Recommender Systems

Professor Vwani Roychowdhury

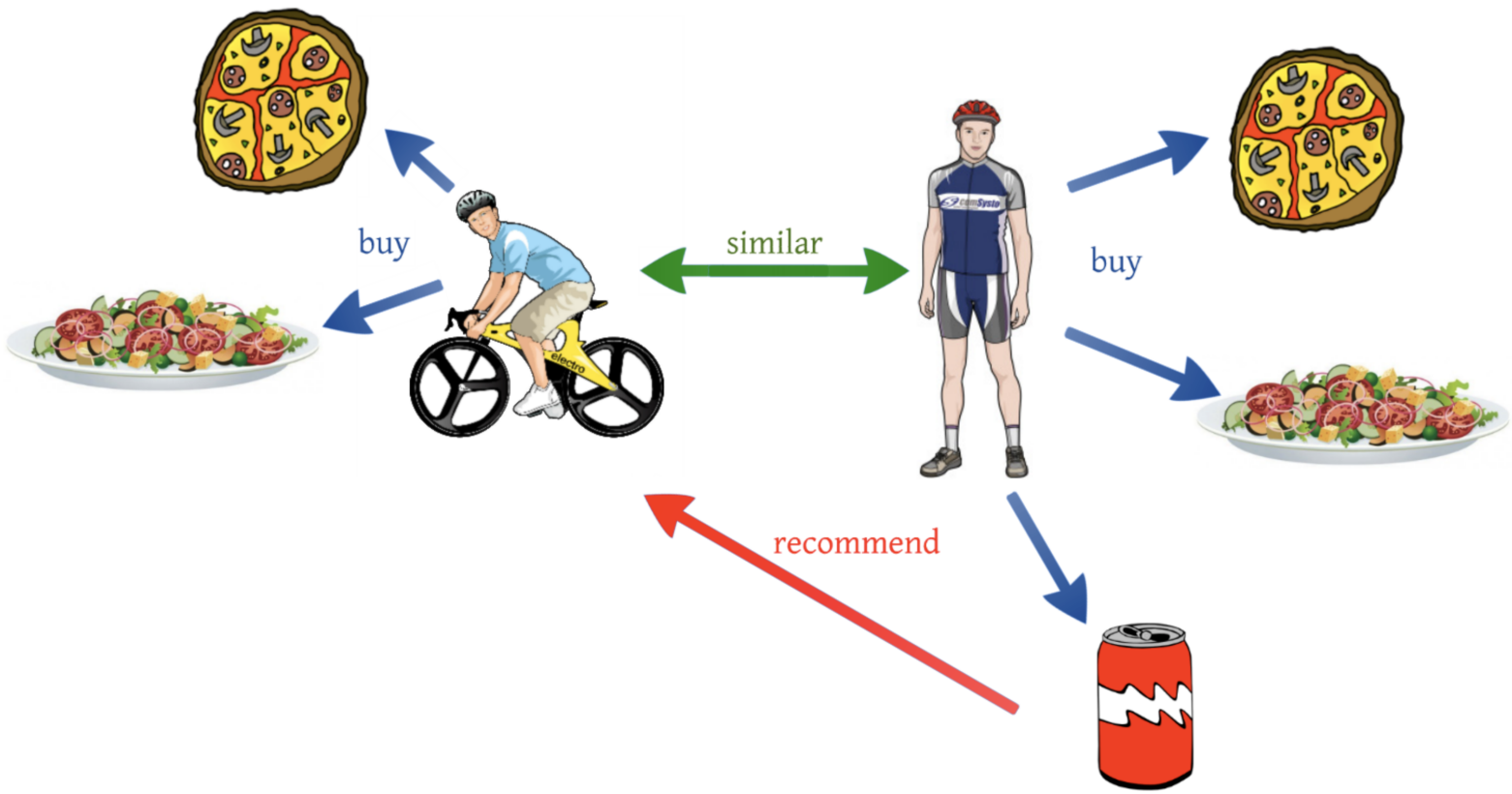
University of California, Los Angeles

January 30, 2019

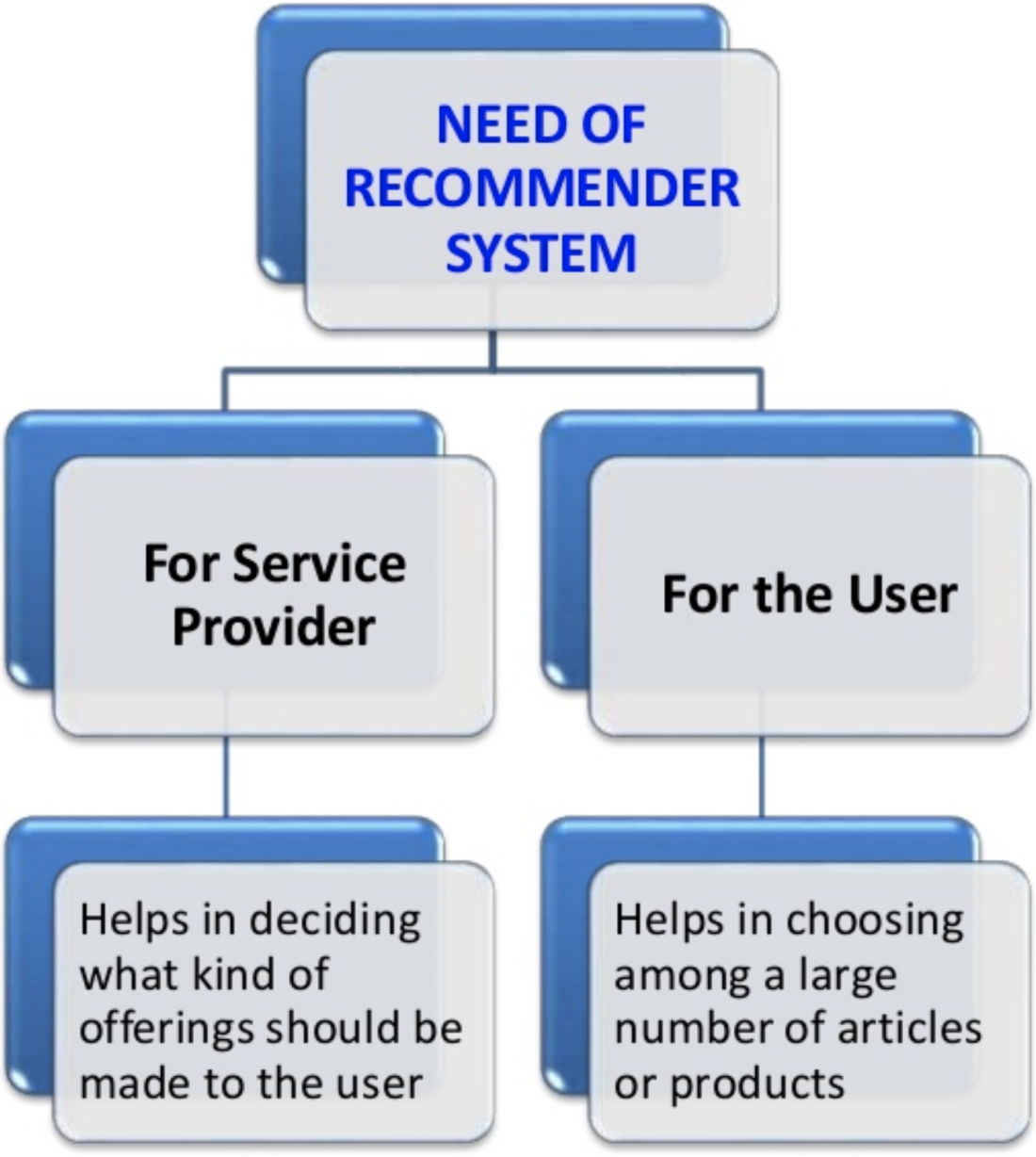
Overview

What is a Recommender System?

Wikipedia: Recommender systems are a subclass of information filtering system that seek to predict the rating that user would give to an item they had not yet considered, using a model built from the users' characteristics of the social environment.



Why a Recommender System?



Examples of recommender system

System	Product Recommended
Amazon	Books and other products
Netflix	DVDs, Streaming Video
Jester	Jokes
GroupLens	News
MovieLens	Movies
Facebook	Friends, Advertisements
Pandora	Music
Tripadvisor	Travel products

Approaches to recommendation

- Collaborative filtering: Recommend items based only on the users' past behavior
  - **User-based:** Find similar users to the target user and recommend what they liked
  - **Item-based:** Find similar items to those that the target user have previously liked
  - **Model-based:** Build a model of the user using machine learning algorithms
- Content-based: Recommend based on item features
- Ranking: Treat recommendation as a ranking problem
- Demographic: Recommend based on user features
- Hybrid: Combination of any of the approaches given above

Ingredients of collaborative filtering

- A list of m Users and n Items
- Each user has a list of observed items with associated feedback
  - Explicit feedback - Rating
  - Implicit feedback - Purchase records
- Target user for whom the collaborative filter prediction task is performed
- Method for predicting a rating for items not currently rated by the target user
  - Neighborhood-based methods
  - Latent factor-based methods
  - Decision trees
  - Rule-based methods
  - Bayesian methods
  - ...

