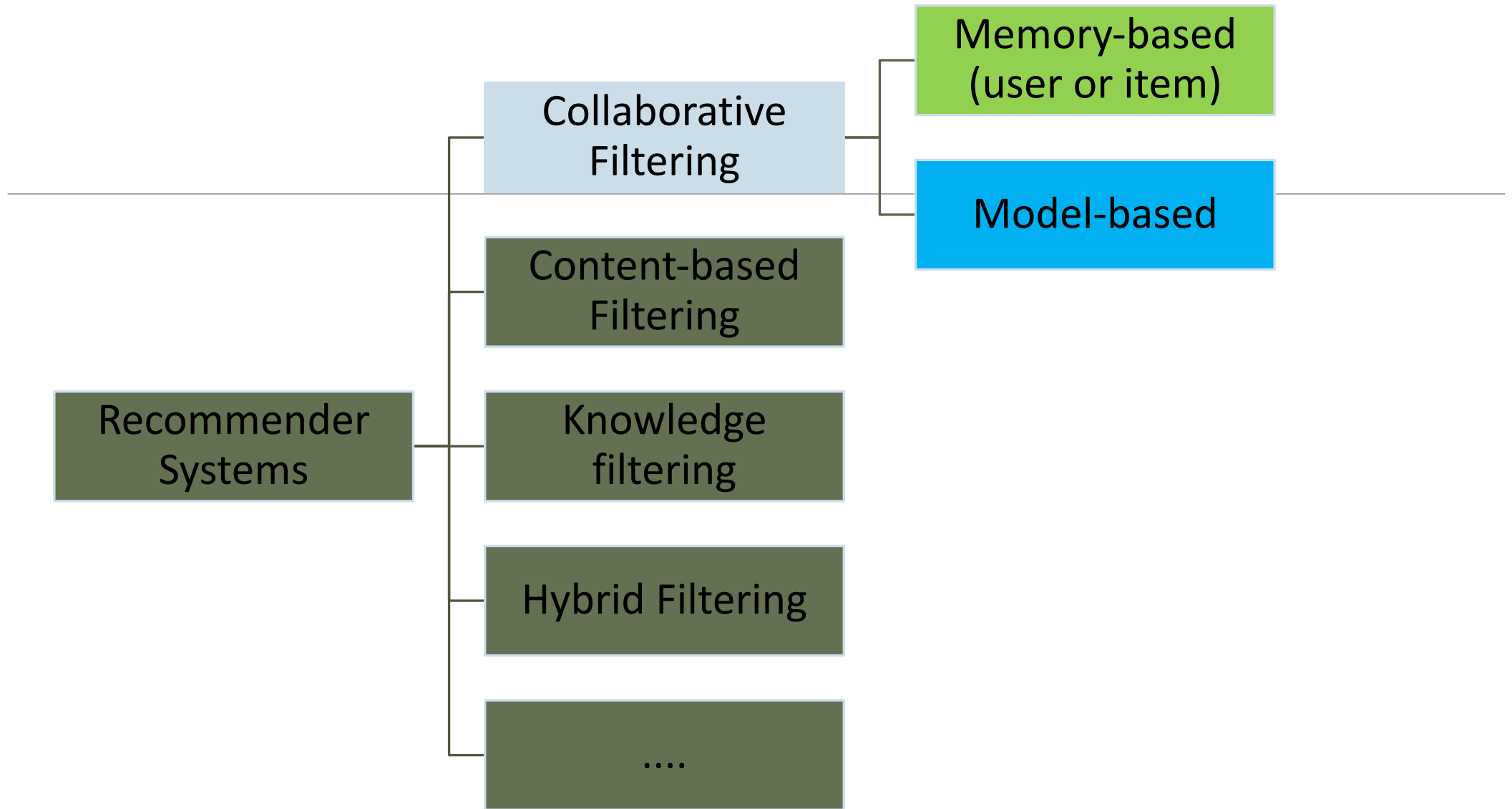# Collaborative Filtering

MEI

CONSTANTINO MARTINS, CATARINA FIGUEIREDO, DULCE MOTA AND JOAQUIM SANTOS

Disclaimer

- **Some of this material/slides are adapted from several:**
  - Presentations found on the internet;
  - Papers
  - Books;
  - Web sites
  - …

Collaborative Filtering (CF)

The most prominent approach to generate recommendations:
- ❑ Used by large, commercial e-commerce sites
- ❑ Well-understood, various algorithms and variations exist
- ❑ Applicable in many domains (music, movies, ..)

Basic assumption and idea:
- ❑ Users give ratings to items (implicitly or explicitly)
- ❑ Customers who had similar interests/preferences/tastes in the past, will have similar interests/preferences/tastes in the future

