Machine Learning for Business

Module 14: Recommender Engine

Day 7, 13.00 – 16.00

Asst. Prof. Dr. Santitham Prom-on

Department of Computer Engineering, Faculty of Engineering
King Mongkut's University of Technology Thonburi

BIG DATA EXPERIENCE CENTER

Content-based filtering

Part 1

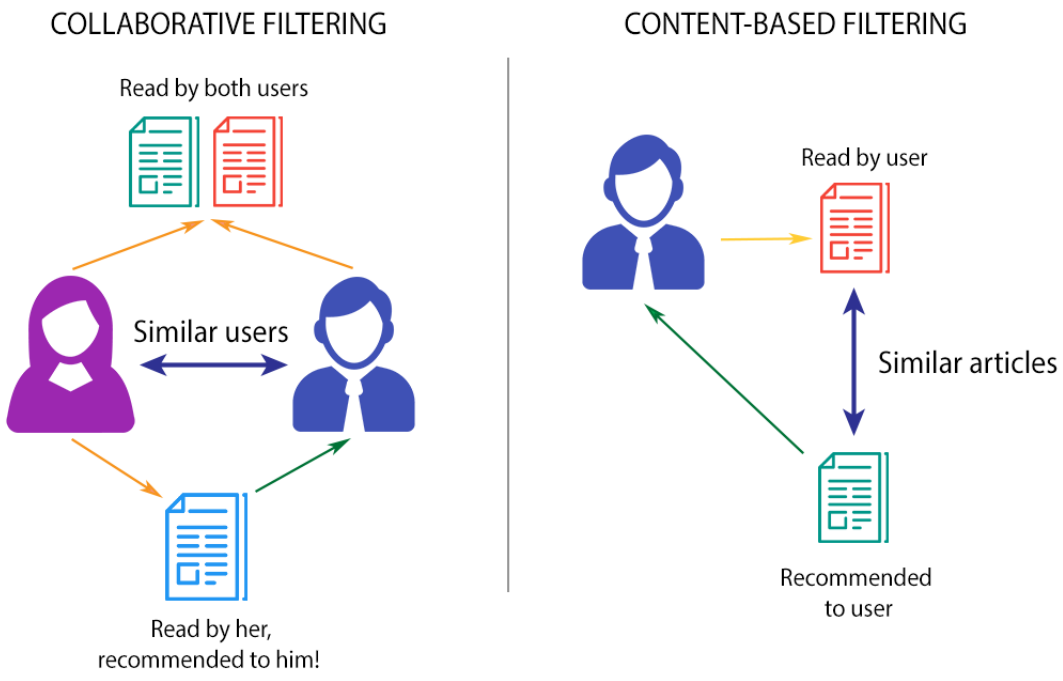
BIG DATA EXPERIENCE

What is a recommender?

- A system that provides recommendation
- Recommendation systems use a number of different technologies. We can classify these systems into two broad groups.
  - Content-based systems examine properties of the items recommended, e.g. Netflix
  - Collaborative filtering systems recommend items based on similarity measures between users and/or items. The items recommended to a user are those preferred by similar users.

g·able

BIG DATA EXPERIENCE



g·able

Utility Matrix

Example 9.1 : In Fig. 9.1 we see an example utility matrix, representing users' ratings of movies on a 1–5 scale, with 5 the highest rating. Blanks represent the situation where the user has not rated the movie. The movie names are HP1, HP2, and HP3 for *Harry Potter* I, II, and III, TW for *Twilight*, and SW1, SW2, and SW3 for *Star Wars* episodes 1, 2, and 3. The users are represented by capital letters *A* through *D*.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Applications of recommender systems

1. Product Recommendations: Perhaps the most important use of recommendation systems is at on-line retailers.
2. Movie Recommendations: Netflix offers its customers recommendations of movies they might like.
3. News Articles: News services have attempted to identify articles of interest to readers, based on the articles that they have read in the past.