User-based collaborative filtering

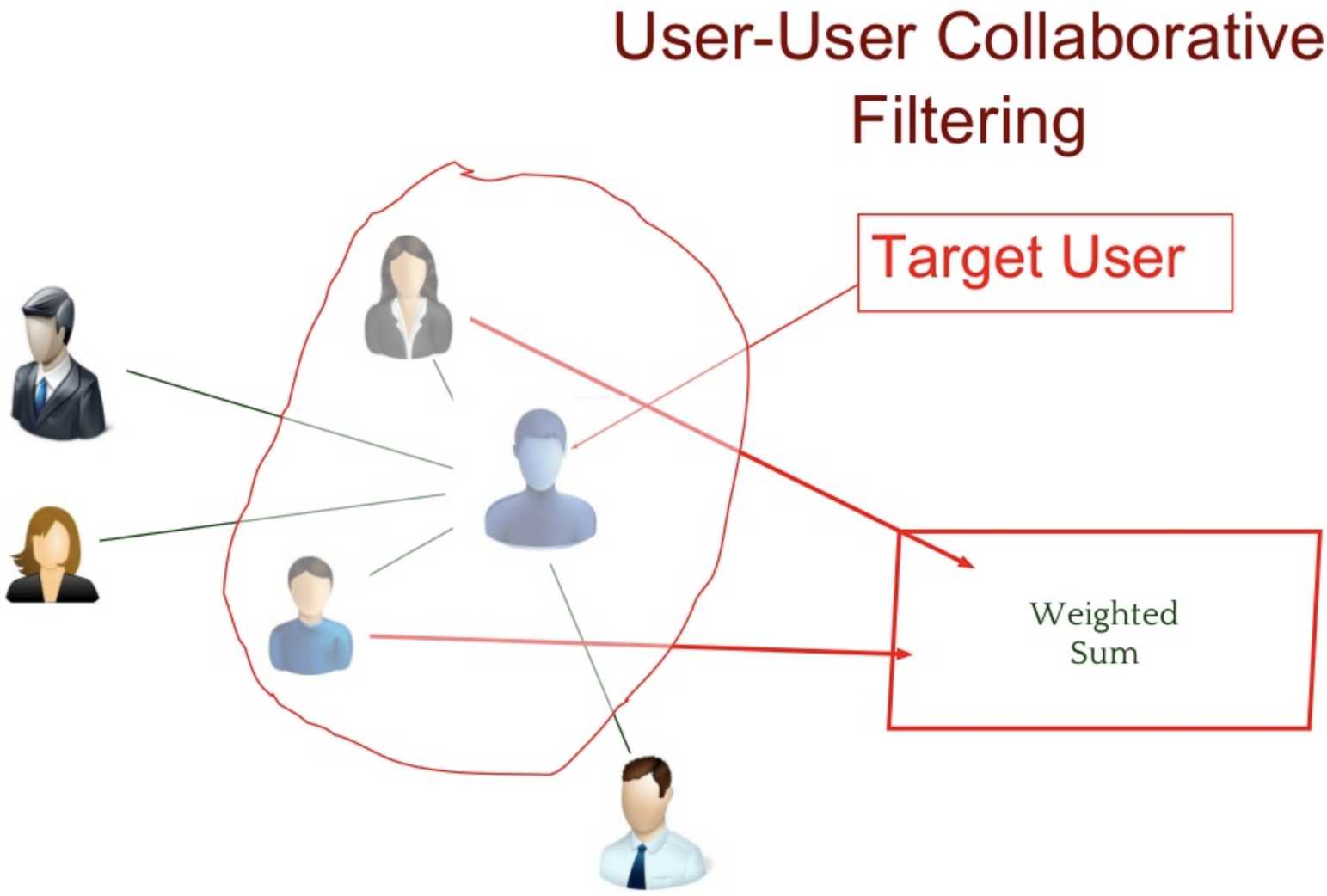


Figure 1: User-based collaborative filtering

Steps in user-based collaborative filtering

Suppose we want to predict the rating that the target user u would give to item j. In user-based collaborative filtering, we would take the following steps to accomplish the task:

1. Identify set of ratings for the target user u
2. Neighborhood formation - Identify set of users most similar to target user u according to a similarity function
3. Generate a prediction - Take the weighted average of the ratings provided by the users (for item j) in the neighborhood of the target user u

Neighborhood formation

k-Nearest neighbor of target user u, denoted by P_u, is the set of k users with the highest similarity with target user u. There are various ways to compute similarity between users:

- Pearson similarity
- Raw-cosine similarity
- Adjusted-cosine similarity
- ...

