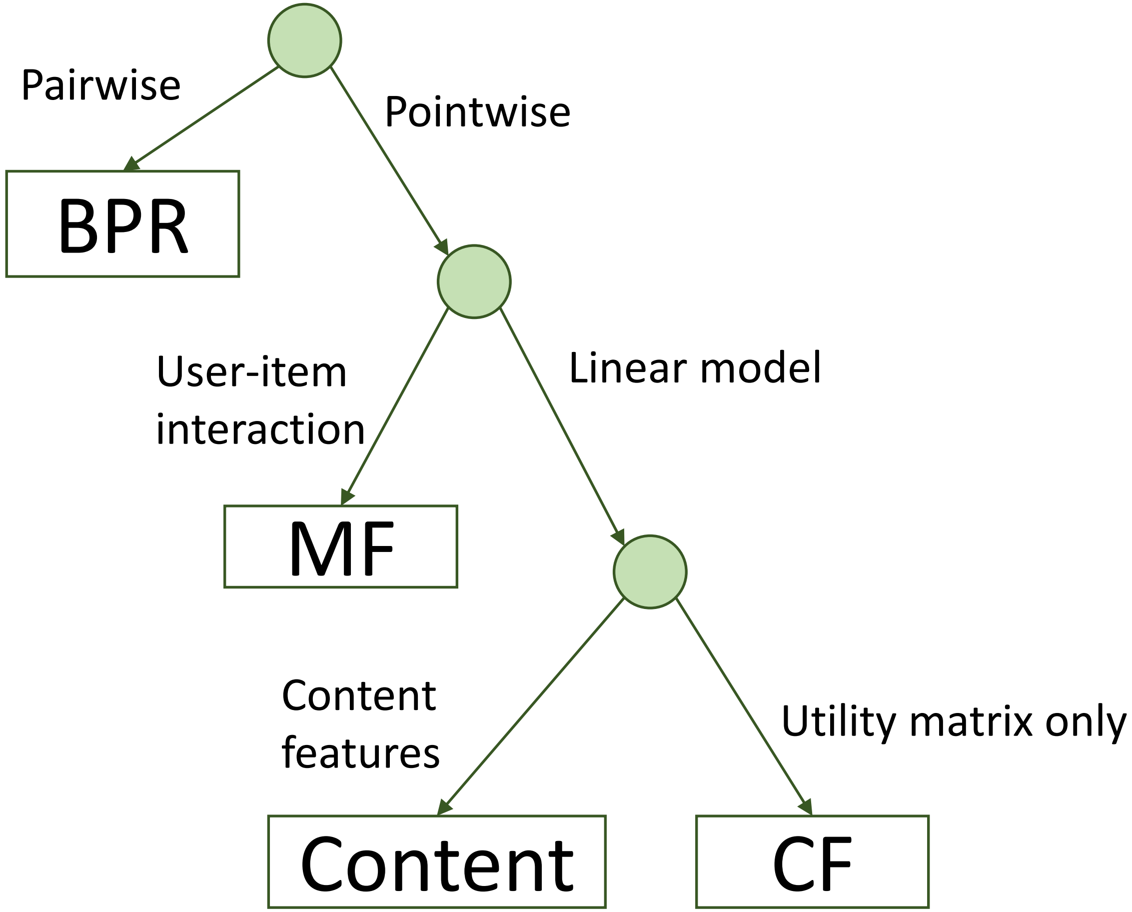
Pros and Cons of BPR

Pros	Cons
Directly model relative preference	Ignore absolute preferences
Incorporate implicit feedback	Computationally-intensive
Handles sparsity well	Cold-start problem
Great compatibility	Additional hyperparameter: **negative sample rate**



