Collaborative filtering

- Used by recommender systems. Has two senses : narrow and general.
- Method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating)
- ***Underlying assumption*** **:** if a person *A* has the same opinion as a person *B* on an issue, A is more likely to have B's opinion on a different issue *x* than to have the opinion on x of a person chosen randomly.

Introduction

Problem Statement :

- Growth of the Internet has made it much more difficult to effectively extract useful information from all the available online information.
- Overwhelming amount of data necessitates mechanisms for efficient information filtering.
- Collaborative filtering is one of the techniques used for dealing with this problem.

Motivation :

People often get the best recommendations from someone with tastes similar to themselves. Collaborative filtering encompasses techniques for matching people with similar interests and making recommendations on this basis.

Requirement of CF algorithms :

- Users' active participation
- An easy way to represent users' interests
- Algorithms that are able to match people with similar interests.

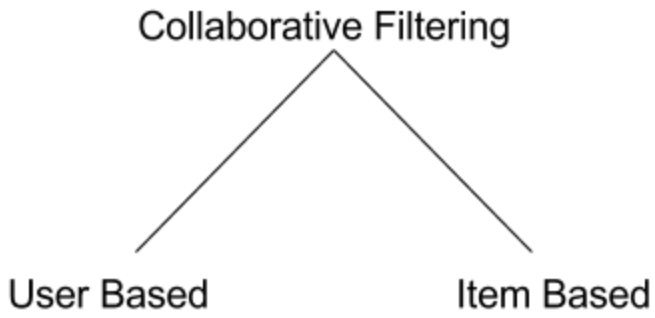
Typical workflow of a CF system :

1. A user expresses his preferences by rating items of the system. These ratings can be viewed as an approximate representation of the user's interest in the corresponding domain.
2. The system matches this user's ratings against other users' and finds the people with most "similar" tastes.
3. With similar users, the system recommends items that the similar users have rated highly but not yet being rated by this user.(presumably the absence of rating is often considered as the unfamiliarity of an item)

Key Problem : How to combine and weight the preferences of user neighbors ??

Possible solution : Sometimes, users can immediately rate the recommended items. As a result, the system gains an increasingly accurate representation of user preferences over time.

Methodology

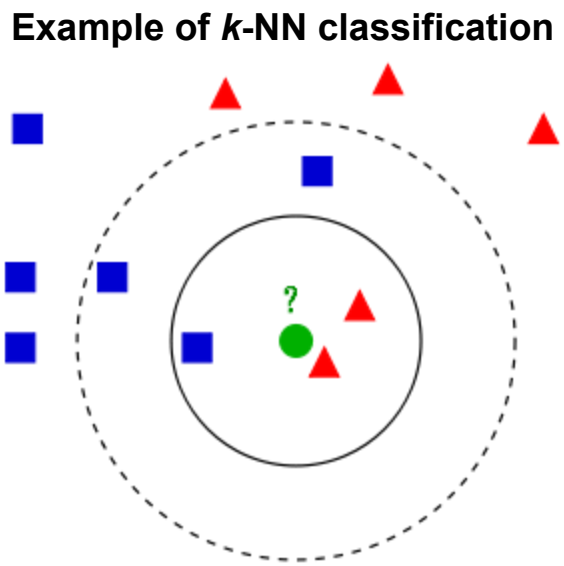


User Based CF:

1. Look for users who share the same rating patterns with the active user (the user whom the prediction is for).
2. Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user.

Example : *User-based Nearest Neighbor algorithm*

Intuition : Each **user can be thought of a vector** with features being the ratings given by the user to the sites. Now instance based learning methods can be used to classify a users based on the features.



The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If *k = 3* (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If *k = 5* (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).

Recommendation Procedure : Each user can be recommended based on the ratings of k nearest users.

- **Computing similarity between users** : For this measures like Euclidean Distance, Cosine Similarity etc can be used.
- **Computing recommendation score :** This can be done by taking a weighted average of the ratings of k nearest users. Weights can be taken as the similarity measure between users.