BIG DATA EXPERIENCE

Content-based recommendation Item profiles

- In a content-based system, we must construct for each item a profile, which is a record or collection of records representing important characteristics of that item.
- For examples,
  - The set of actors of the movie. Some viewers prefer movies with their favorite actors.
  - The director. Some viewers have a preference for the work of certain directors.
  - The year in which the movie was made. Some viewers prefer old movies; others watch only the latest releases.
  - The genre or general type of movie. Some viewers like only comedies, others dramas or romances.

g·able

BIG DATA EXPERIENCE

R: load data (movies)

```
movies <- read.csv("movies.csv")
str(movies)
```

```
'data.frame':   9125 obs. of  3 variables:
 $ movieId: int  1 2 3 4 5 6 7 8 9 10 ...
 $ title  : Factor w/ 9123 levels "¡Three Amigos! (1986)",..: 8301 4319 1
1 3591 6860 8253 7673 3287 ...
 $ genres : Factor w/ 902 levels "(no genres listed)",..: 329 394 687 64
377 2 124 ...
```

	movieId	title	genres
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|
2	2	Jumanji (1995)	Adventure|Children|Fantasy
3	3	Grumpier Old Men (1995)	Comedy|Romance
4	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	5	Father of the Bride Part II (1995)	Comedy
6	6	Heat (1995)	Action|Crime|Thriller
7	7	Sabrina (1995)	Comedy|Romance

Showing 1 to 7 of 9,125 entries

g·able

R: load data (rating)

```
ratings <- read.csv("ratings.csv")
str(ratings)
```

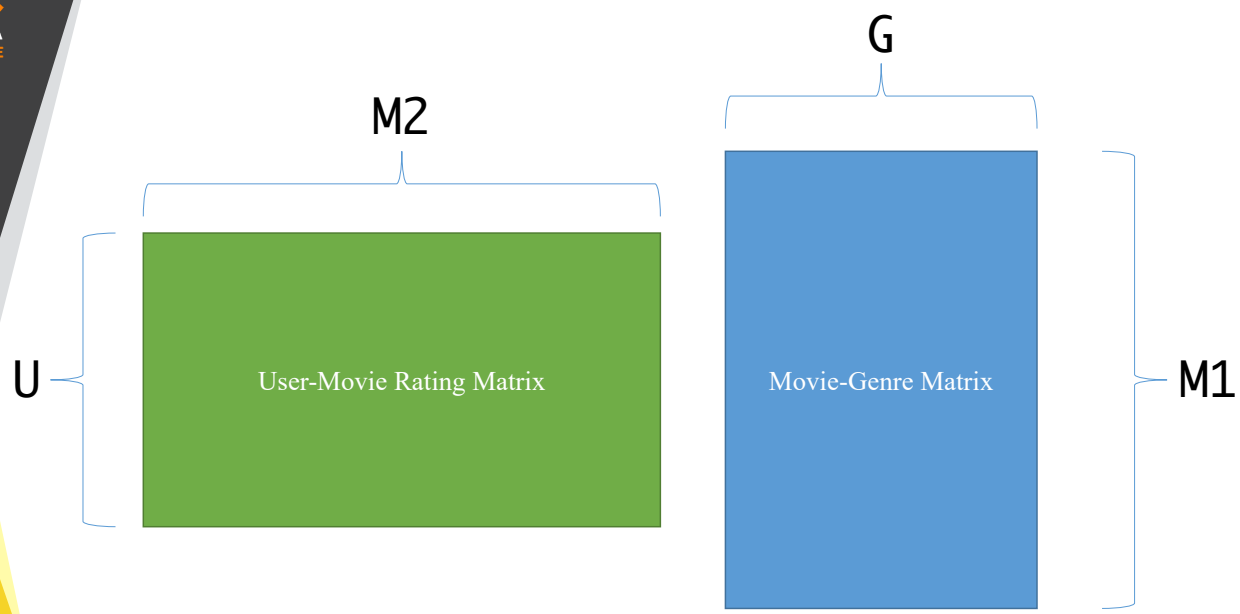
```
'data.frame':   100004 obs. of  4 variables:
 $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
 $ movieId  : int  31 1029 1061 1129 1172 1263 12
 $ rating   : num  2.5 3 3 2 4 2 2 2 3.5 2 ...
 $ timestamp: int  1260759144 1260759179 12607591
1260759187 1260759148 1260759125 1260759131 ...
```

	userId	movieId	rating	timestamp
1	1	31	2.5	1260759144
2	1	1029	3.0	1260759179
3	1	1061	3.0	1260759182
4	1	1129	2.0	1260759185
5	1	1172	4.0	1260759205

