Recommender Systems: Basics

Jaemin Yoo

School of Electrical Engineering

Kim Jaechul Graduate School of AI

Outline

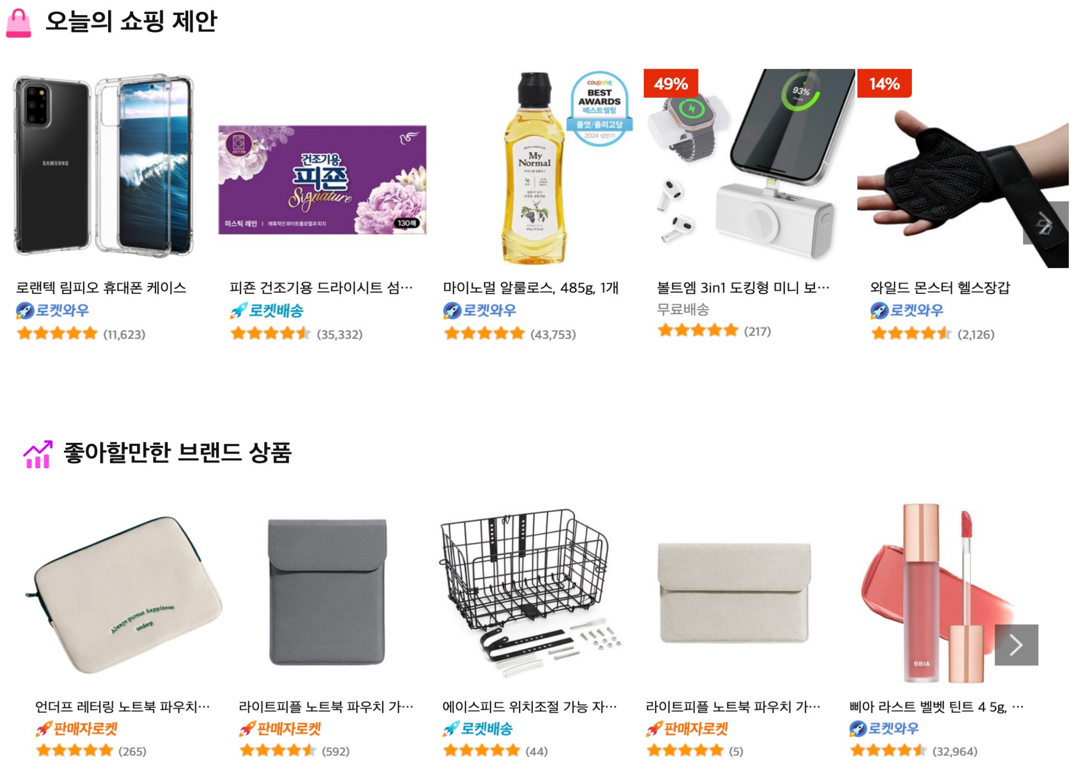
1. **<u>Introduction</u>**
2. Content-based models
3. Collaborative filtering
4. Latent factor models
5. Summary

Recommender Systems

- Class of applications that predict user responses to options.
- **Non-personalized recommendations:**
  - *Editorial and hand-curated:* List of favorites, essential items, …
  - *Simple aggregates:* Top 10, most popular, recent uploads, …
- **Personalized recommendations:**
  - *Tailored to individual users:* Amazon, Netflix, YouTube, …