Approach:
- ❑ Use the "wisdom of the crowd" to recommend items
- ❑ Key characteristic of CF: predicts the utility of items for a user based on the items previously rated by users with similar interests or preferences
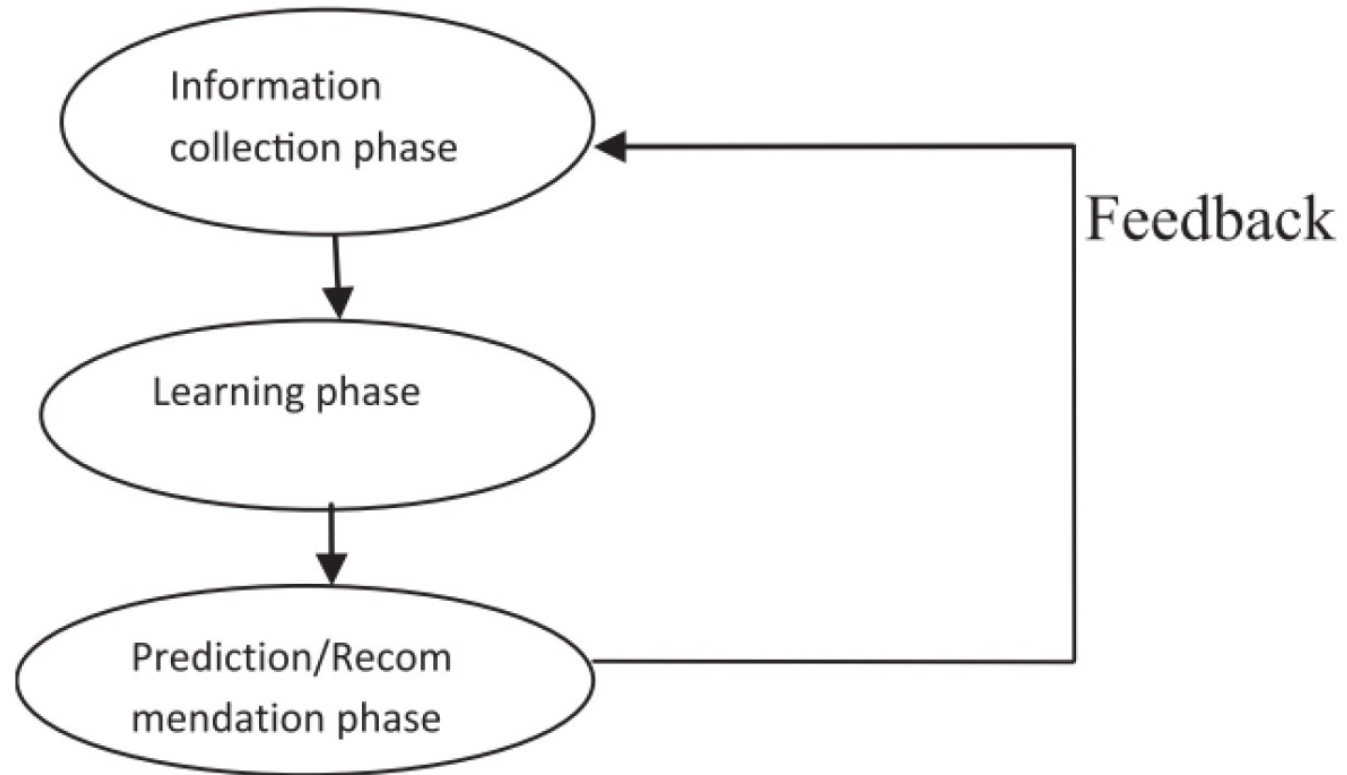
Concepts:
User: Any individual who can provides ratings to a system
Items: Anything for which a human can provide a rating

Some of the techniques will be discussed in more detail in the TP
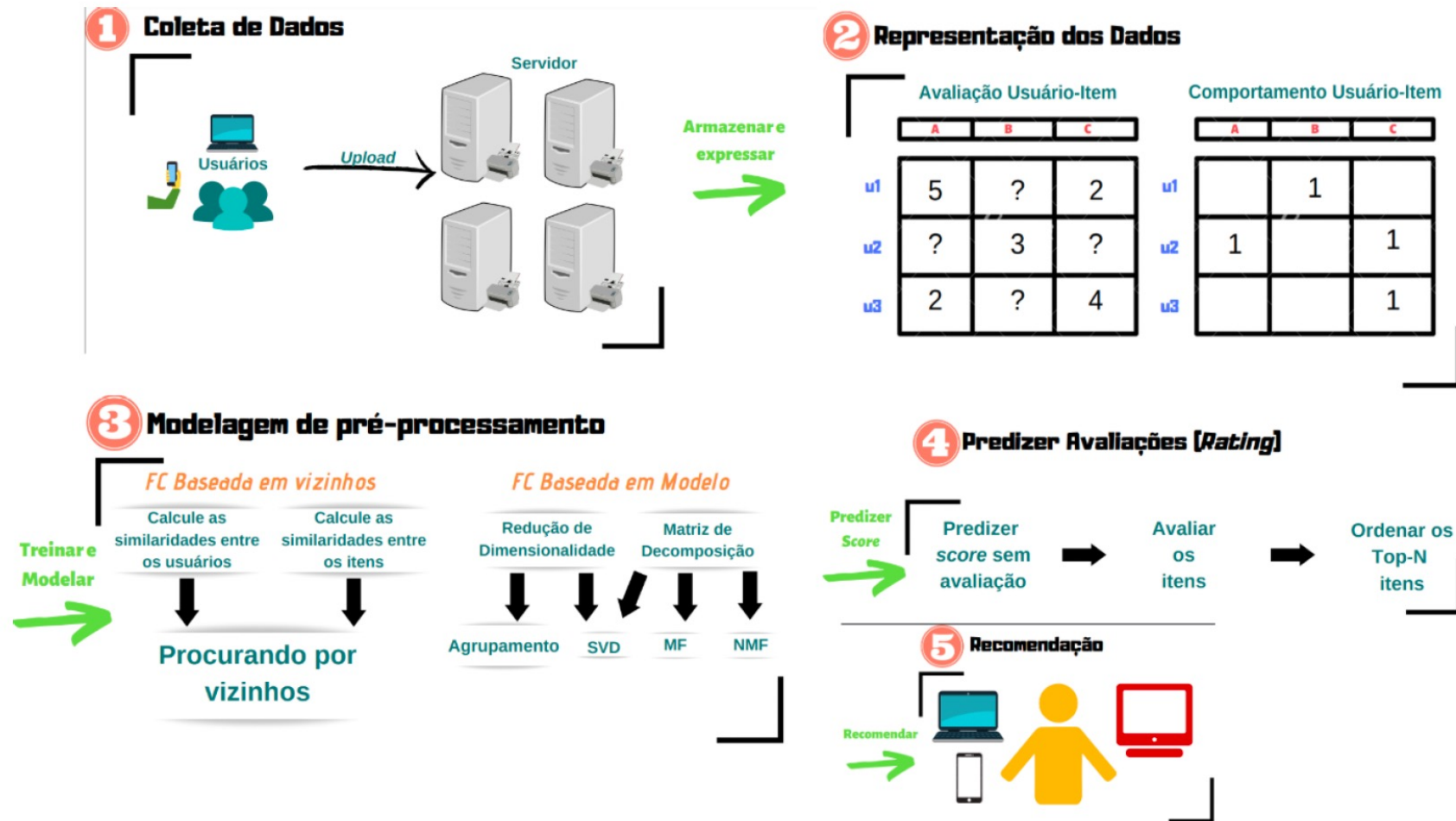
Recommendation Phases

