Personalized Machine Learning

Memory-based Approaches

Rodrigo Alves

October 16, 2025

Recommender Systems

- Recommenders can recommend:
  - Items to users (most common)
  - Users to items
  - Items to items
  - Users to users
- Items can include movies, products, news, music, books, recipes, and more.

Working in pairs: Open an app (e.g., Instagram) or visit a website (e.g., seznam.cz) and find one example of **each** of the four recommender scenarios above that is already applied or where you think it could be implemented.

Recommender Systems

Recommenders are mostly seen as two different learning problems:

- Predictive modeling: predicting the rating of item j by user i.
- Retrieval modeling: learning ranking systems – User i prefer item j_1 more than item j_2.

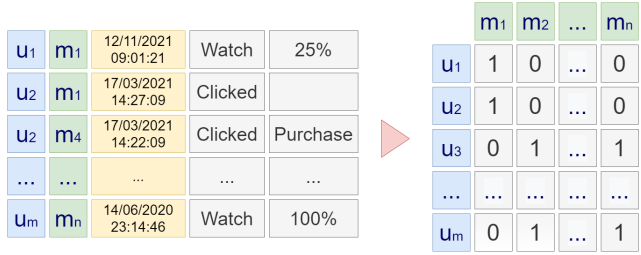
Typically, these systems are based on **past interactions** and (or) **attributes** (from users and items).

- **Interactions:** Normally modeled as an interaction matrix.
  - Explicit: When a user rates a song with 4 stars on a scale from 0 to 5.
  - Implicit: When a user watches 80
- **Attributes:** Normally modeled as attribute matrices.
  - Users: Gender, age, location, and more.
  - Items: Text, video, metadata, and more.

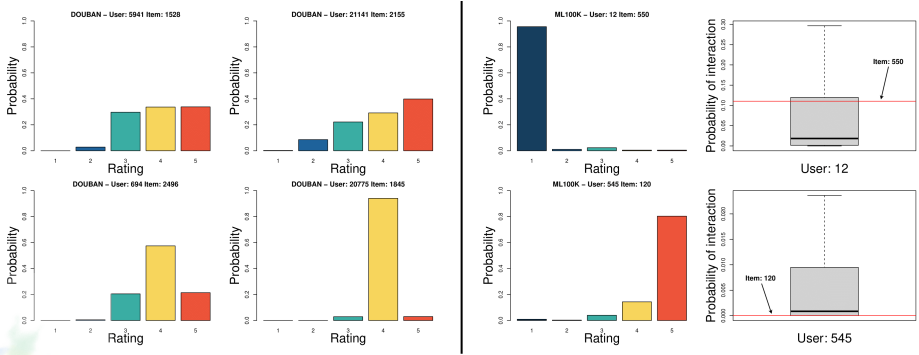
Modelling Interactions: Explicit feedback

	m_1	m_2	...	m_n
u_1	?	2	...	3
u_2	5	1	...	?
u_3	?	3	...	1
...	...	...	...	...
u_m	4	4	...	?

Modelling Interactions: Implicit feedback



Mixing Implicit and Explicit



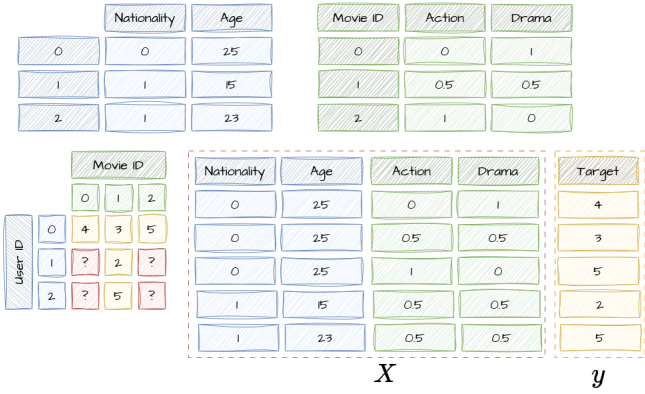
Personalized Machine Learning

- Personalization **is not** a simple regression or classification problem.
- A personalized model implies that if users have different interactions (or attributes), the recommendations *might* be different.
- Suppose the vectors a_i and a_j are attribute vectors of user i and item j, respectively.
- We can use linear regression to predict how user i will like item j:

$$r_{i,j} = \omega^\top \times \begin{bmatrix} a_i \\ a_j \end{bmatrix}$$

- Is linear regression a personalized model for recommenders? **Why?**

PML vs ML </>



$$\min_{\omega} ||y - X\omega||^2 + \lambda||\omega||^2$$

Recommendation Algorithms

Collaborative Filtering

Day One: Joe and Julia independently read an article on police brutality

Day Two: Joe reads an article about deforestation, and then Julia is recommended the deforestation article