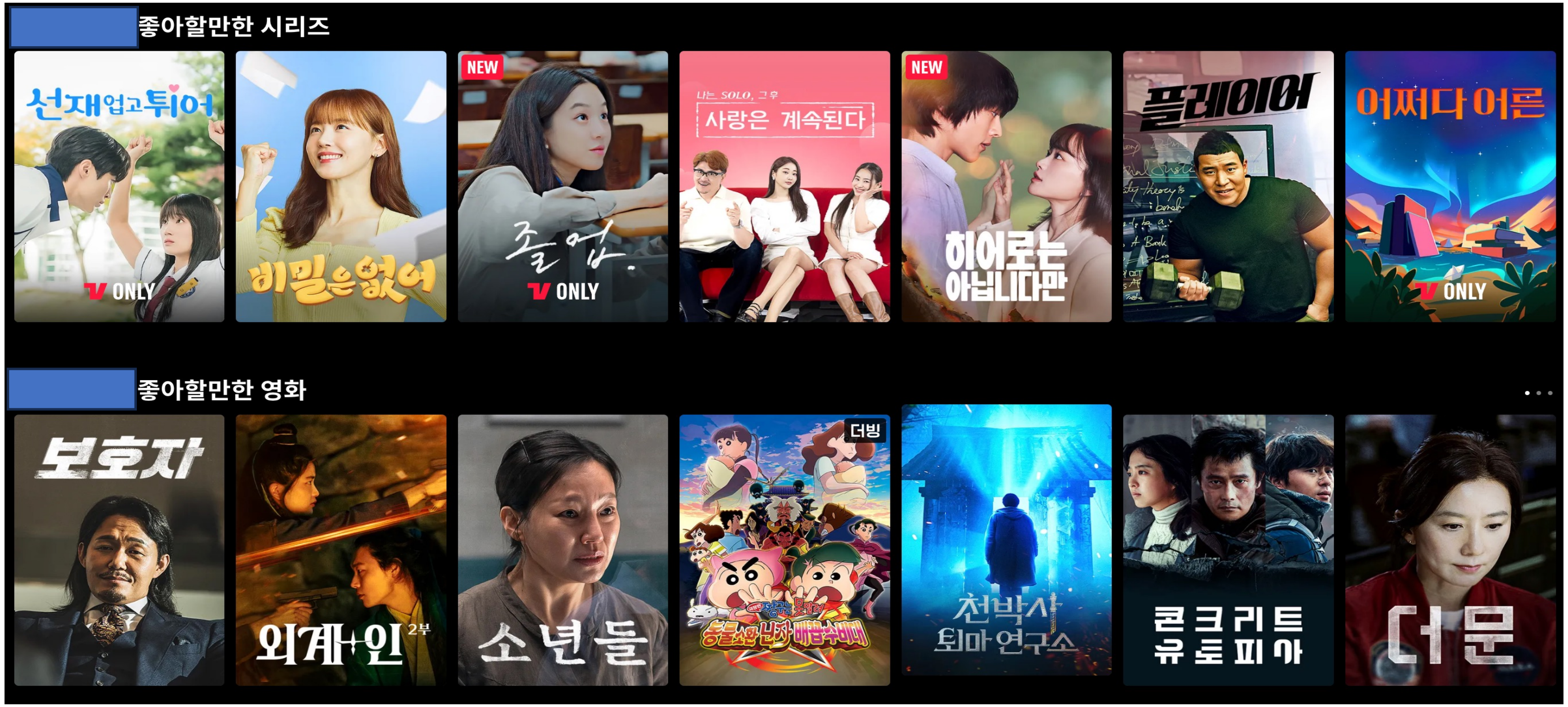
Recommender Systems: Examples

- Personalized recommender systems in Coupang:



Recommender Systems: Examples

- Personalized recommender systems in Netflix:

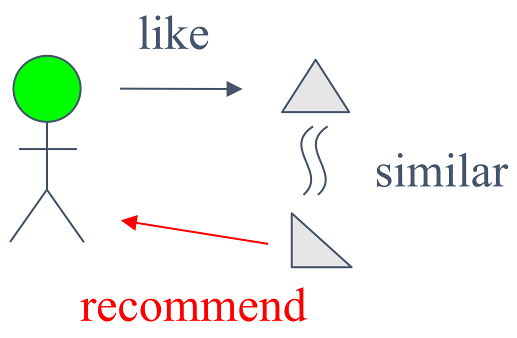


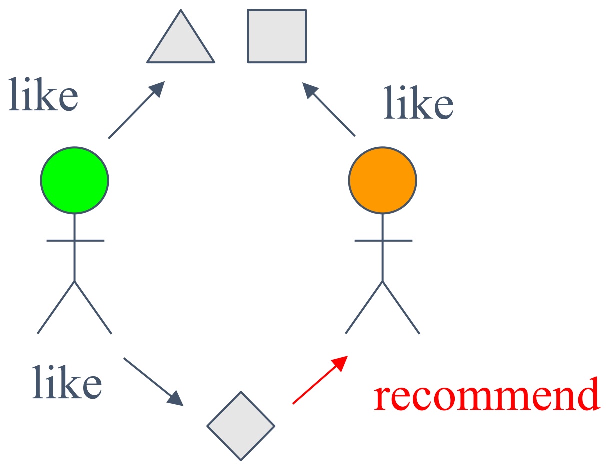
Recommender Systems: Examples

- Item-based (not user-based) recommender systems:

Personalized Recommender Systems

- Two groups of recommender systems:
  - **Content-based:** Focus on the profiles (or features) of users/items.
  - **Collaborative filtering:** Focus on the interactions between users/items.
  - **Hybrid:** Use both content-based and interaction-based information.





Utility Matrix

- We consider two classes of entities: **Users** and **items**.
- **Utility matrix** shows the preference of users for items.
  - The values come from an ordered set, e.g., 1-5 stars.
  - Assumed to be sparse, i.e., most entries are unknown.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Gathering Ratings

- **Explicit feedback:** Ask users to rate items.
  - E.g., YouTube asks for likes/dislikes of watched videos.
  - Users are generally unwilling to provide responses.
  - *Biased* as it comes from people willing to provide ratings.
- **Implicit feedback:** Learn ratings from user actions.
  - If a user watches a movie, the user is said to “like” it.
  - Hard to model low ratings: 0 (no rating) or 1 (like).

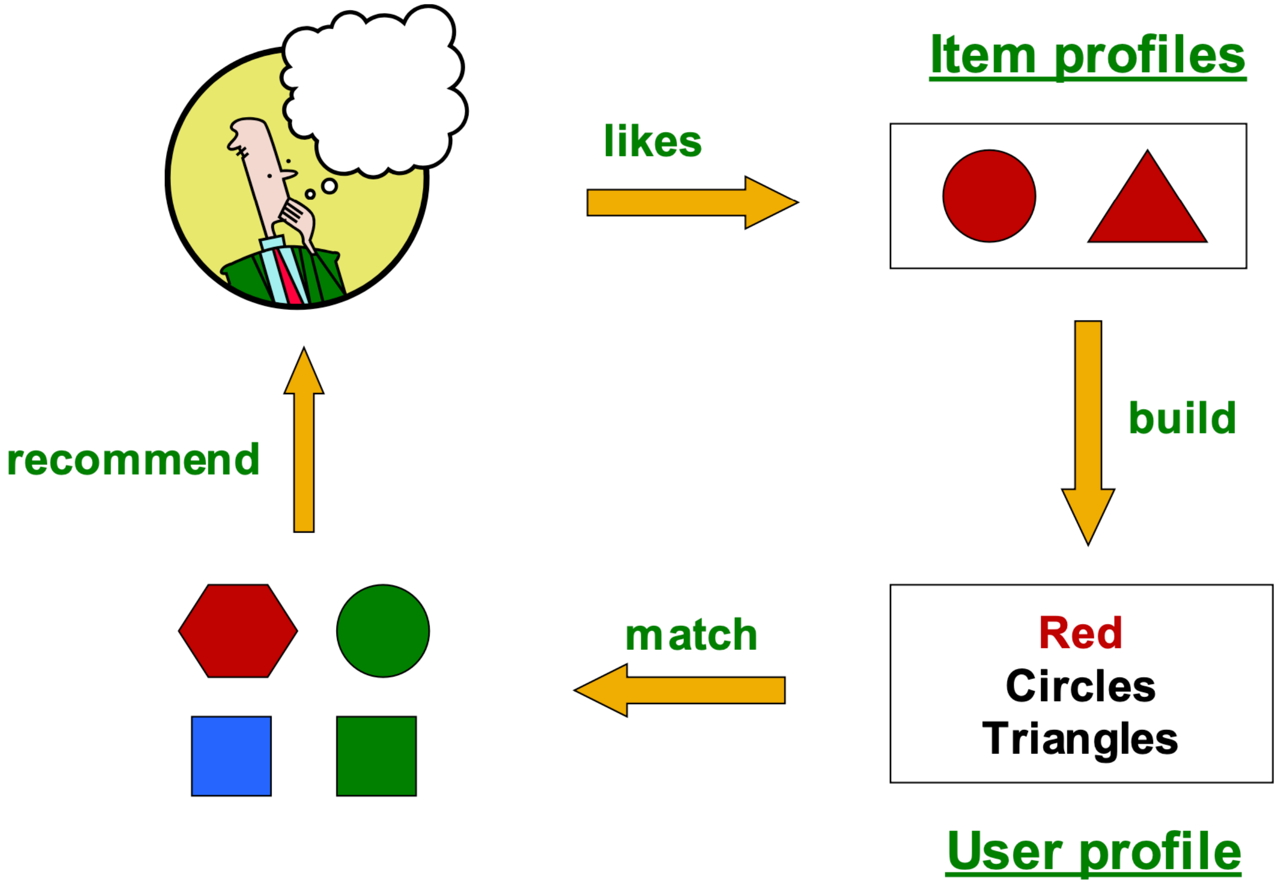
Outline

1. Introduction
2. **<u>Content-based models</u>**
3. Collaborative filtering
4. Latent factor models
5. Summary

Content-based Recommendation

- **Main idea:** Use the profiles of items.
  - E.g., for a movie, { genre, director, actors, plot, release year }
  - Recommend items similar to previous highly-rated items.
- **Examples:**
  - Recommend movies with same actor(s), director, genre, …
  - Recommend websites or blogs with “similar” content.

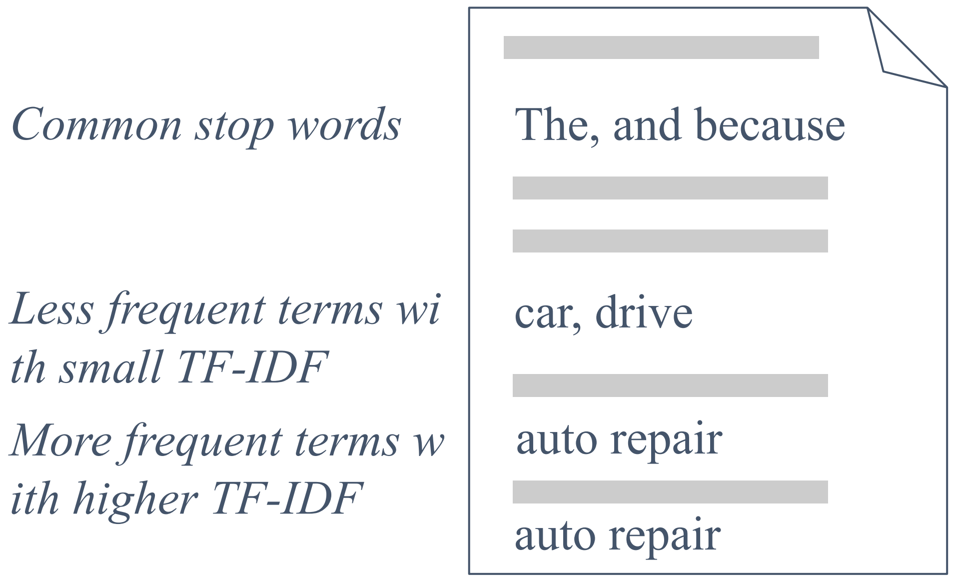
Plan of Action



Source: Stanford CS246 (2022)

Item Profiles

- For each item, we create an **item profile** as a set of features.
- Recently, deep learning is used to find a good item profile.
  - E.g., a CNN is used to create latent information of item images.
- **Method 1:** Use pre-trained models.
- **Method 2:** Train an extractor in an end-to-end way.
  - Minimizing a recommendation loss.