Figura 1. *Framework* de Sistemas de Recomendação baseados em Filtragem Colaborativa - Adaptada de [Chen et al. 2018]

Example of a common scenario

The most common scenario is the following:

❑ A set of users has initially rated some subset of music (e.g., on the scale of 1 to 5) that they have already seen

❑ These ratings serve as the input

❑ The recommendation system uses these known ratings to predict the ratings that each user would give to those not rated movies by him/her

❑ Recommendations of music are then made to each user based on the predicted ratings

The Recommendation Problem

❑ We have a set of users U and a set of items S to be recommended to the users

❑ Let p be an utility function that measures the usefulness of item s ($\in S$) to user u ($\in U$), i.e.,

◦ $p:U×S \to R$, where R is a totally ordered set (e.g., non-negative integers or real numbers in a range)

❑ Objective

◦ Learn p based on the past data

◦ Use p to predict the utility value of each item s ($\in S$) to each user u ($\in U$)

Different variations

❑In some applications, there is no rating information while in some others there are also additional attributes

- ◦ About each user (e.g., age, gender, income, marital status, etc), and/or
- ◦ About each music (e.g., title, genre, compositor, etc)

❑When no rating information, the system will not predict ratings but predict the likelihood that a user will enjoy watching a movie

Types of collaborative filtering techniques

Memory based:

❑ Calculation of user or item similarity

❑ User-item filtering "Users who liked this item also liked …"

❑ Item filtering "Users who are similar to you also liked …"

❑ The **rating matrix** is directly used to find neighbors / make predictions

❑ Does not scale for most real-world scenarios

❑ Large e-commerce sites have millions of customers and millions of items

Types of collaborative filtering techniques

Model based:

❑Based on an offline pre-processing or "model-learning" phase

❑At run-time, only the learned model is used to make predictions

❑Models are updated / re-trained periodically

❑Large variety of techniques used

❑Model-building and updating can be computationally expensive

❑Item-based CF is an example for model-based approaches

User feedback

Explicit feedback: Users indicate their preference for items:

❑Ratings scales (1-5, 1-10, for example using stars) ★☆☆☆☆   ★★★☆★☆

❑Binary values (liked/disliked, Agree/Disagree, Good/Bad, etc)

❑Unary ratings (Good, Purchase, etc).

❑Others?

Implicit feedback: Users interact with items:

❑Purchase history

❑Viewing duration

❑Click patterns

❑Others?

User feedback

Explicit vs. Implicit ratings

❑ Explicit ratings
  ◦ Users rate themselves for an item
  ◦ Most accurate descriptions of a user's preference
  ◦ Challenging in collecting data

❑ Implicit ratings
  ◦ Observations of user behavior
  ◦ Can be collected with little or no cost to user
  ◦ Ratings inference may be imprecise

Practical Issues : Cold Start

New user
- ❑ Rate some initial items
- ❑ Non-personalized recommendations
- ❑ Describe tastes
- ❑ Demographic info.

