Collaborative Filtering

Recommendation System

Agenda

- What is collaborative filtering
- Types of collaborative filtering
- User-user, item-item
- Challenges
- Steps
- Matrix Factorization - SVD
- SVD for collaborative filtering
- Surprise library
- Hands-on

greatlearning
Learning for Life

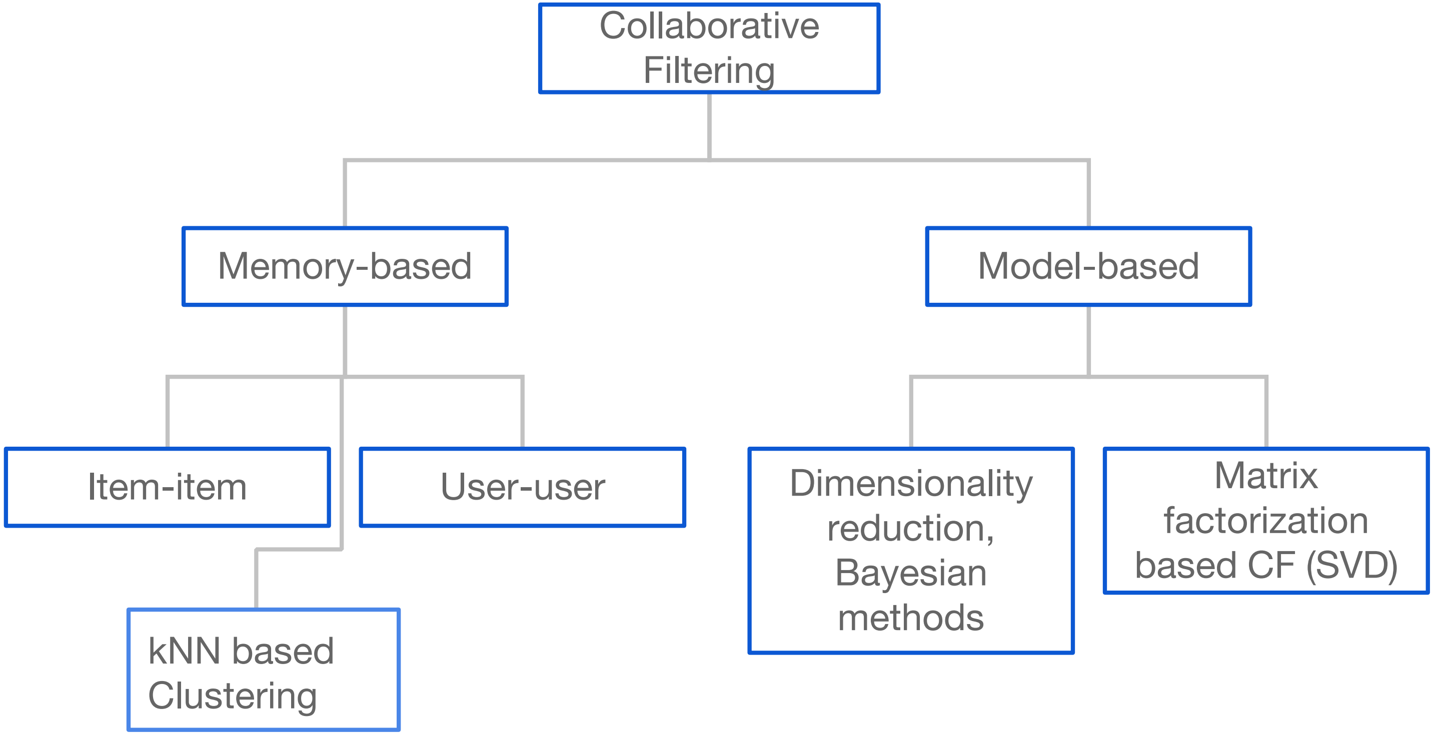
Recommendation systems (TOC)

S. No.	Topic	Scope	Objective
1	What is collaborative filtering	Understand collaborative filtering, recommendation using collaborative models	To be able to define collaborative filtering, understand the theory behind it
2	Types of collaborative filtering	Model based, memory based, Item-item, user-user	To discuss various ways to build a collaborative filtering model
3	Steps	Discuss major steps involved in CF, similar user, rating calculation, evaluation metrics	
4	Matrix Factorization	Matrix factorization, different techniques like SVD, NMF, SVD for recommendation	To utilize svd for building recommendation systems
5	Surprise library	How to deal with surprise dataset, SVD, KNNWithMeans, uid, iid,	To be able to work with surprise library