Content-Based Filtering

Day One: Julia watches a Drama

Day Two: Dramas are recommended

Memory-based Approaches

- Memory-based approaches are the simplest and, perhaps, the most ubiquitous models in recommendation.
- Items are recommended because they are similar to one (or more) that the user has interacted with before.
- Choosing the correct method for computing similarity is the key to this approach.
- Similarity could be based on:
  - User/item attributes
  - **User-item interactions**

Does similarity depend on perception?

Modelling Attributes

	Color			Price	Category	
				Float	T-Shirt	Chair
m_1	1	0	0	0.9996	0	1
m_2	0	0.7	0.7	1.0000	0	1
m_3	0	0.7	0.7	0.0008	1	0
m_4	1	0	0	0.0000	0	1

How Could We Recommend from Attributes?

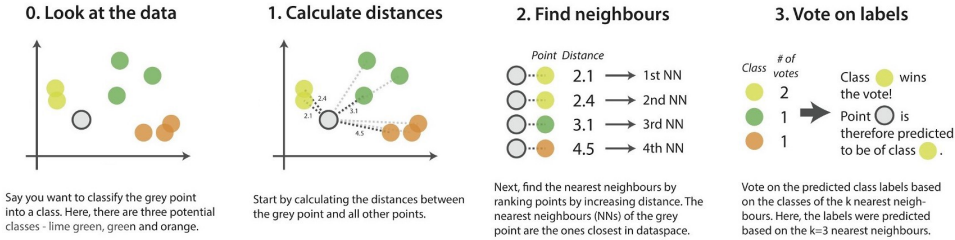


Figure source: kdnuggets

k-NN Algorithm for Recommendation

- k-NN is an instance-based method that can be either content-based or collaborative filtering.
- k-NN is also commonly used for regression and classification. More formally,

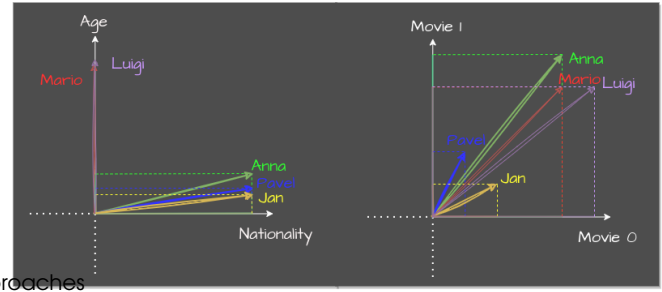
User-based Nearest Neighbors

- Let u_i be a feature vector of user i with an unknown rating $r_{i,j}$ for item j.
- Compute the **distance** $d(x, i)$ between u_i and all u_x.
- Find the k instances (neighbors) in u_x that are closest to u_i and have observed item j.
- Combine their $r_{x,j}$ values.
- Output the combination.

The simplest way of combining is by computing the average of k nearest instances. We will further discuss **distance** $d(x, i)$.

	Nationality	Age
Anna	1	0.31
Pavel	1	0.27
Jan	1	0.18
Mario	0	0.89
Luigi	0	0.91

	Movie ID		
	0	1	2
Anna	4	5	?
Pavel	1	2	5
Jan	2	1	4
Mario	4	4	1
Luigi	5	4	2



Weighted k-NN Algorithm

- In k-nearest neighbors, one can find users with different similarities.
- The prediction can take into consideration and give more weight according to users' similarities.
- For example, for the prediction of the rating of user i for item j, we have

$$\hat{r}_{i,j} = \frac{\sum_{l \in \mathcal{K}} \text{sim}(i,l) \times r_{l,j}}{\sum_{l \in \mathcal{K}} \text{sim}(i,l)}$$

where $\mathcal{K}$ represents the k nearest neighbors.