Pearson similarity function is most widely used in user-based collaborative filters

Pearson-correlation coefficient

I_u : Set of item indices for which ratings have been specified by user u
I_v : Set of item indices for which ratings have been specified by user v
μ_u: Mean rating for user u computed using her specified ratings
r_{uk}: Rating of user u for item k

Pearson correlation coefficient between user u and v, denoted by $Pearson(u, v)$, is given by equation 1

$$Pearson(u, v) = \frac{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u)(r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u)^2} \sqrt{\sum_{k \in I_u \cap I_v} (r_{vk} - \mu_v)^2}} \quad (1)$$

Prediction function

The predicted rating of user u for item j, denoted by $\hat{r}_{uj}$, is given by equation 2:

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P_u} Pearson(u, v)(r_{vj} - \mu_v)}{\sum_{v \in P_u} |Pearson(u, v)|} \tag{2}$$

From the above expression, it can be seen that we first determine the neighborhood of the target user u (P_u) and then take the weighted average of the mean-centered ratings, for item j, provided by the users in the neighborhood of target user u (r_{vj}). Each mean-centered rating is weighted with the Pearson-correlation coefficient of it's owner to the target user ($Pearson(u, v)$)

Example of user-based collaborative filtering

Item-Id ⇒	1	2	3	4	5	6	Mean Rating
User-Id ⇓							
1	7	6	7	4	5	4	5.5
2	6	7	?	4	3	4	4.8
3	?	3	3	1	1	?	2
4	1	2	2	3	3	4	2.5
5	1	?	1	2	3	3	2

Table 1: Rating matrix

Let's consider the rating matrix given by table 1. We want to predict the ratings target user 3 would give to items 1 and 6. The steps involved in the prediction process are:

- Neighborhood formation of target user 3 using pearson similarity
- Predicting ratings using prediction function

Neighborhood formation using pearson similarity

We will use equation 1 to compute the Pearson-correlation coefficient between target user 3 and all the other users.

$$Pearson(1,3) = 0.894$$
$$Pearson(2,3) = 0.939$$
$$Pearson(3,3) = 1$$
$$Pearson(4,3) = -1$$
$$Pearson(5,3) = -0.817$$

In this example, let's use the 2-nearest neighbors of target user 3. According to the definition of nearest neighbors, the 2-nearest neighbors of target user 3 are users 1 and 2.

$$P_3 = \{1, 2\}$$

Rating prediction

Having computed the 2-nearest neighbors of target user 3 (P_3), we can predict the ratings user 3 would give to items 1 ($\hat{r}_{31}$) and 6 ($\hat{r}_{36}$) using equation 2

$$\hat{r}_{31} = 2 + \frac{1.5 * 0.894 + 1.2 * 0.939}{0.894 + 0.939} = 3.35$$

$$\hat{r}_{36} = 2 + \frac{-1.5 * 0.894 - 0.8 * 0.939}{0.894 + 0.939} = 0.86$$

From the above predictions we can observe that $\hat{r}_{31} > \hat{r}_{36}$. Therefore, item 1 should be prioritized over item 6 as a recommendation to target user 3

Phases in Neighborhood-based methods

Neighborhood-based methods can be partitioned into two phases:

- Offline phase: User-User similarity values and neighborhood of users are computed
- Online phase: Similarity values and neighborhoods are leveraged to make prediction with the use of prediction function (equation 2)

Offline phase is computationally intensive, and hence the computation complexity of neighborhood-based methods arises from the Offline phase.