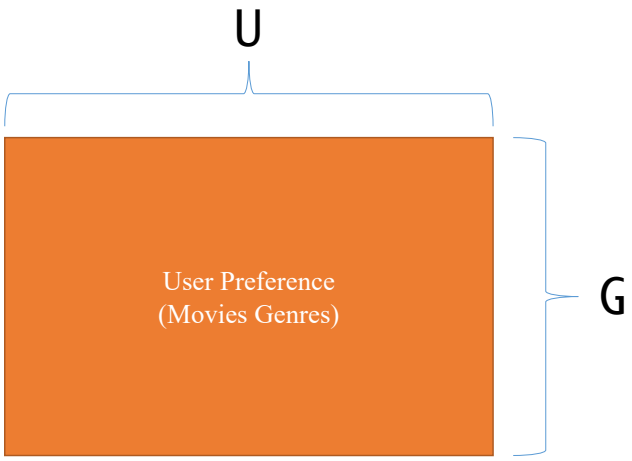
Showing 1 to 6 of 100,004 entries

Movie recommender based on genres

- We will be building a basic content-based recommender engine based on *movie genres* only.
- A user profile is necessary to determine what a user is inclined to prefer, and can be constructed based on the user's preferences or viewing behavior



Content-based filtering
What we are aiming for…



BIG DATA EXPERIENCE

Data preprocessing

- To obtain the movie utility matrix, the pipe-separated genres available in the movies dataset had to be split.
- The data.table package has a tstrsplit() function that works well here to perform string splits.

```
install.packages("data.table")
library(data.table)
genres <- as.data.frame(movies$genres, stringsAsFactors = FALSE)
genres2 <- as.data.frame(tstrsplit(genres[,1],'[|]',type.convert =
TRUE), stringsAsFactors = FALSE)
colnames(genres2) <- 1:7
```

g·able

BIG DATA EXPERIENCE

```
View(genres2)
```

	1	2	3	4	5	6	7	NA	NA	NA
1	Adventure	Animation	Children	Comedy	Fantasy	<NA>	<NA>	<NA>	<NA>	<NA>
2	Adventure	Children	Fantasy	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>
3	Comedy	Romance	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>
4	Comedy	Drama	Romance	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>
5	Comedy	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>
6	Action	Crime	Thriller	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>

g·able

BIG DATA EXPERIENCE

Genre Utility (Feature) Matrix (1)

- Then you create a matrix with columns representing every unique genre, and indicate whether a genre was present or not in each movie.

```
genre_list <- c("Action", "Adventure", "Animation", "Children", "Comedy",
"Crime","Documentary", "Drama", "Fantasy","Film-Noir", "Horror",
"Musical", "Mystery","Romance","Sci-Fi", "Thriller", "War", "Western")
genre_matrix <- matrix(0,9126,18) #empty matrix
genre_matrix[1,] <- genre_list #set first row to genre list
colnames(genre_matrix) <- genre_list #set column names to genre list
```

g·able

BIG DATA EXPERIENCE

Mapping the values

```
#iterate through matrix
for (i in 1:nrow(genres2)) {
 for (c in 1:ncol(genres2)) {
    genmat_col = which(genre_matrix[1,] ==
genres2[i,c])
    genre_matrix[i+1,genmat_col] <- 1
 }
}
```

g·able

Genre Utility (Feature) Matrix (2)

```
#convert into dataframe
genre_matrix2 <- as.data.frame(genre_matrix[-1,], stringsAsFactors=FALSE)
#remove first row, which was the genre list
for (c in 1:ncol(genre_matrix2)) {
      genre_matrix2[,c] <- as.integer(genre_matrix2[,c])
} #convert from characters to integers
```

	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fa
1	0	1	1	1	1	0	0	0	
2	0	1	0	1	0	0	0	0	
3	0	0	0	0	1	0	0	0	
4	0	0	0	0	1	0	0	1	
5	0	0	0	0	1	0	0	0	
6	1	0	0	0	0	1	0	0	
7	0	0	0	0	1	0	0	0	
8	0	1	0	1	0	0	0	0	
9	1	0	0	0	0	0	0	0	
10	1	1	0	0	0	0	0	0	
11	0	0	0	0	1	0	0	1	
12	0	0	0	0	1	0	0	0	

Showing 1 to 12 of 9,125 entries

BIG DATA EXPERIENCE

g·able

User Profile Matrix

- This can be easily done with the dcast() function in the reshape2 package.
- We first convert the ratings into a binary format to keep things simple.
- Ratings of 4 and 5 are mapped to 1, representing likes, and ratings of 3 and below are mapped to -1, representing dislikes.