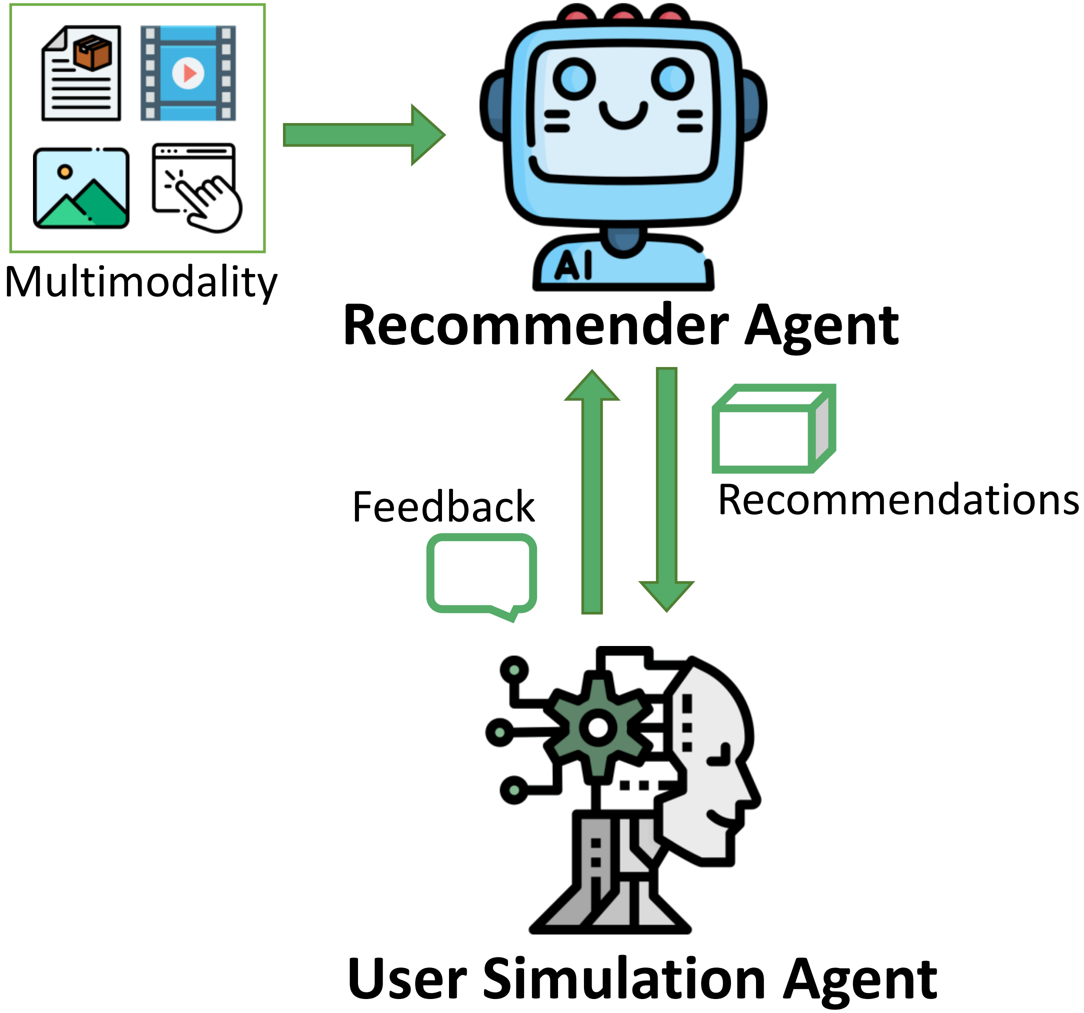
Takeaways

- Choose models according to data and scenarios
- Combine any of them is possible
- Cold-start and sparsity issue
- Compatible to new tech (e.g., NN)



Agentic Recommender System Era

- Pros
  - Easy to integrate multimodality
  - Unlimited feedback augmentation
  - Dynamic and continuous self-improvement
  - Flexible to extend novelty
- Cons
  - Intensive computational cost
  - Unintended outcomes
  - Unstable performances
  - Overamplified biases (data poisoning)
  - Security and privacy concerns



Appendix: Glossary of Hot Topics

- Adversarial Training
- Attention Network
- Contrastive Learning
- Cross-domain Rec.
- Data Augmentation
- Fairness Rec.
- Federated Learning
- Generative Models
- Graph Neural Networks
- Knowledge Graph-based Rec.
- Meta-learning
- Multi-modal Learning
- Neural Collaborative Filtering
- Self-supervised Learning
- Sequence Rec.
- Session-based Rec.
- Social, spatial, and temporal Rec.
- Transformer-based Rec.

References and Further Reading

- Jure Leskovec, *Stanford CS246 lecture note*, 2024
- Jingbo Shan, *UCSD DSC148 lecture note*, 2023
- Steffen Rendle et al.: BPR: *Bayesian Personalized Ranking from Implicit Feedback*. UAI 2009