Challenges in neighborhood-based methods

The two main challenges in neighborhood-based methods are:

- Sparsity: Sparse rating matrices limits the coverage of neighborhood-based methods. It creates challenges for robust similarity computation when the number of mutually rated items between two users is small
- Scalability: Offline phase of neighborhood-based methods can sometimes be impractical in large-scale settings

The Sparsity Problem

- Large product sets, user ratings for a small percentage of item
- Amazon: Millions of books and a user may have bought hundreds of books
  - Probability that two users that have bought 100 books have a common book (in a catalogue of 1 million books) is 0.01 (with 50 and 10 million is 0.0002)
- If none of the target users' neighbors have rated an item, then it is not possible to provide a rating prediction of that item. Therefore, sparsity limits the coverage of neighborhood-based methods

The Scalability Problem

- Nearest neighbor algorithms require computations that grows with both the number of customers and products
- With millions of customers and products a web-based recommender can suffer serious scalability problem
- Worst case complexity is $O(mn)$ (m customers and n products)
- In practice the complexity is $O(m + n)$ since for each customer only a small number of products are considered

Idea of latent factor model

- Basic idea is to exploit the correlation between the rows and columns of rating matrix R
- R has built-in redundancies and can be approximated well by product of low-rank factors

$$R \approx UV^T$$

- Low-rank approximation is possible even for sparse R
- The fully-specified low-rank approximation provides a robust estimation of the missing entries of R

Low-rank intuition for latent factor model

To build intuition let's consider a fully-specified rating matrix R with 7 users and 6 movies and the movies can be categorized into 2 genres. The rank-2 approximation of R is given below in figure 4:

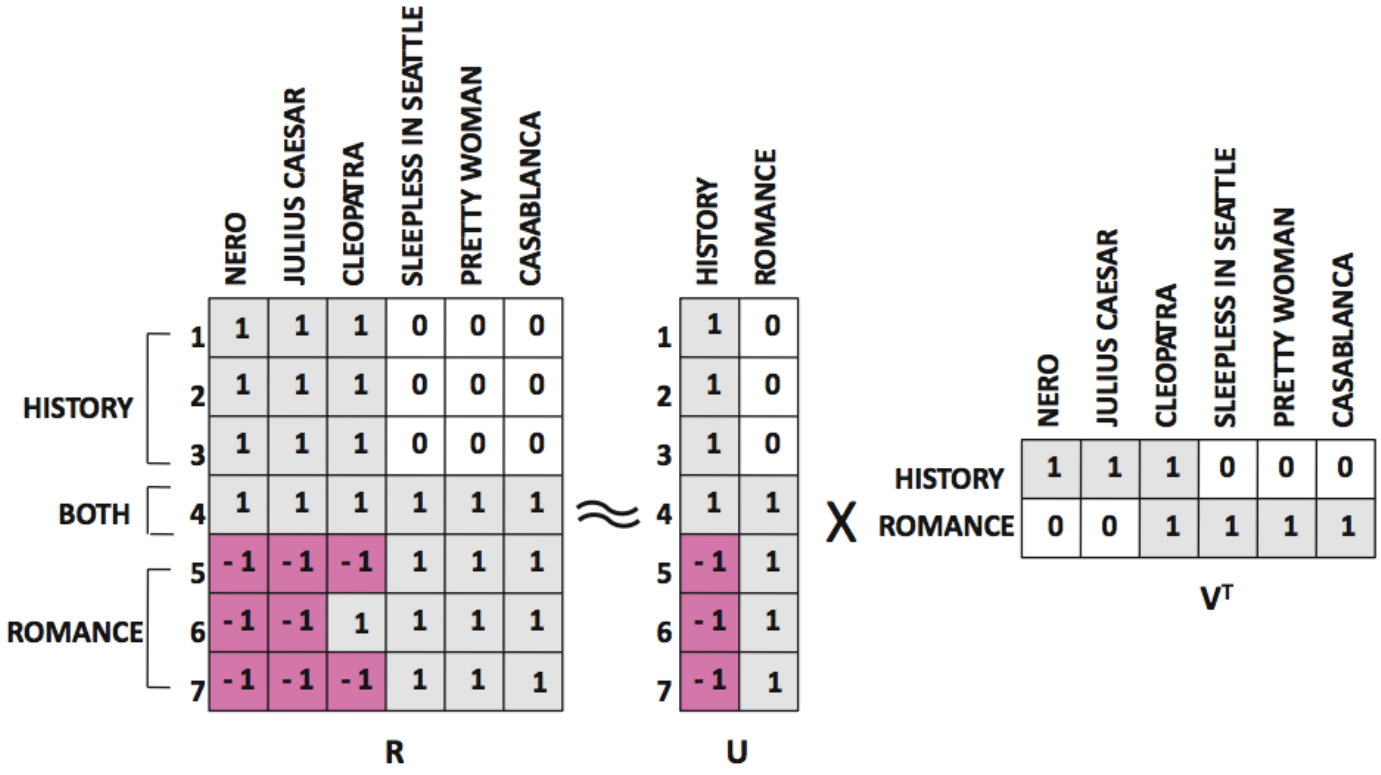


Figure 2: Example of rank-2 matrix factorization

Low-rank intuition (continued)