Content-based: Pros and Cons

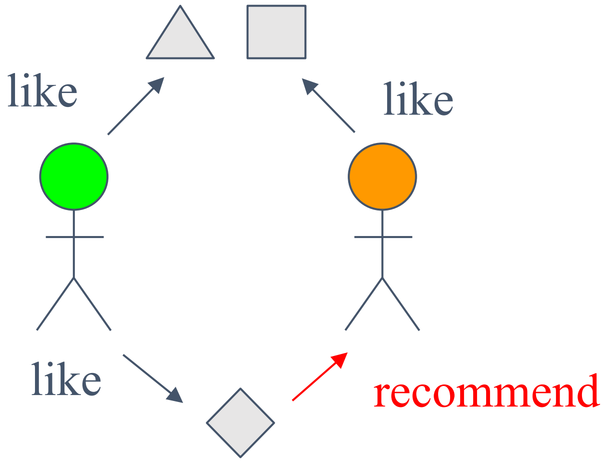
- **Pros:**
  - No need for data on other users.
  - Able to recommend items to users with unique tastes.
  - Able to recommend new or unpopular items.
  - Able to provide explanations (by listing content features).
- **Cons:**
  - Finding the appropriate features may be difficult.
  - New users may not have a profile.
  - **Overspecialization:** Cannot recommend beyond user's profile.

Outline

1. Introduction
2. Content-based models
3. **<u>Collaborative filtering</u>**
4. Latent factor models
5. Summary

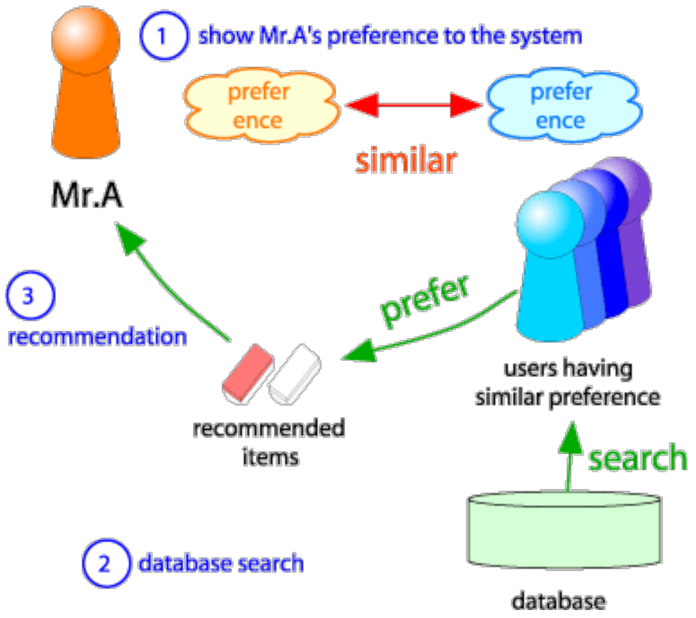
Collaborative Filtering

- **Collaborative filtering** focuses on the interactions, not contents.
  - Does not build item profiles or user profiles.
  - Uses rows/columns of the utility matrix as profile vectors.
- Comes in two flavors:
  - User-user collaborative filtering.
  - Item-item collaborative filtering.



User-User Collaborative Filtering

- Given a user U, find users whose ratings are similar to U's ratings.
- Estimate U's ratings based on the similar users.



Finding Similar Users

- There are various ways for defining which users are "similar."
- **Jaccard similarity:**
  - Treat ratings as sets, ignoring the values (i.e., likes vs. dislikes).
  - For this example, it seems intuitively wrong.
    - Since $\mathrm{Jaccard}(A, B) = 1/5 < \mathrm{Jaccard}(A, C) = 2/4$. Does it make sense?

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Finding Similar Users

- There are various ways for defining which users are "similar."
- **Cosine similarity:**
  - Treat ratings as points (or vectors), considering blanks as 0.
    - Questionable, since *no rating* doesn't mean *dislike*.
    - In this example, $v_A^\top v_B / \|v_A\| \|v_B\| = 0.380$ and $v_A^\top v_C / \|v_A\| \|v_C\| = 0.322$.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Rating Predictions

- **From similarity metrics to recommendations:**
  - Let r_x be the vector of user x's ratings.
  - Let N be the set of k users most similar to user x.
  - Let $N' \subseteq N$ be the subset of users who rated item i.
- Prediction for item i of user x:
  - **Simple version:** $r_{xi} = \frac{1}{k'} \sum_{y \in N'} r_{yi}$.
  - **Complex version:** $r_{xi} = \frac{\sum_{y \in N'} \mathrm{sim}(x,y) \cdot r_{yi}}{\sum_{y \in N'} \mathrm{sim}(x,y)}$.