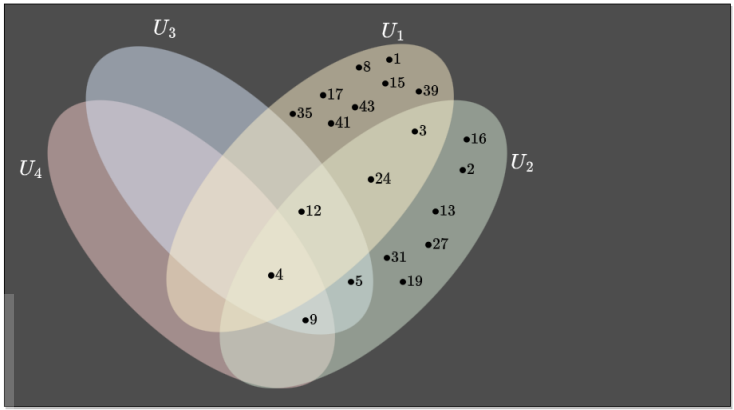
Defining a Similarity Function

- So far, all the examples that we have seen were well-defined because the vectors being compared were fully known.
- However, in Recommender Systems, we frequently have partially known vectors since we only observe part of the user's interactions.
- To simplify, let's consider the implicit feedback case where users interacted or not with an item. For instance:

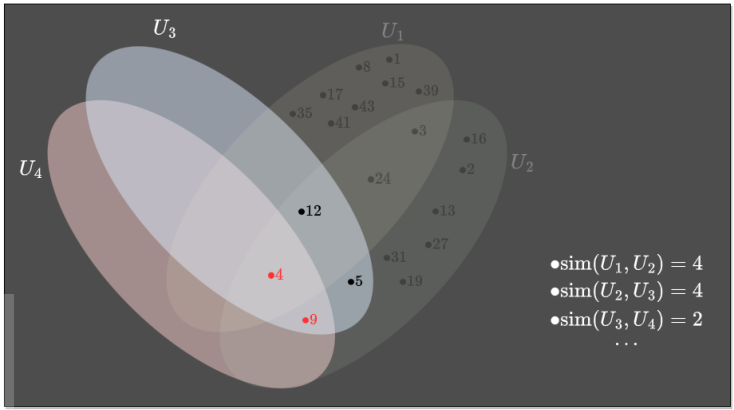
$$
\begin{aligned}
U_1 &= \{1, 3, 4, 8, 12, 15, 17, 24, 35, 39, 41, 43\}, \\
U_2 &= \{2, 3, 4, 5, 9, 12, 13, 16, 19, 24, 17, 31\}, \\
U_3 &= \{4, 5, 9, 12\}, \text{ and} \\
U_4 &= \{4, 9\}.
\end{aligned}
$$

- Naively, we could define $\text{sim}(i, l) = |U_i \cap U_l|$.

Intersection as Similarity

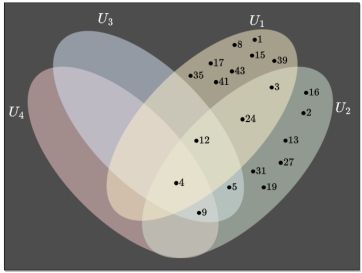


Intersection as Similarity



Is it reasonable to say $\text{sim}(1,2) > \text{sim}(3,4)$?

Intersection as Similarity



In pairs, discuss with *a colleague* whether, in **terms of recommender systems**, it is reasonable to compute the similarities as shown in the previous slides:

- $\text{sim}(1, 2) = 4$
- $\text{sim}(2, 3) = 4$
- $\text{sim}(3, 4) = 2$

- Note that users 1 and 2 are highly motivated users who have interacted with many different items.
- Such types of users tend to bias the similarity function by making them similar to every user.

The Jaccard Similarity

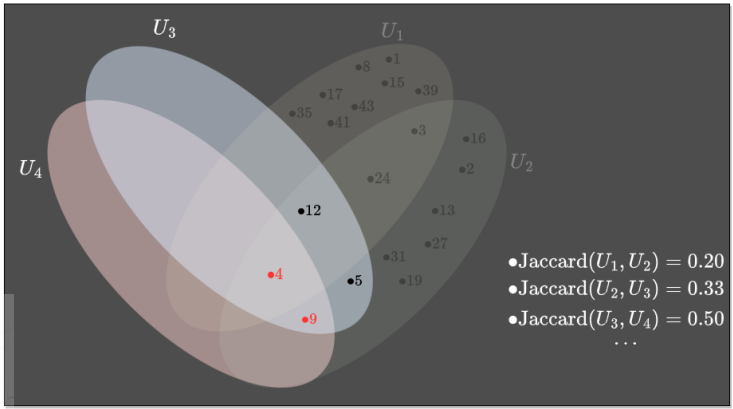
- Our first attempt at correcting this issue would be by normalizing the similarity by the users' popularity.
- The Jaccard similarity does that by dividing the union by the intersection:

$$\text{Jaccard}(i, l) = \frac{|U_i \cap U_l|}{|U_i \cup U_l|}$$

- The Jaccard similarity is only defined for non-empty sets.