Collaborative Filtering

- Technique to filter items that a user might like
- Based on assumption that people who like similar things have similar taste
- Idea is to find similarity between users and items.
- Uses user behavior for recommendation
- Algorithm is based on the past behavior and not on context
- Data contains set of users and items and rating/reaction
- Make use of rating rating matrix to find similar users

Types of Collaborative Filtering



greatlearning
Learning for Life

Collaborative Filtering

Memory based	Model Based
• Similarity between users and/or items are calculated and used as weights to predict a rating • Approach collaborative filtering problem using the entire dataset • Not learning any parameter here. Non-parametric approaches. • Quality of prediction is good • Scalability is an issue Eg - Item based, User based, kNN clustering	• Uses statistics and machine learning methods on rating matrix • Speed and scalability issues can be addressed • Can solve problem of huge database & sparse matrix • Better at dealing with sparsity • Predicts ratings of unrated items • Inference is not traceable due to hidden factors Eg-Matrix factorization (SVD, PMF, NMF), Neural nets based

Collaborative Filtering

Item based - similarity between each pair of items is calculated.

- Neighboring items are considered
- Let's say item X and Y are purchased together, and if someone is buying X then Y will be recommended to him

User based - similar users are considered

- Let's say A and B like same movies, then a new movie liked by A will be recommended to B as well

Collaborative Filtering

User based	Item based
Similar users are considered “Users who are similar to you also liked…” $pred(u,i) = \bar{r}_u + \frac{\sum_{n \subset neighbors(u)} sim(u,n) \cdot (r_{ni} - \bar{r}_n)}{\sum_{n \subset neighbors(u)} sim(u,n)}$	Similar items are considered “Users who liked this item also liked…” More efficient as number of item would be less compared to number of user $pred(u,i) = \frac{\sum_{j \in ratedItems(u)} sim(i,j) \cdot r_{ui}}{\sum_{j \in ratedItems(u)} sim(i,j)}$

Challenges with CF

- **Cold Start problem:** The model requires enough amount of other users already in the system to find a good match.
- **Sparsity:** If the user/ratings matrix is sparse, and it is not easy to find the users that have rated the same items.
- **Popularity Bias**: Not possible to recommend items to users with unique tastes.
    - Popular items are recommended more to the users as more data being available on them
    - This may begin a positive feedback loop not allowing the model to recommend items with less popularity to the users
- **Shilling attacks**
    - Users can create fake preferences to push their own items
- **Scalability**
    - Collaborative filtering models are computationally expensive

Dataset - rating matrix

Matrix with mostly empty cells is called **sparse**, and the opposite to that (a mostly filled matrix) is called **dense**.

Explicit ratings :

- Users rate for an item
- Most accurate description of a user's preference
- Challenging in collecting data

Implicit ratings:

- Observation of user behaviour
- Can be collected with less cost to user
- Ratings inference may not be precise

Steps for CF

1. Determine similar users
    a. Calculate similarity matrix using similarity distance and user-item ratings. Get top similar neighbors
2. Estimate rating that a user would give to an item based on the ratings of similar users
    a. Estimated rating R for a user U for an item I would be close to average rating given to I by the top n users most similar to U
    b. $Ru = (\sum^{n}_{u=1} Ru)/n$
    c. Weighted average - multiply each rating by similarity factor
3. Accuracy of estimated ratings
    a. RMSE (root mean squared error)
    b. MAE (mean absolute error)

Note - user bias can be removed by subtracting mean rating given by that user to all the items for each item rated by that user.