BIG DATA EXPERIENCE

g·able

Creating user profile matrix

```
binaryratings <- ratings
binaryratings$rating <-
ifelse(binaryratings$rating > 3,
1, -1)
```

	userId	movieId	rating	timestamp
1	1	31	-1	1260759144
2	1	1029	-1	1260759179
3	1	1061	-1	1260759182
4	1	1129	-1	1260759185
5	1	1172	1	1260759205
6	1	1263	-1	1260759151
7	1	1287	-1	1260759187
8	1	1293	-1	1260759148
9	1	1339	1	1260759125
10	1	1343	-1	1260759131
11	1	1371	-1	1260759135
12	1	1405	-1	1260759203

Showing 1 to 12 of 100,004 entries

Reshaping the user profile matrix

- To obtain the binaryratings matrix in the correct format we need, We use the dcast() function in the reshape2 package.
- This basically transforms the data from a long format to a wide format.
- This also creates many NA values because not every user rated every movie.
- We substituted the NA values with 0.

Reshaping the user profile matrix

```
binaryratings2 <- dcast(binaryratings, movieId~userId, value.var =
"rating", na.rm=FALSE)
for (i in 1:ncol(binaryratings2)){
  binaryratings2[which(is.na(binaryratings2[,i]) == TRUE),i] <- 0
}
#remove movieIds col. Rows are movieIds, cols are userIds
binaryratings2 = binaryratings2[,-1]
```

cast {reshape2} R Documentation

Cast functions Cast a molten data frame into an array or data frame.

Description

Use `acast` or `dcast` depending on whether you want vector/matrix/array output or data frame output. Data frames can have at most two dimensions.

Usage

```
dcast(data, formula, fun.aggregate = NULL, ..., margins = NULL,
  subset = NULL, fill = NULL, drop = TRUE,
  value.var = guess_value(data))
```

BIG DATA EXPERIENCE

Remove movies that have never been rated

```
#Remove rows that are not rated from movies
movieIds <- unique(movies$movieId)
datasetmovieIds <- unique(movies$movieId)
ratingmovieIds <- unique(ratings$movieId)
movies2 <- movies[-which((movieIds %in% ratingmovieIds) == FALSE),]
rownames(movies2) <- NULL
#Remove rows that are not rated from genre_matrix2
genre_matrix3 <- genre_matrix2[-which((movieIds %in% ratingmovieIds) == FALSE),]
rownames(genre_matrix3) <- NULL
```

BIG DATA EXPERIENCE

Dot Product

- Now we can calculate the dot product of the genre matrix and the ratings matrix and obtain the user profiles.

BIG DATA EXPERIENCE

Dot Product

```
#Calculate dot product for User Profiles
result = matrix(0,18,671)
for (c in 1:ncol(binaryratings2)){
  for (i in 1:ncol(genre_matrix3)){
    result[i,c] <- sum((genre_matrix3[,i]) *
(binaryratings2[,c]))
  }
}
result <- ifelse(result < 0, 0, 1)
```

g·able

BIG DATA EXPERIENCE

User profile matrix

- This user profiles shows the aggregated inclination of each user towards movie genres.
- Each column represents a unique userId, and positive values shows a preference towards a certain genre.
- The values were again simplified into a binary matrix — positive values were mapped to 1 to represent likes, negative values were mapped to 0 to represent dislikes.

g·able

Content-based or Collaborative

- Now that we have the user profiles, we can go 2 ways from here.
  1. Predict if a user likes an item based on the item descriptions (movie genres). This can be done by predicting user movie ratings. (content-bases)
  2. Assume that users like similar items, and retrieve movies that are closest in similarity to a user's profile, which represents a user's preference for an item's feature. (Collaborative)