New Item
- ❑ Non-CF: content analysis, metadata
- ❑ Randomly selecting items

New Community
- ❑ Provide rating incentives to subset of community
- ❑ Initially generate non-CF recommendation
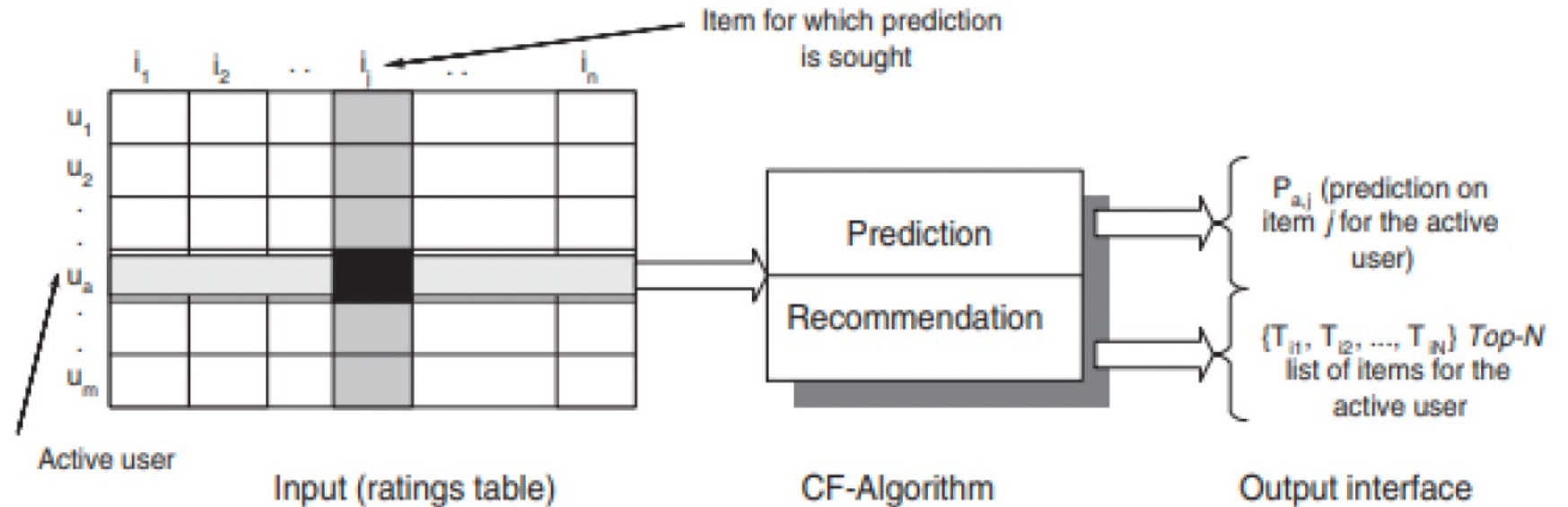- ❑ Start with other set of ratings from another source outside community

Pure CF Approaches

Input: Only a matrix of given user–item ratings
Output types:
A (numerical) prediction indicating to what degree the current user will like or dislike a certain item
A top-N list of recommended items

Ratings/Utility Matrix

Infer user preferences about items

Produce **highly relevant** & **personalized** lists of items

Items (e.g., movies, music, products)

	i_1	...	i_k	...	i_m
u_1	★★★★★		★★★★★		★☆☆☆☆
⋮					
u_j	★★★☆☆		?		★★☆☆☆
⋮					
u_n	★★☆☆☆		★★★★☆		★★☆☆☆

Users

The non-empty entries correspond to: Items the user has clicked, liked etc
If the matrix was dense, we wouldn't be able to make recommendations
It means every user has already seen every item, so we have nothing new to recommend
The matrix must be sparse in order to actually have items to recommend

Sparse matrix: Most values are unknown

Predicting: The task of filling the unknown values

Memory based

User-based collaborative filtering (UBCF)

Assumption: Similar users have similar preferences

☐ **User similarity: agreement on co-rated items**

☐ Prediction: **weighted sum of similar user's ratings**

Item-based collaborative filtering (IBCF)

Assumption: Users have similar tastes for similar items

☐ **Item similarity: agreement within users rated both items**

☐ Prediction: **weighted sum of similar items' ratings**

Similarity measures

❑The similarity measure allows quantifying the resemblance between users or items

❑The choice of similarity measure has a direct impact on the quality of the recommendation system

❑The more data used to compare users' opinions, the more reliable the calculation of similarity between them will be

❑The two most popular measures for calculating similarity between users/items are based on the **Pearson correlation coefficient** and **cosine similarity**

❑Pearson correlation coefficient is typically used when user ratings are available on a numerical scale from 1 to 5 (example: Netflix, MovieLens, and Amazon.com).

❑Cosine similarity method is more commonly used for similarity calculations when it's a binary evaluation of 0 or 1, corresponding to liked or disliked.

❑Another measure is the Euclidean distance, which measures the distance between two points in two-dimensional or multidimensional space.

Pearson Correlation Coefficient

❑The Pearson Correlation Coefficient is used to measure how much two variables are linearly related to each other. The correlation value ranges between -1 and 1. Within a Recommender System (RS), this value indicates how much the active user u_a matches each of the other users in the ratings they have made in common

Similarity between user u and their neighborhood v

$$sim(u, v) = \frac{\sum_{i \in I_u \cap I_u}(r_{u,i} - \bar{r}_u) \times (r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_u}(r_{u,i} - \bar{r}_u)^2} \times \sqrt{\sum_{i \in I_u \cap I_u}(r_{v,i} - \bar{r}_u)^2}}$$

The variable $\bar{r}_u$ represents the mean of user u's ratings, $\bar{r}_v$ represents the mean of user v's ratings. $r_{u,i}$ corresponds to the rating given by user u to item i. $r_{v,i}$ corresponds to the rating given by user v to item i

Pearson Correlation Coefficient

❑The Pearson correlation coefficient ranges from -1 to 1, where:

◦ 1 indicates a perfect positive correlation (when one variable increases, the other also increases proportionally)

◦ -1 indicates a perfect negative correlation (when one variable increases, the other decreases proportionally).

◦ 0 indicates that there is no linear correlation between the two variables.