- Rows of U are the user latent vectors
- Columns of V^T are the item latent vectors
- From the rank-2 approximation we can conclude the following:
  - Users 1 to 3 like historical movies but are neutral to romance genre
  - User 4 likes movies of both genres
  - Users 5 to 7 like movies belonging to the romance genre, but they explicitly dislike historical movies

In this example, R was fully specified, and therefore the factorization is not particularly helpful from the perspective of missing value estimation (rating prediction). However, we can still approximate sparse R into product of low-rank factors using optimization formulation of matrix factorization.

Unconstrained matrix factorization

Let the set of all user-item pairs (i, j), which are observed in R, be denoted by S

$$S = \{(i, j) : r_{ij} \textit{is observed}\}$$

Then the unconstrained optimization problem for matrix factorization is given by equation 3

$$\underset{U,V}{\text{minimize}} \quad \frac{1}{2} \sum_{(i,j) \in S} (r_{ij} - \sum_{s=1}^{k} u_{is}.v_{js})^2 \tag{3}$$

where $U \in \mathbb{R}^{m \times k}, V \in \mathbb{R}^{n \times k}$. Since matrix R is sparse, so the observed set S of ratings is small, which can cause overfitting. A common approach for addressing this problem is to use regularization.

Unconstrained matrix factorization with regularization

The unconstrained optimization problem for matrix factorization with regularization is given by equation 4

$$\underset{U,V}{\text{minimize}} \quad \frac{1}{2}\sum_{(i,j)\in S}(r_{ij} - \sum_{s=1}^{k} u_{is}.v_{js})^2 + \frac{\lambda}{2}\sum_{i=1}^{m}\sum_{s=1}^{k} u_{is}^2 + \frac{\lambda}{2}\sum_{j=1}^{n}\sum_{s=1}^{k} v_{js}^2 \tag{4}$$

The regularization parameter λ is always non-negative and it controls the weight of the regularization term. Although, we have used the same regularization parameter for both the item and user factors in the above formulation, but they can be different.

Different matrix factorization methods

There are many variants to the unconstrained matrix factorization formulation depending on the modification to the objective function and the constraint set. We will discuss two such variations in this lecture:

- Non-Negative Matrix Factorization (NNMF)
- Matrix Factorization with Bias (MF with Bias)

Non-Negative Matrix Factorization (NNMF)

NNMF is a form of matrix factorization in which the matrices U and V are constrained to be non-negative. The optimization formulation of NNMF is given by equation 5

$$\begin{aligned} \underset{U,V}{\text{minimize}} \quad & \frac{1}{2}\sum_{(i,j)\in S}(r_{ij}-\sum_{s=1}^{k}u_{is}.v_{js})^2+\frac{\lambda}{2}\sum_{i=1}^{m}\sum_{s=1}^{k}u_{is}^2+\frac{\lambda}{2}\sum_{j=1}^{n}\sum_{s=1}^{k}v_{js}^2 \\ \text{subject to} \quad & U\geq 0, V\geq 0 \end{aligned} \tag{5}$$

The major advantage of NNMF is not necessarily one of accuracy, but that of the high level of interpretability it provides in understanding the user-item interactions.

Interpretability advantages of NNMF

In order to illustrate the interpretability advantage of NNMF, let's consider an example with 6 items and 6 customers