Item-Item Collaborative Filtering

- **Another view: Item-item collaborative filtering.**
  - For item i, find other similar items.
  - Estimate rating for item i based on ratings for similar items.
  - Can use the same similarity metrics as in the user-user model.

$$r_{xi} = \frac{\sum_{j \in N(i;x)} \mathrm{sim}(i,j) \cdot r_{xj}}{\sum_{j \in N(i;x)} \mathrm{sim}(i,j)}$$

- $N(i; x)$ is the set of items similar to item i and rated by user x.

Item-Item vs. User-User

- **Item-item similarity is often more reliable.**
  - Intuitively, items are classifiable in simple terms, e.g., one genre.
  - Users may like multiple genres, so harder to compute similarity.
- However, there is no clear advantage of one from another.
  - E.g., user-user is better for relatively new items.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Collaborative Filtering: Pros and Cons

- **Pros:**
  - Do not have to come up with features (or profiles).
- **Cons:**
  - Need enough users in the system to find a match.
  - Cannot recommend new or unpopular items that have not been rated.
  - Cannot recommend items to someone with unique taste.
    - I.e., tends to recommend popular items.

Hybrid Approach

- Advanced recommender systems are **hybrid** and **multi-modal**.
  - **Hybrid:** Use both content and interaction information.
  - **Multi-modal:** Use different data modalities at the same time.
    - Textual reviews from users.
    - Image description of items.
    - Graph-structured interactions between users and items.
- There is no fixed way in deep learning.
  - Since different components can be combined in an end-to-end way.

Outline

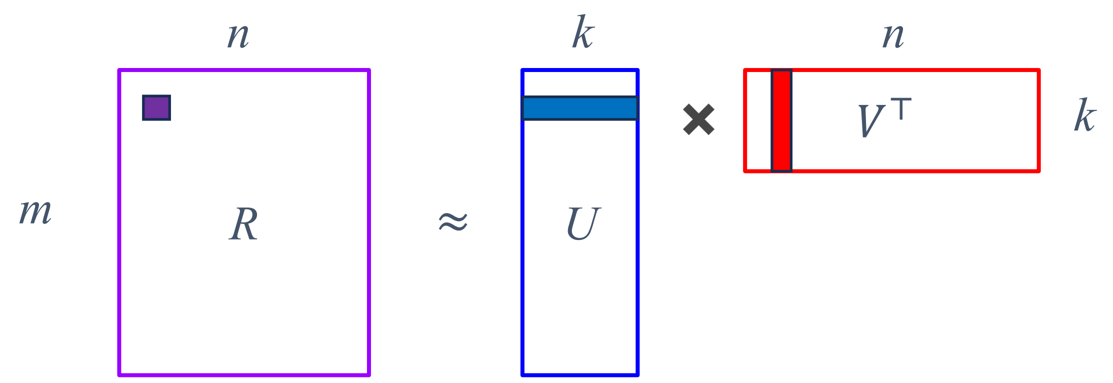
1. Introduction
2. Content-based models
3. Collaborative filtering
4. **Latent factor models**
5. Summary

Latent Factor Models

- **Latent factor models** assume that:
  - There are **latent factors** that can represent users and items well.
  - Such latent factors can be extracted from the utility matrix.
- Many people consider latent factor models as a part of CF.
  - Since they share the same philosophy.
  - CF uses the rows and columns of R without modification.
  - Latent factor models extract (better) latent factors from R.

Latent Factor Models

- **Idea:** Consider a utility matrix as the product of **factor matrices**.
  - E.g., users react to certain genres, famous actors, or directors.
- **UV decomposition** decomposes a utility matrix into U and V.
  - Each user and movie is summarized as a low-dimensional vector.



UV Decomposition

- **Given** an $m \times n$ utility matrix R (i.e., m users and n items).
- **Find** an $m \times k$ matrix U and $n \times k$ matrix V such that:
  - $UV^\top$ closely approximates R for the **non-blank entries**.
- **Use** the elements of $UV^\top$ to estimate the blank entries in R.
  - Compute $\hat{r}_{xi} = u_x^\top v_i$ to predict r_{xi}.

$$\underset{R}{\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix}} = \underset{U}{\begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \\ u_{41} & u_{42} \\ u_{51} & u_{52} \end{bmatrix}} \times \underset{V^\top}{\begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} \end{bmatrix}}$$

Error Function

- **Root-mean-square error (RMSE)** measures the difference.