❑Example, one value of 0.4 suggests that the variables are positively related, but the relationship is not as strong as a perfect correlation

A simple example

$$sim(u,v) = \frac{\sum_{i \in I_u \cap I_u}(r_{u,i} - \bar{r}_u) \times (r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_u}(r_{u,i} - \bar{r}_u)^2} \times \sqrt{\sum_{i \in I_u \cap I_u}(r_{v,i} - \bar{r}_u)^2}}$$

User/item	Item 1	Item 2	Item 3
User X	2	4	3
User Y		1	2
User Z		1	3
User V			

Mean of user X, $\bar{r}_X=(2+4+3)/3=3$

Mean of user Y, $\bar{r}_y=(1+2)/2=1.5$

Simliraty beetwen X and Usey Y

$$sim(X,Y) = \frac{(4-3)*(1-1,5)+(3-3)*(2-1,5)}{\sqrt{((4-3)^2+(3-3)^2)} * \sqrt{((1-1,5)^2+(2-1,5)^2)}} = \frac{-0,5}{0,7}$$

$$= -0,71$$

Cosine Similarity

❑Users or items are represented by vectors to calculate similarity

❑Each user vector contains the ratings corresponding to the items they have rated

❑In the case of items, each vector represents the set of users who have rated that item

❑**The elements of the vector are greater than zero to express a rating already given and zero for ratings not yet given**

❑The cosine similarity measure between the ratings of the active user and the other users is represented by the measure of the angle between them

❑The cosine similarity is always a non-negative number since the ratings are always non-negative numbers

❑**The similarity measure varies between 0 (weak correlation) and 1 (strong correlation)**

$$sim(u, v) = \frac{\sum_i r_{u,i} \times r_{v,i}}{\sqrt{\sum_i (r_{u,i})^2} \times \sqrt{\sum_i (r_{v,i})^2}}$$

Euclidean Distance

❑The Euclidean distance measures the distance between two points in two-dimensional or multidimensional space

❑The closer the points are in the reference space, the more similar they will be

❑In the distance between two vectors representing user profiles, the smaller the distance from one vector to another, the more similar the users are. The value of the distance varies from 0, indicating higher similarity, to 1, indicating lower similarity

$W_{u,v}$ represents the distance between the active user u and another user v, u_i is the rating that user u gave for item i, v_i is the rating of the other user for the same item

$$W_{u,v} = \sqrt{\sum_i (u_i - v_i)^2}$$

Can be applied to items?

Simple Collaborative Filtering

❑The fundamental idea is that typically items that are more popular "by critics" will have a higher probability of being appreciated by the majority of the public

❑Solve the Cold start problem?

❑For example, in the case of movie ratings, the equation used by IMDb for defining the "Top Rated 250 titles"

❑This formula provides a true "Bayesian" estimate, which takes into consideration the number of votes each title has received, the minimum votes required to be on the list, and the average votes for all titles

$$WR = \left(\frac{v}{v+m} \times R \right) + \left(\frac{m}{v+m} \times C \right)$$

- v is the number of votes for the movie (votes)
- m is the minimum number of votes to be considered in the "Top Rated" list
- R is the average rating of the movie (rating)
- C is the mean of all average ratings

Simple Collaborative Filtering

❑The choice of the value of m implies the removal of items that have a number of ratings lower than a certain **threshold m**

❑Currently, in the case of the "IMDb Top Rated 250 titles", the value of m is 25,000 (only titles with more than 25,000 votes are considered)

❑With the calculation of the popularity of all items, a Top-N recommendation is obtained. This method is not sensitive to the interests and tastes of specific users

User based Collaborative Filtering

❑ Memory-based algorithms predict the rating of a user u_a for an item by calculating an average of ratings from similar users who share the same interests as u_a

❑ The process is divided into two phases:
1. Calculating the similarity between users
2. predicting ratings for items that have not yet been rated

❑ The already rated items allow selecting a set of k "neighbors" of the active user u_a, with higher similarity values compared to this user

❑ Therefore, the items preferred by the neighbors are recommended to u_a

❑ If a Top-N recommendation is desired, predictions can be calculated for the items not yet rated, presenting the items with the N highest predicted ratings

User-based nearest-neighbor collaborative filtering

Example
- A database of ratings of the current user, Constantino, and some other users is given:

	Item1	Item2	Item3	Item4	Item5
Constantino	5	3	4	4	?
Catarina	3	1	2	3	3
Dulce	4	3	4	3	5
Joaquim	3	3	1	5	4
Schmidt	1	5	5	2	1

- Constantino has not yet rated or seen *Item5*
- Determine whether Constantino will like or dislike *Item5*

User-based nearest-neighbor collaborative filtering

Some first questions

- ◦ How do we measure similarity?
- ◦ How many neighbors should we consider?
- ◦ How do we generate a prediction from the neighbors' ratings?