Matrix factorization

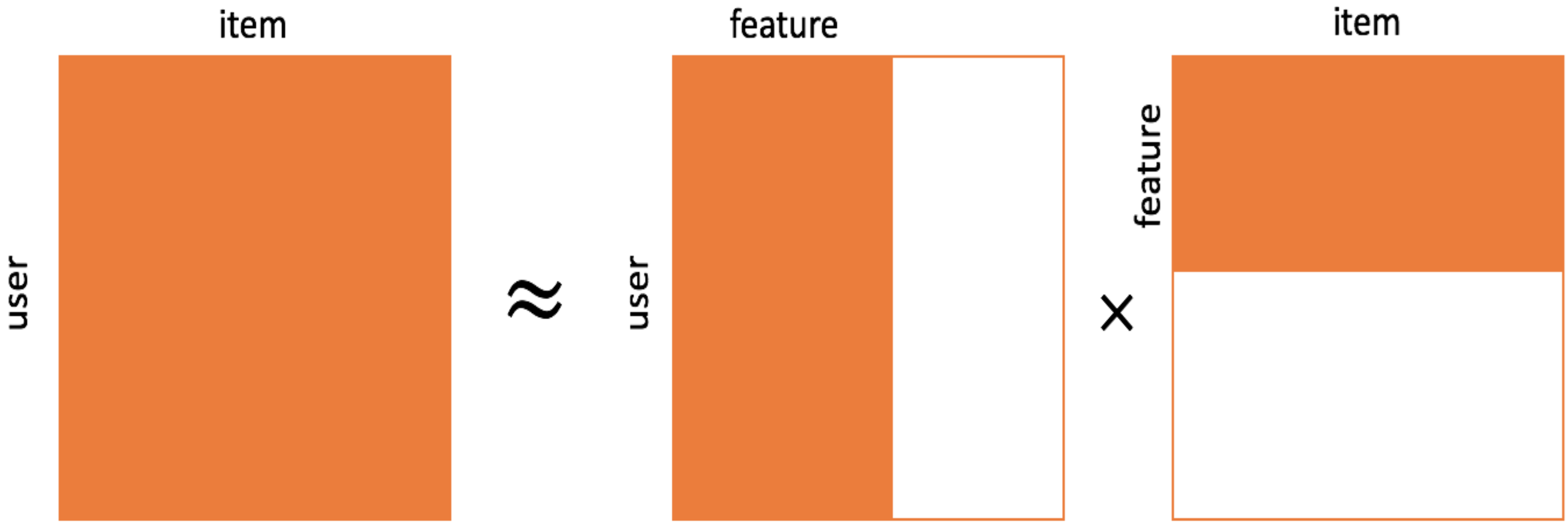
- Idea is to find preferences using some hidden factors
- Idea is to break down a large matrix (user-item) into a product of smaller ones

  Ex - 12 = 6*2, 3*4

  Similarly, a matrix **A** with dimension m*n can be reduced to product of two matrices **X** and **Y** with dimension m*p and p*n respectively. [p should be common]

- Here, m and n matrices represents latent factors
- Techniques
  - SVD - singular value decomposition (orthogonal factorization)
  - PMF - probabilistic factorization
  - NMF - non-negative factorization

Matrix Factorization



Singular value decomposition

- Given a square or non square matrix A, linear Algebra theorem SVD specifies that: $A_{(m\times n)} = U_{(m\times m)} * S_{(m\times n)} * V_{(n\times n)}^T$

Where,

A is m×n matrix

U is an m×m orthogonal matrix

S is a m×n diagonal matrix- (singular value)

V is a n×n orthogonal matrix

- The matrices U and V are orthogonal so: $U^T.U=V.V^T=I$

The columns of U are orthonormal eigenvectors of AA^T , the columns of V are orthonormal eigenvectors of A^TA, and S is a diagonal matrix containing the square roots of eigenvalues from U or V in descending order.

Singular value decomposition

n matrix (2*4)

	2		
	1.6		

m (3*2) matrix

3	1.2

	4		

Matrix A : m*n (3*4)

Surprise Library

- It’s a add-on package for scipy. It is hosted and developed separately from main scipy distribution
- Comes with various recommender algorithms and similarity metrics
- How to install -
    - `$ pip install scikit-surprise`
    - `$ conda install -c conda-forge scikit-surprise`
- Common commands
    - `from surprise import Dataset, Reader, SVD, KNNWithMeans`
    - `from surprise.model_selection import GridSearchCV`
- `# Loads Pandas dataframe`
    - `df = pd.DataFrame(ratings_dict)`
    - `reader = Reader(rating_scale=(1, 5))`
    - `data = Dataset.load_from_df(df[["user", "item", "rating"]], reader)`

Ref - https://surprise.readthedocs.io/en/stable/

Hands-on

Case study

Thank you!

Happy Learning :)