Content-based filtering

```
colnames(result) <- colnames(binaryratings2)
rownames(result) <- colnames(genre_matrix3)
View(result)
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Action	0	0	1	1	1	0	0	1	0	1	1	1	1	0	0	1	1	0	0	1
Adventure	0	1	0	1	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0	1
Animation	0	1	0	1	1	1	1	1	1	1	1	0	1	0	0	1	0	1	1	1
Children	0	0	0	1	1	1	1	1	1	1	1	0	1	0	0	1	0	1	1	1
Comedy	0	0	1	1	1	0	0	1	1	1	1	0	1	0	0	1	1	0	0	1
Crime	1	1	1	1	1	1	0	1	1	0	1	1	1	1	0	1	1	0	1	0
Documentary	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1
Drama	0	1	1	1	1	1	0	1	1	1	1	0	1	0	0	1	1	0	0	1
Fantasy	0	0	0	1	1	1	0	1	1	1	1	0	1	1	0	0	1	1	1	1
Film-Noir	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
Horror	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	1	1	0	0	1
Musical	0	1	0	1	1	1	1	1	1	0	1	0	1	0	0	1	1	1	1	1

Showing 1 to 12 of 18 entries

BIG DATA EXPERIENCE

Collaborative filtering

- We use Jaccard Distance to measure the similarity between user profiles, and the movie genre matrix.
- Jaccard Distance is suitable for set binary data.
- We will use the dist() function from the proxy library to calculate Jaccard Distance.

g·able

BIG DATA EXPERIENCE

Calculate Jaccard distance

```
result2 <- result[,1] #First user's profile
sim_mat <- rbind.data.frame(result2, genre_matrix3)
sim_mat <- data.frame(lapply(sim_mat,function(x){as.integer(x)}))

#Calculate Jaccard distance between user profile and all movies
library(proxy)
sim_results <- dist(sim_mat, method = "Jaccard")
sim_results <- as.data.frame(as.matrix(sim_results[1:9067]))
```

g·able

Get the most similar

```
rows <- which(rank(sim_results,
ties.method='min') <= 3)
#Recommended movies
result2
movies[rows,]
```

Recommendation

```
result2
```

Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama
0	0	0	0	0	1	1	0
Fantasy	**Film-Noir**	**Horror**	**Musical**	**Mystery**	**Romance**	**Sci-Fi**	**Thriller**
0	1	1	0	1	1	0	0
War	**Western**						
0	0						