	Item1	Item2	Item3	Item4	Item5
Constantino	5	3	4	4	?
Catarina	3	1	2	3	3
Dulce	4	3	4	3	5
Joaquim	3	3	1	5	4
Schmidt	1	5	5	2	1

Measuring user similarity

A popular similarity measure in user-based CF: **Pearson correlation**

a, b : users

$r_{a,p}$: rating of user a for item p

P : set of items, rated both by a and b

◦ Possible similarity values between -1 and 1

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

Measuring user similarity

a, b : users

$r_{a,p}$: rating of user a for item p

	Item1	Item2	Item3	Item4	Item5
Constantino	5	3	4	4	?
Catarina	3	1	2	3	3
Dulce	4	3	4	3	5
Joaquim	3	3	1	5	4
Schmidt	1	5	5	2	1

sim = 0,85

sim = 0,00

sim = 0,70

sim = -0,79

Making predictions

❑After determining the similarity between users and the subset of neighboring users, it is necessary to aggregate the rating information to generate the prediction value. The prediction is made using a weighted average of the ratings of neighboring users on item

$$P_{u,i} = \overline{r_u} + \frac{\sum_{v \in V} sim(r_u, r_v) \times (r_{v,i} - \overline{r_v})}{\sum_{v \in V} |sim(r_u, r_v)|}$$

Equation allows to calculate the predicted rating, where i corresponds to the item for which the prediction is desired for user u (who has not yet rated this item), with $r_{v,i}$ being the rating assigned by user v to item i and V representing the neighborhood of user u

Subtracting the rating from the user's mean ($\overline{r}v$) helps to compensate for differences between users in the use of the rating scale (some users tend to give higher ratings than others)
It remains to determine the size of the neighborhood to consider

Improving the metrics / prediction function

Not all neighbor ratings might be equally "valuable"
- Agreement on commonly liked items is not so informative as agreement on controversial items
- **Possible solution**: Give more weight to items that have a higher variance

Value of number of co-rated items
- Use "significance weighting", by e.g., linearly reducing the weight when the number of co-rated items is low

Case amplification
- Intuition: Give more weight to "very similar" neighbors, i.e., where the similarity value is close to 1.

Neighborhood selection
- Use similarity threshold or fixed number of neighbors

Item based Collaborative Filtering

❑ Item-based Collaborative Filtering considers the set of items that the active user u_a has previously rated and determines how similar they are to a target item

❑ In this technique, there is a need to identify similar users each time a recommendation is requested, resulting in faster recommendations

❑ The algorithm first evaluates the K most similar items for each item according to their similarities, identifying the set I = {i_1, i_2, ..., i_n} of candidate recommendation items

❑ Subsequently, it takes the union among the K most similar items, removing each item in the item set $I_{u\ i}$ that the user has already rated; then it calculates the similarities between each item in the set I and the set $I_{u\ i}$. The resulting set of items in I is sorted in descending order of similarity and recommended as a Top-N list

❑ To calculate the similarity between two items i_1 and i_2, it is necessary to identify users who have rated these items and calculate the similarity between them

❑ After finding the cluster of similar items, the next step is to analyze user ratings and choose a technique to generate predictions of items of their interest

Item-based collaborative filtering

Basic idea:

◦ Use the similarity between items (and not users) to make predictions

Example:

◦ Look for items that are similar to Item5

◦ Take Alice's ratings for these items to predict the rating for Item5

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

The cosine similarity measure

Produces better results in item-to-item filtering

Ratings are seen as vector in n-dimensional space

Similarity is calculated based on the angle between the vectors

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

Adjusted cosine similarity
- take average user ratings into account, transform the original ratings
- U: set of users who have rated both items a and b

$$sim(\vec{a}, \vec{b}) = \frac{\sum_{u \in U}(r_{u,a} - \overline{r_u})(r_{u,b} - \overline{r_u})}{\sqrt{\sum_{u \in U}(r_{u,a} - \overline{r_u})^2} \sqrt{\sum_{u \in U}(r_{u,b} - \overline{r_u})^2}}$$

Making predictions

A common prediction function:

$$pred(u, p) = \frac{\sum_{i \in ratedItem(u)} sim(i, p) * r_{u,i}}{\sum_{i \in ratedItem(u)} sim(i, p)}$$

❑Neighborhood size is typically also limited to a specific size

❑Not all neighbors are taken into account for the prediction

❑An analysis of the MovieLens dataset indicates that "in most real-world situations, a neighborhood of 20 to 50 neighbors seems reasonable"

Pre-processing for item-based filtering

❑Item-based filtering does not solve the scalability problem itself

❑Pre-processing approach by Amazon.com
- Calculate all pair-wise item similarities in advance
- The neighborhood to be used at run-time is typically rather small, because only items are taken into account which the user has rated
- Item similarities are supposed to be more stable than user similarities

Memory requirements
- Up to N^2 pair-wise similarities to be memorized (N = number of items) in theory
- In practice, this is significantly lower (items with no co-ratings)
- Further reductions possible
  - Minimum threshold for co-ratings
  - Limit the neighborhood size (might affect recommendation accuracy)

Model-based approaches

Different techniques proposed in the last years:

❑ Matrix factorization techniques, statistics
   ◦ Singular Value Decomposition (SVD), principal component analysis
❑ K-NN (K-Nearest Neighbors)
❑ Association rule mining
   ◦ compare: shopping basket analysis
❑ Probabilistic models
   ◦ clustering models, Bayesian networks, probabilistic Latent Semantic Analysis
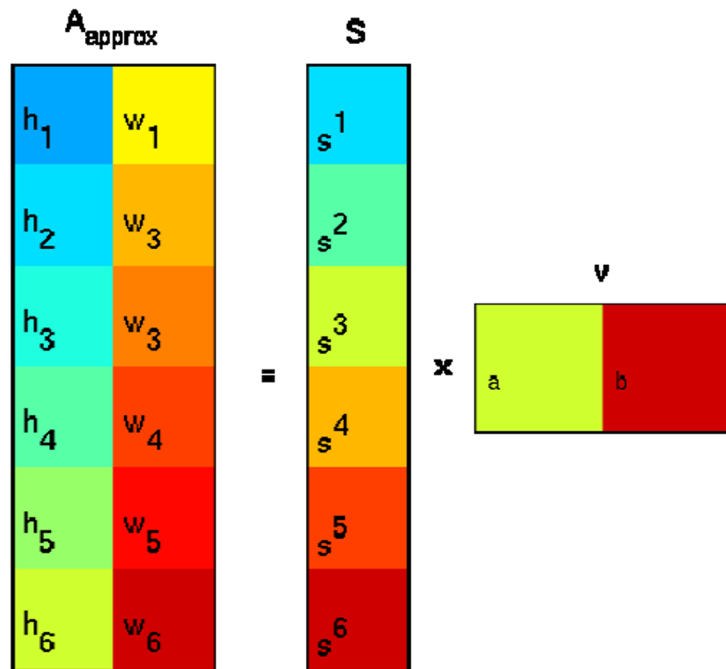❑ Various other machine learning approaches