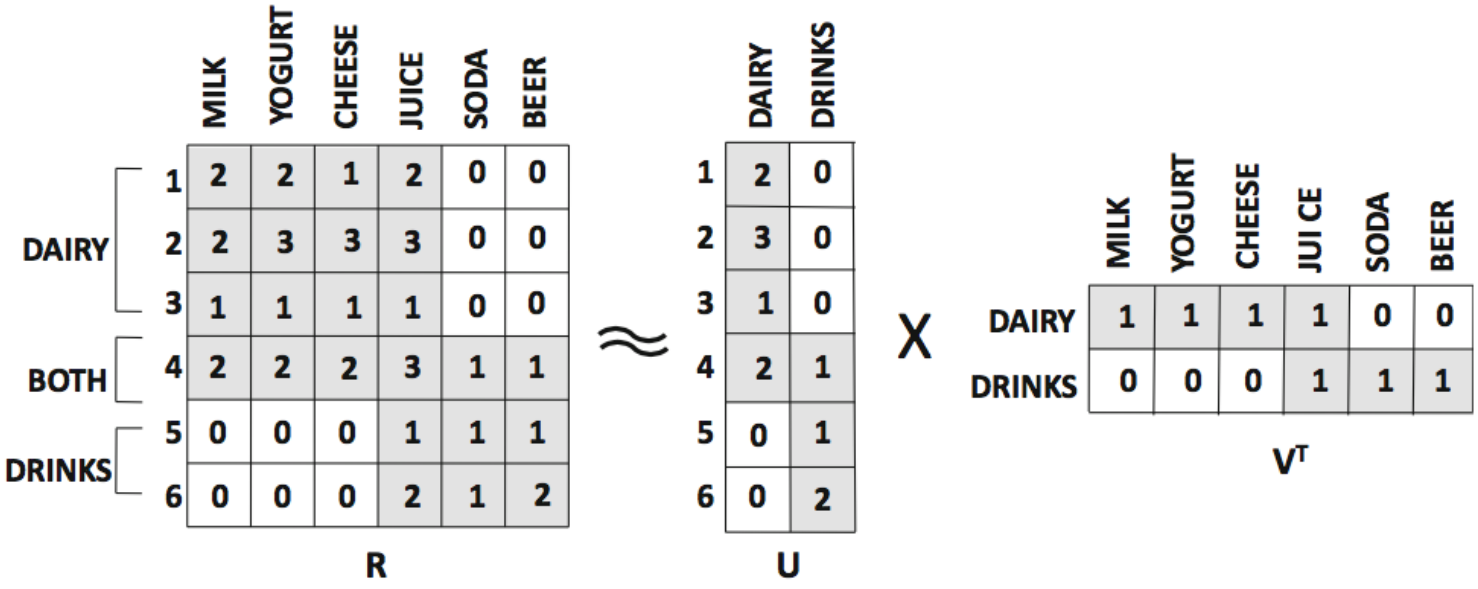


Figure 3: Example of Non-negative matrix factorization

Interpretability advantages of NNMF (continued)

- 2 classes of products corresponding to dairy products and drinks. These classes of items are referred to as latent factors
- Factor matrices (U, V) provide a clear interpretability about the affinity of customers and items to these latent factors
  - U captures the user-latent factor affinity
  - V captures the item-latent factor affinity
  - From the U in figure 3, we observe that customers 1 to 4 like dairy products whereas customers 4 to 6 like drinks

Since entries of U and V are non-negative so we have the following "sum-of-parts" decomposition:

$$\begin{aligned} r_{ij} \approx & [(\text{Affinity of user } i \text{ to dairy factor}) \\ & \times (\text{Affinity of item } j \text{ to dairy factor})] \\ & + [(\text{Affinity of user } i \text{ to drinks factor}) \\ & \times (\text{Affinity of item } j \text{ to drinks factor})] \end{aligned}$$

Each of these parts (terms inside the square bracket) is viewed as user-item co-cluster

Prediction function in NNMF

The optimal U and V found by solving the optimization problem given by equation 5 can be used for prediction in NNMF.The predicted rating of user i for item j, denoted by $\hat{r}_{ij}$, is given by equation 6

$$\hat{r}_{ij} = \sum_{s=1}^{k} u_{is} \cdot v_{js} \tag{6}$$

Matrix Factorization with Bias (MF with bias)

In MF with bias, we modify the cost function of the optimization formulation given by equation 5 by adding bias terms for each user and each item. Then with this modification, the optimization formulation of MF with bias is given by equation 7

$$\underset{U,V,b_u,b_i}{\text{minimize}} \quad \frac{1}{2}\sum_{(i,j)\in S}(r_{ij}-\sum_{s=1}^{k}u_{is}.v_{js})^2+\frac{\lambda}{2}\|U\|_F^2+\frac{\lambda}{2}\|V\|_F^2+\frac{\lambda}{2}\sum_{u=1}^{m}b_u^2 + \frac{\lambda}{2}\sum_{i=1}^{n}b_i^2 \tag{7}$$

In the above formulation, b_u is the bias of user u and b_i is the bias of item i, and we jointly optimize over U, V, b_u, b_i to find the optimal values. Learning the individual biases of the users and the items enhances the performance of the biased matrix factorization in predicting the missing ratings.