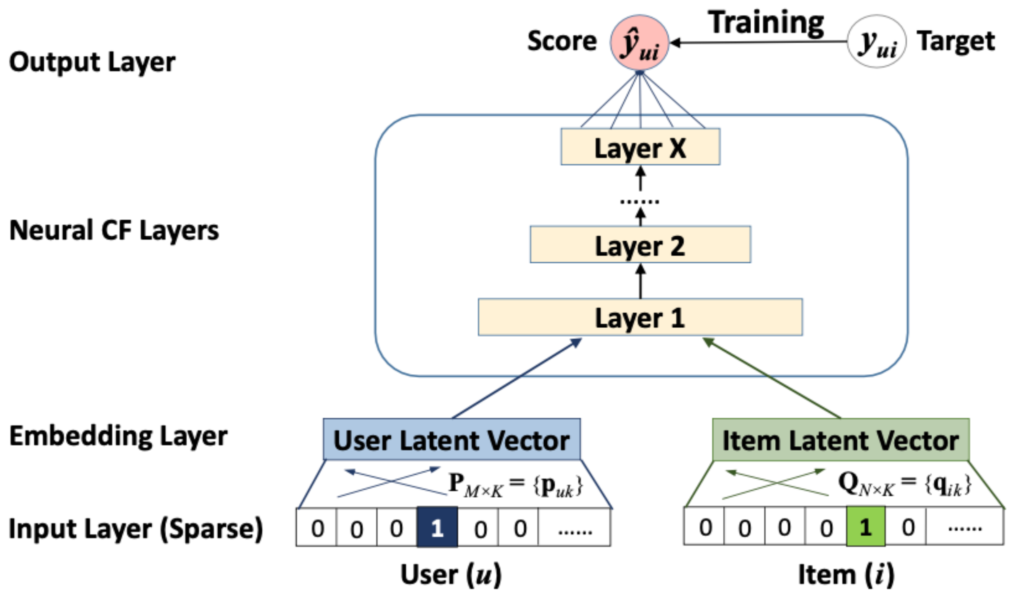
$$\mathrm{RMSE}(\hat{R}, R) = \mathrm{sqrt}\left(\frac{1}{|E|}\sum_{(x,i)\in E}(\hat{r}_{xi} - r_{xi})^2\right)$$

  - E is the set of non-blank entries.

$$\mathrm{RMSE}\Big(\underbrace{\begin{bmatrix} 5 & 2 \\ 3 & \end{bmatrix}}_{R}, \underbrace{\begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}}_{UV^\top}\Big) = \sqrt{\frac{(5-2)^2+(2-2)^2+(3-2)^2}{3}} = 1.826$$