Costs of pre-processing

❑ Usually not discussed
❑ Incremental updates possible?

Some of the techniques will be discussed in more detail in the TP

Model based Collaborative Filtering
Matrix factorization

Factorization

- In the Recommendation Systems field, SVD models users and items as vectors of latent features which when cross product produce the rating for the user of the item

- With SVD a matrix is factored into a series of linear approximations that expose the underlying structure of the matrix.

- The goal is to uncover latent features that explain observed ratings

Matrix factorization

❑Informally, the SVD theorem (Golub and Kahan 1965) states that a given matrix M can be decomposed into a product of three matrices as follows

$$M = U \times \Sigma \times V^T$$

◦ where U and V are called *left* and *right singular vectors* and the values of the diagonal of Σ are called the *singular values*

❑We can approximate the full matrix by observing only the most important features – those with the largest singular values

Example for SVD-based recommendation

- SVD: $$M_k = U_k \times \Sigma_k \times V_k^T$$

U_k	Dim1	Dim2
Alice	0.47	-0.30
Bob	-0.44	0.23
Mary	0.70	-0.06
Sue	0.31	0.93

V_k^T	Terminator	Die Hard	Twins	Eat Pray Love	Pretty Woman
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

Σ_k	Dim1	Dim2
Dim1	5.63	0
Dim2	0	3.23

- Prediction: $\hat{r}_{ui} = \bar{r}_u + U_k(Alice) \times \Sigma_k \times V_k^T(EPL)$

  $= 3 + 0.84 = 3.84$

The projection of U and V^T in the 2 dimensional space (U_2, V_2^T)

CF Matrix factorization

❑ Matrix factorization has gained popularity for CF in recent years due to its superior performance both in terms of recommendation quality and scalability.

❑ Part of its success is due to the **Netflix Prize contest** for movie recommendation, which popularized a Singular Value Decomposition (SVD) based matrix factorization algorithm.

◦ The prize winning method of the Netflix Prize Contest employed an adapted version of SVD

A possible architecture for the project?
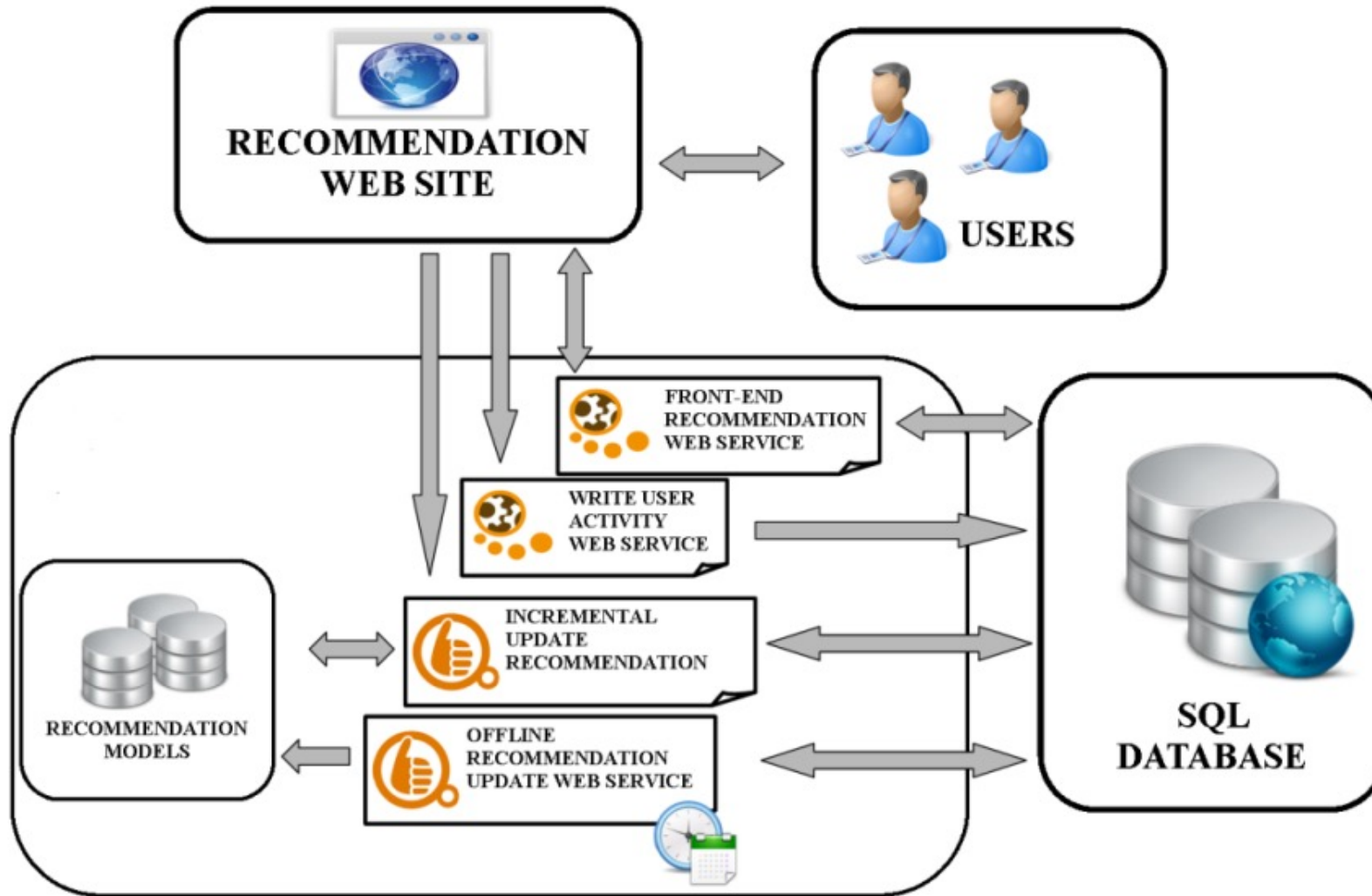
Some notes