Item Based CF:

1. Build an item-item matrix determining relationships between pairs of items.

2. Infer the tastes of the current user by examining the matrix and matching that user's data.

   Examples :
   Slope One Algorithms
   Amazon's item-to-item patented algorithm

Slope One based approach :

Customer	Item A	Item B	Item C
Amit	5	3	2
Saachi	3	4	Didn't rate it
Ayush	Didn't rate it	2	5

Intuition : Compute and store the information about ***difference in overall rating of item x and item y.***

- Average difference in ratings between item B and A is (2+(-1))/2=0.5. Hence, on average, item A is rated above item B by 0.5.
- Similarly, the average difference between item C and A is 3.
- Hence, if we attempt to predict the rating of Ayush for **item A using his rating for item B**, we get 2+0.5 = 2.5.
- Similarly, if we try to predict her rating for **item A using her rating of item C**, we get 5+3=8.

Which one should we choose ?

Solution : Take the weighted average of the ratings of items.

What weights should we choose ?

Solution : Number of users having rated both items.

In the above example, we would predict the following rating for Ayush on item A:

$$\frac{2 \times 2.5 + 1 \times 8}{2 + 1} = 4.33$$

Complexity : Hence, given *n* items, to implement Slope One, all that is needed is to compute and store the average differences and the number of common ratings for each of the n^2 **pairs of items**.

Deeper Analysis

- Suppose there are ***n* items**, ***m* users**, and ***N* ratings**.
- Computing the average rating differences for each pair of items :
    - Space complexity : $\mathbf{n^2}$
    - Time complexity : $\mathbf{mn^2}$
- Better bound (Assuming users have rated upto y items) for Time Complexity : n^2+my^2
- Suppose user has entered **x** ratings :
    - Predicting a single rating : **x** time steps
    - Predicting all his missing ratings : $(n-x)x$ time steps.
    - Updating database when user enters a new rating : **x** time steps.

Can space complexity be reduced ?

- Partitioning the data
- Using sparse storage : pairs of items having no (or few) users can be omitted.

But what should we do if the ratings are not available ?

Solution : Get the ratings out of other data available.

Example : Amazon's item-to-item patented algorithm.

- Data available is whether a particular user purchased / not purchased a particular item.

Customer	Item 1	Item 2	Item 3
Amit	Bought it	Didn't buy it	Bought it
Saachi	Didn't buy it	Bought it	Bought it

Ayush	Didn't buy it	Bought it	Didn't buy it

- This data can be converted into a binary rating (1 - Purchased, 0 - Not Purchased)

Customer	Item 1	Item 2	Item 3
Amit	1	0	1
Saachi	0	1	1
Ayush	0	1	0

- Now **each Item can be treated as a vector in m dimensions.**
- **Cosine Similarity** between each pair of item can be calculated.

  Similarity (Item 1, Item 2)

  $$\frac{(1,0,0)\cdot(0,1,1)}{\|(1,0,0)\|\|(0,1,1)\|}=0$$

  Similarity (Item 1, Item 3)

  $$\frac{(1,0,0)\cdot(1,1,0)}{\|(1,0,0)\|\|(1,1,0)\|}=\frac{1}{\sqrt{2}}$$

  Similarity (Item 2, Item 3)

  $$\frac{(0,1,1)\cdot(1,1,0)}{\|(0,1,1)\|\|(1,1,0)\|}=\frac{1}{2}$$

- Recommendation procedure :
  - a user visiting item 1 would receive item 3 as the 1st recommendation.
  - a user visiting item 2 would receive item 3 as the 1st recommendation.
  - a user visiting item 3 would receive item 1 (and then item 2) as a recommendation.

Complexity : The model uses a single parameter per pair of item (the cosine) to make the recommendation. Hence, if there are *n* items, up to *n(n-1)/2* cosines need to be computed and stored. Therefore time and space complexity is **$O(n^2)$**