```
movies[rows,]
```

	movieId	title	genres
763	942	Laura (1944)	Crime|Film-Noir|Mystery
1039	1284	Big Sleep, The (1946)	Crime|Film-Noir|Mystery
1186	1464	Lost Highway (1997)	Crime|Drama|Fantasy|Film-Noir|Mystery|Romance
4104	5378	Star Wars: Episode II - Attack of the Clones (2002)	Action|Adventure|Sci-Fi|IMAX
5691	25886	Random Harvest (1942)	Drama|Romance
6128	33145	Lot Like Love, A (2005)	Comedy|Drama|Romance
8408	104879	Prisoners (2013)	Drama|Mystery|Thriller
9067	152025	SOMM: Into the Bottle (2016)	Documentary

BIG DATA EXPERIENCE

Content-based filtering

Good

- It does not require a lot of user data.
- It just needs item data and you are able to start giving recommendations to users.
- It does not depend on lots of user data, so it is possible to give recommendations to even your first customer.

Bad

- Your item data needs to be well distributed.
- It won't be effective to have a content-based recommender if 80% of your movies are action movies.
- Complements are more likely discovered through collaborative techniques.

g·able

BIG DATA EXPERIENCE

Collaborative filtering

Part 2

g·able

BIG DATA EXPERIENCE

User-Based Collaborative Filtering

- The User-Based Collaborative Filtering approach groups users according to prior usage behavior or according to their preferences, and then recommends an item that a similar user in the same group viewed or liked.
- To put this in layman terms,
  - User 1 liked movie A, B and C
  - User 2 liked movie A and B
  - Then movie C might make a good recommendation to User 2.
- The User-Based Collaborative Filtering approach mimics how word-of-mouth recommendations work in real life.

g·able

BIG DATA EXPERIENCE

User-Based Collaborative Filtering

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1 	8	1	?	2	7
User 2 	2	?	5	7	5
User 3 	5	4	7	4	7
User 4 	7	1	7	3	8
User 5 	1	7	4	6	5
User 6 	8	3	8	3	7

g·able

BIG DATA EXPERIENCE

Similarity between users

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1 	8	1	?	2	7
User 2 	2	?	5	7	5
User 4 	7	1	7	3	8

- How similar are users 1 and 2?
- How similar are users 1 and 5?

- How do you calculate similarity?

g·able

BIG DATA EXPERIENCE

Similarity between users: simple way

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1 	8	1	?	2	7
User 2 	2	?	5	7	5

- Only consider items both users have rated
- For each item:
  - Calculate difference in the users' ratings
  - Take the average of this difference over the items

$$\text{Sim}(\text{User1}, \text{User2}) = \frac{\Sigma_j \,|\, \text{rating}(\text{User1}, \text{Item } j) - \text{rating}(\text{User2}, \text{Item } j) \,|}{\text{Num. of items}}$$

g·able

BIG DATA EXPERIENCE

g·able

Problem: similarity between users

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	1	2	3	4	5
User 2	5	4	3	2	1

Sim(User1, User2) = 12/5 = 2.4

	Item 1	Item 2	Item 3	Item 4	Item 5
User 3	1	2	3	4	5
User 4	4	5	6	7	8

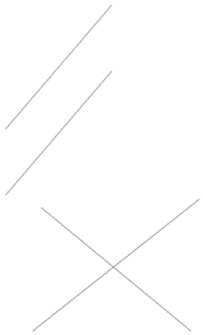
Sim(User3, User4) = 15/5 = 3

BIG DATA EXPERIENCE

g·able

Better solution

- Use Statistical Correlation Metrics (e.g., Pearson's)
  - These measure how well two data sets fit on a straight line
  - Corrects for grade inflation



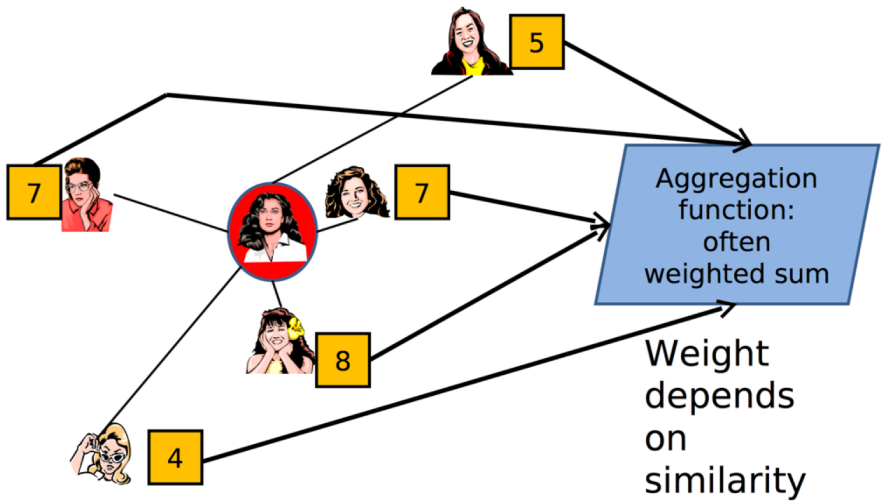
Perfect Correlation for User3, User4

Inverse Correlation for User1, User2

BIG DATA EXPERIENCE

g·able

Naïve method: using entire matrix



BIG DATA EXPERIENCE

g·able

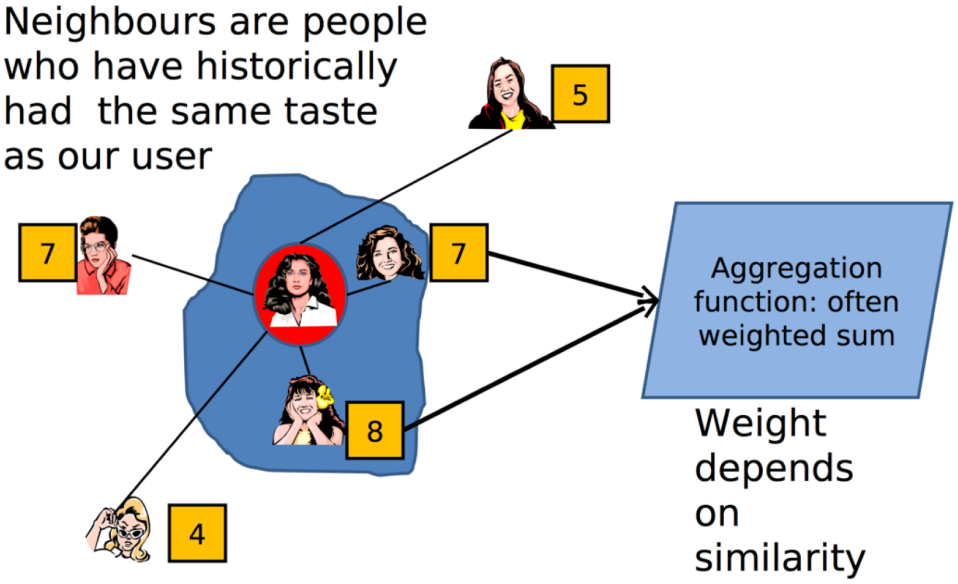
Problem: handling large matrices

- MovieLens database
  - 100k dataset = 1682 movies & 943 users
  - 1 million dataset = 3900 movies & 6040 users
- Netflix dataset
  17700 movies, 250k users, 100 million ratin
- Realistically, cannot make use of all users in real time

BIG DATA EXPERIENCE

g·able

Better way: K-nearest neighbor



BIG DATA EXPERIENCE

g·able

recommenderlab

- We will use User-Based Collaborative Filtering to generate a top-10 recommendation list for users using the recommenderlab package available in R.
- The recommenderlab package makes it really easy to implement some of the popular collaborative filtering algorithms.

recommenderlab's UBCF

- The User-based Collaborative Filtering recommender model was created with recommenderlab with the below parameters and the ratings matrix:

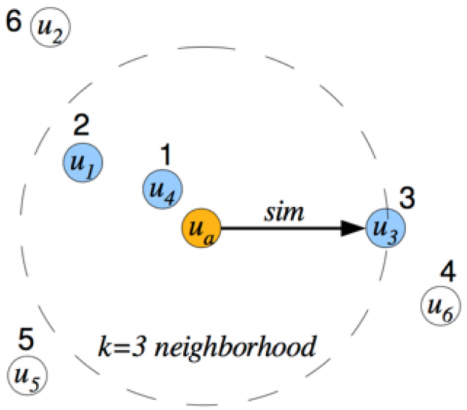
  Method: UBCF
  Similarity Calculation Method: Cosine Similarity
  Nearest Neighbors: 30

- The predicted item ratings of the user will be derived from the 5 nearest neighbors in its neighborhood.
- When the predicted item ratings are obtained, the top 10 most highly predicted ratings will be returned as the recommendations.

UBCF

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	?	4.0	4.0	2.0	1.0	2.0	?	?
u_2	3.0	?	?	?	5.0	1.0	?	?
u_3	3.0	?	?	3.0	2.0	2.0	?	3.0
u_4	4.0	?	?	2.0	1.0	1.0	2.0	4.0
u_5	1.0	1.0	?	?	?	?	?	1.0
u_6	?	1.0	?	?	1.0	1.0	?	1.0
u_a	?	?	4.0	3.0	?	1.0	?	5.0
$\hat{r}_a$	3.5	4.0			1.3		2.0	

(a)



(b)

BIG DATA EXPERIENCE

Data preprocessing