❑Pearson coefficient, cosine similarity, and Euclidean distance (or other techniques) can be adapted by assigning or utilizing weights

❑Introducing weights can allow for a more flexible approach, where certain features or attributes have more influence on the similarity measure than others

❑Weights can be assigned based on the relative importance of the features or through optimization techniques to adjust the weights automatically during the similarity calculation process

❑Weights can be for example determined based on domain knowledge, data analysis, or through optimization methods to improve the performance of the recommendation system

❑Adjusting the weights assigned to different features, we can customize the recommendation process to better match the user preferences

❑More accurate and personalized recommendations

IBCF

Compute similarity between items

set of users rated both items   rating on item *i*   rating on item j

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_u)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_u)^2}}$$

deviation from the average rating

Prediction of rating for item *i*
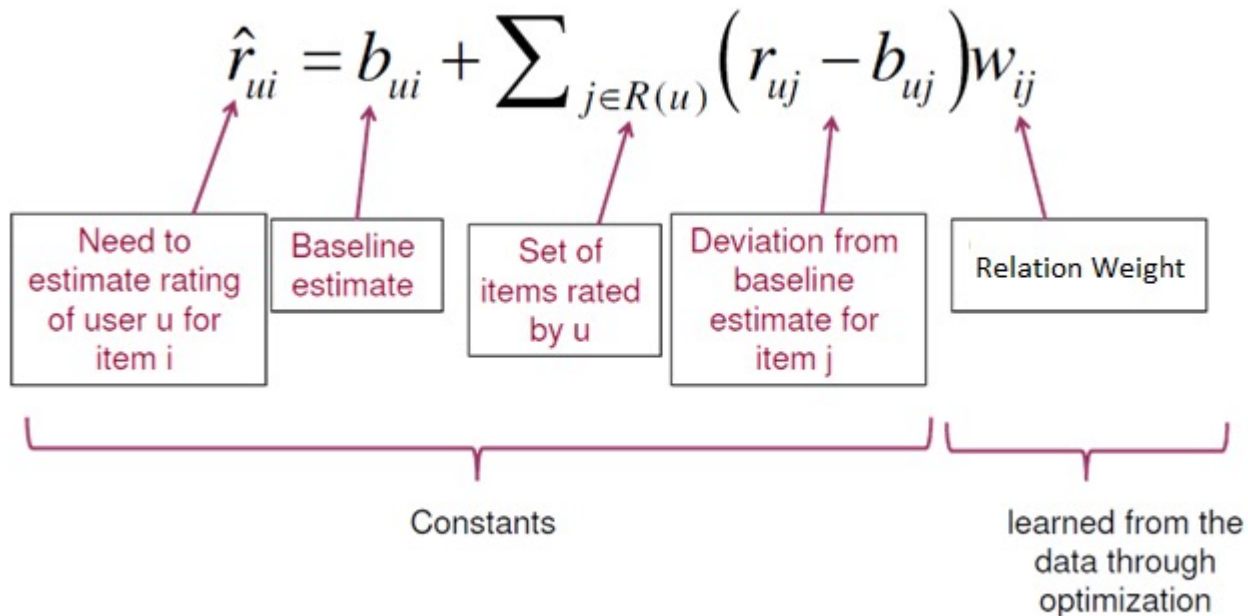
most similar items to *i*   similarity as weight   rating on similar item

$$p_{u,i} = \frac{\sum_{j \in S} sim(i,j)R_{u,j}}{\sum_{j \in S}|sim(i,j)|}$$

Other basic example - Nearest Neighbor
Using optimization

$$b_{ui} = \mu + b_u + b_i$$

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in R(u)} \left(r_{uj} - b_{uj} \right) w_{ij}$$

Need to estimate rating of user u for item i	Baseline estimate	Set of items rated by u	Deviation from baseline estimate for item j	Relation Weight

Constants

learned from the data through optimization

$$\min \sum (r_{ui} - \hat{r}_{ui})^2$$

References

Serão colocadas depois da entrega do primeiro trabalho