Prediction function in MF with bias

The optimal U, V, b_u, b_i found by solving the optimization problem given by equation 7 can be used for prediction in MF with bias.The predicted rating of user i for item j, denoted by $\hat{r}_{ij}$, is given by equation 8

$$\hat{r}_{ij} = \mu + b_i + b_j + \sum_{s=1}^{k} u_{is} \cdot v_{js} \tag{8}$$

where μ is the mean of all ratings, b_i is the bias of user i, and b_j is the bias of item j.

Ranking:an alternate formulation of recommendation problem

There are two primary ways in which a recommendation problem may be formulated.

- Prediction version of problem: Predict the rating value for a user-item combination
- Ranking version of problem: Recommend the top t items to a target user

We have looked at various collaborative filtering techniques for solving the prediction version. In the next couple of slides, we will explore algorithms for solving the ranking version.

Solving the ranking problem

There are two approaches to solve the ranking problem:

1. Design algorithms for solving the ranking problem directly
2. Solve the prediction problem and then rank the predictions

In the first approach, ranking algorithms are designed by directly optimizing ranking-based evaluation criterion such as the mean-reciprocal rank. Since we have explored methods for solving the prediction problem, so for continuity we will explore the second approach for solving the ranking problem.

Ranking using predictions

The main idea of the second approach is that it is possible to rank all the items using the predicted ratings. Recommendation list for a target user can be generated in the following manner:

- For the target user, compute it's predicted ratings for all the items using one of the collaborative filtering techniques
- Store the predicted ratings as a list L
- Sort L in descending order
- Select the first t-items from the sorted list L to recommend to the user

Evaluating ranking

There are many graphical techniques to evaluate the relevance of a ranked list, among which the most popular ones are:

- Receiver operating characteristic (ROC) curve
- Precision-Recall curve

Both of the methods gives similar evaluation, but precision-recall curve is more interpretable in the context of ranking evaluation.

Precision and Recall

In the context of ranking, precision and recall are defined as follows:

- Precision: It is the percentage of recommended items that are relevant (ground-truth positives)
- Recall: It is the percentage of the relevant items (ground-truth positives) that have been recommended

The mathematical expressions for precision and recall are given by equations 9 and 10 respectively

$$Precision(t) = \frac{|S(t) \cap G|}{|S(t)|} \tag{9}$$

$$Recall(t) = \frac{|S(t) \cap G|}{|G|} \tag{10}$$

where $S(t)$ is the set of top t recommended items and G is the the set of items liked by the target user (ground-truth positives)

Precision-Recall curve

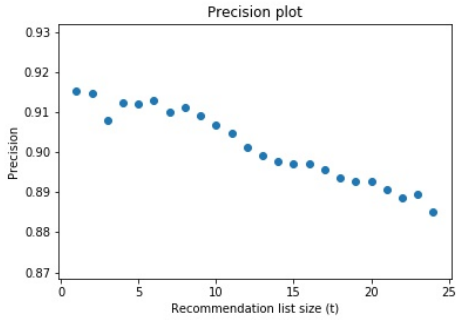


Figure 4: Precision plot

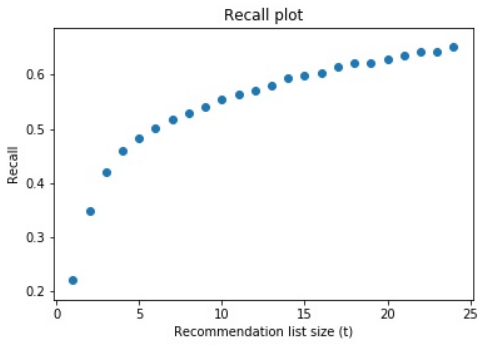


Figure 5: Recall plot

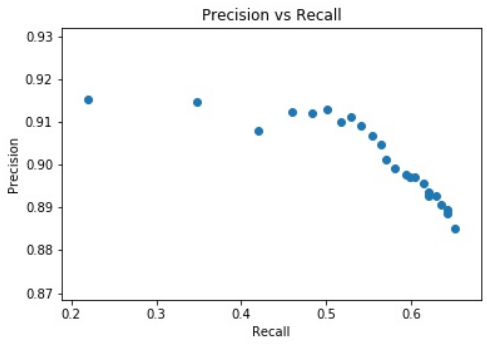


Figure 6: Precision-Recall plot

From figure 4, we can observe that precision plot is not monotonic in t. This is in compliance with equation 9, where both numerator and denominator are functions of t and may change with t differently. Due the non-monotonic nature of precision plot, the precision-recall curve is also non-monotonic.