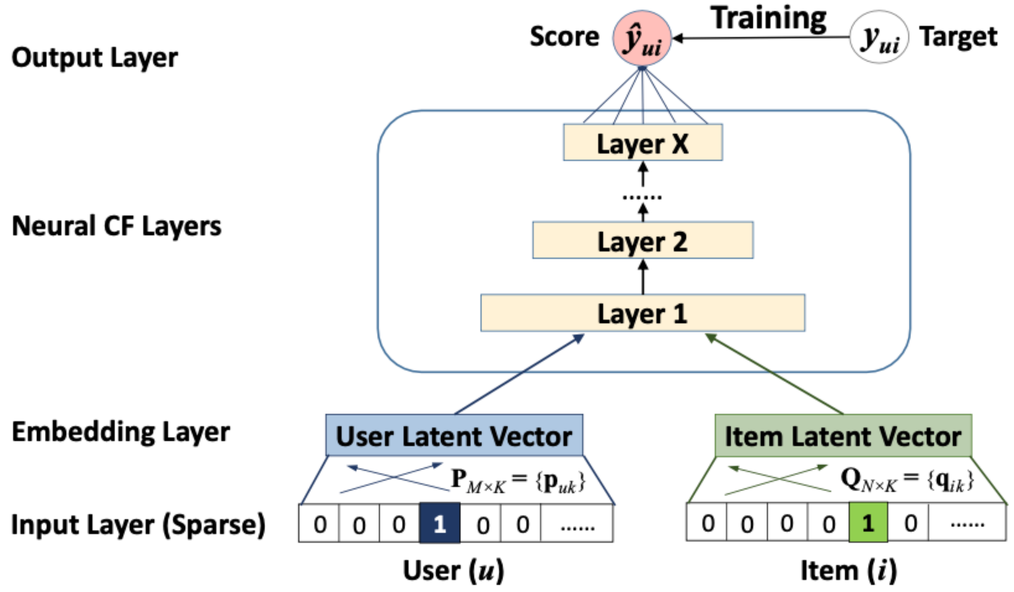
Neural Collaborative Filtering

- The classical latent factor model is still a **linear** method.
- **Neural collaborative filtering** captures complex interactions.
  - By utilizing the nonlinearity of neural networks.



Neural Collaborative Filtering

- **Input & embedding layers:**
  - Not different from the latent factor model.
- **Neural CF layers:**
  - Generalizes the (linear) dot product.
  - Take concatenated $h_u \parallel h_v$ as input.
- **Training:**
  - Can use a proper loss for each data.



Dealing with Implicit Feedback

- What if the utility matrix R contains only implicit feedback?
  - Each entry is either 0 (not watched) or 1 (watched).
- Training the model with RMSE loss is not desirable.
  - RMSE assumes 0 as a dislike, not “not watched.”
  - Model will be trained not to recommend all unwatched movies.

Ranking Loss

- **Idea:** Let's consider the task as **ranking**, not *elementwise prediction.*
  - Given a user x, suppose that $r_{xi} = 1$ while $r_{xj} = 0$.
  - It's hard to assume that user x **dislikes** movie j.
  - But we can safely assume that user x likes i **more than** j.
    - If x really likes j, they would have watched it before i.
  - Let's train the model by comparing items, so that $u_{xi} > u_{xj}$.

Bayesian Personalized Ranking

- We may use the **Bayesian personalized ranking (BPR)** loss:

$$J_{\mathrm{BPR}}(U, V) = \sum_{x,i,j} -\log \sigma\left(u_x^\top v_i - u_x^\top v_j\right)$$

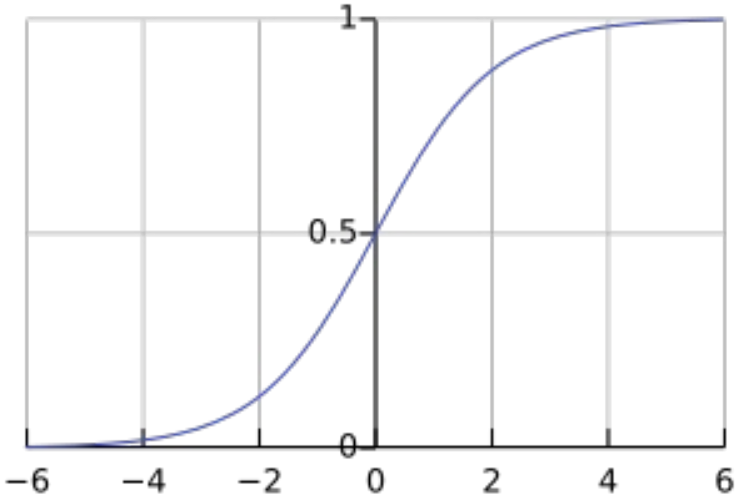
  - Item j is randomly selected from the **negative samples** $\{\, j \mid r_{xj} = 0 \,\}$.
- In this way, the model is trained to satisfy $u_x^\top v_i > u_x^\top v_j$.
- The **sigmoid function** σ is used to balance the difference.

Sigmoid Function

- **Sigmoid function** σ limits the output to be $[0, 1]$:

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

  - Maps $(-\infty, \infty)$ to $(0, 1)$ with $\sigma(0) = 0.5$.
  - Monotonically increasing for all ranges of x.



Outline

1. Introduction
2. Content-based models
3. Collaborative filtering
4. Latent factor models
5. **Summary**

Summary

- Recommendation is an essential task in data mining.
- Collaborative filtering is one of the most popular approaches.
  - Models the interactions between users and items.
- Deep models improve collaborative filtering via nonlinearity.
- Modern approaches utilize the structure under the given data.