Note that $\text{Jaccard}(i, l) \in [0, 1]$. $\text{Jaccard}(i, l) = 0$ if i and l have not interacted with any items in common. On the other hand, $\text{Jaccard}(i, l) = 1$ if all interactions of both users are the same.

The Jaccard Similarity



Is it more reasonable now?

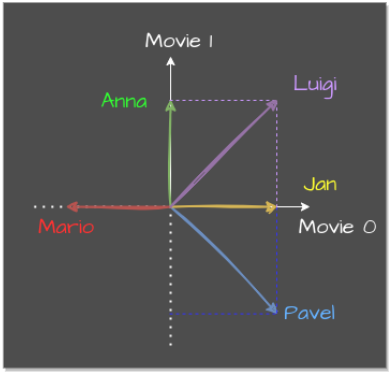
Cosine Similarity

- The Jaccard similarity has a limitation: it is only defined when interactions are represented as sets.
- It would not be defined, for example, when feedback is associated.
- Suppose we have the following recommender system with the following possibilities:
  - Positive feedback (e.g., like): $+1$;
  - Negative feedback (e.g., dislike): -1;
  - Unobserved feedback: 0.
- Cosine similarity measures the angle extension between the vector of interactions and can be computed as:

$$\cos(\theta) = \frac{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}{||\boldsymbol{a}|| \cdot ||\boldsymbol{b}||}$$

Cosine Similarity </>

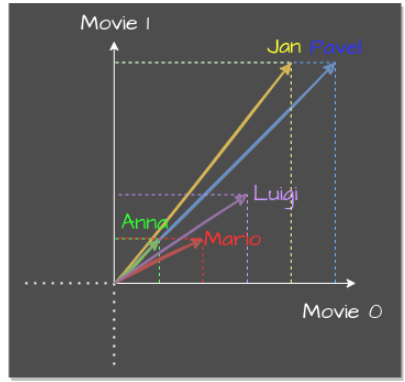
	Movie ID 0	1
Anna	0	1
Pavel	1	-1
Jan	1	0
Mario	-1	0
Luigi	1	1



$$\cos(\theta) = \frac{\langle a\ ,\ b \rangle}{||a||||b||}$$

Cosine Similarity </>

Movie ID	0	1
Anna	1	1
Pavel	5	5
Jan	4	5
Mario	2	1
Luigi	3	2



$$\cos(\theta) = \frac{\langle a\ ,\ b \rangle}{||a||||b||}$$

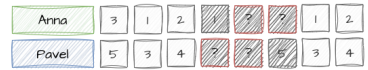
Euclidean Similarity

- In some cases, we would like to measure similarity based on the distances between vectors.
- However, the Euclidean distance, for example, is larger for vectors that are distant and smaller for close (or similar) vectors.
- We can define the **Euclidean Similarity** as follows:

$$\text{ES}(a, b) = \frac{1}{1 + \text{d}(a, b)},$$

  where $\text{d}(a, b)$ is the Euclidean distance between vectors a and b.
- Note that $\text{ES}(a, b) = 1$ when $a = b$ and approaches zero as a becomes initially distant from b.
- In fact, we can define this score for any defined distance measure.

Explicit Feedback



Haters vs. Lovers

- Some users tend to be more rigorous in their ratings, while others are generous.
- Let's take a look at our previous example:

- One way to mitigate this bias is by subtracting the average rating of each user and then performing k-NN.
- In the prediction step, the user's average rating is added back.

k-NN for Recommendation

- ✓ k-NN is considered a memory-based method because no training is needed. Instead, predictions are made directly from data.
- ✓ It is simple to implement and simple to explain.
- ✓ The method is relatively stable with the addition of interactions.
- × The method can have limited coverage due to sparsity.
- × The method can be impractical for large datasets
  - Clustering and dimensionality reduction are frequently used to deal with scalability.

Obrigado :) - Faculty of Information Technology