```
library(reshape2)
#Create ratings matrix. Rows = userId, Columns = movieId
rating_matrix <- dcast(ratings, userId~movieId, value.var = "rating", na.rm=FALSE)
rating_matrix <- as.matrix(rating_matrix[,-1])
#remove userIds
```

g·able

BIG DATA EXPERIENCE

Rating matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	2
1	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
2	NA	NA	NA	NA	NA	NA	NA	NA	NA	4	NA	NA	NA	NA	NA	NA	5	NA	NA	NA	NA	
3	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
4	NA	NA	NA	NA	NA	NA	NA	NA	NA	4	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
5	NA	NA	4	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
6	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
7	3.0	NA	NA	NA	NA	NA	NA	NA	NA	3	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	3.0	
8	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
9	4.0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	4	NA	NA	NA	NA	
10	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
11	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	

Showing 1 to 11 of 671 entries

g·able

BIG DATA EXPERIENCE

realRatingMatrix

```
library(recommenderlab)

#Convert rating matrix into a
recommenderlab sparse matrix
ratingmat <- as(rating_matrix,
"realRatingMatrix")
```

g·able

BIG DATA EXPERIENCE

Create a recommender

```
e <- evaluationScheme(
          ratingmat,
          method = "split",
          train = 0.9,
          k = 1,
          given = 15
)
r <- Recommender(
          getData(e, "train"),
          "UBCF”
)
```

g·able

BIG DATA EXPERIENCE

Make recommendation

```
p <- predict(r,
             getData(e, "known"),
             type="topNList")
p.list <- getList(p)
movies[as.integer(p.list[[1]]),]
```

g·able

BIG DATA EXPERIENCE

Recommendation

	movieId	title	genres
318	352	Crooklyn (1994)	Comedy|Drama
858	1059	William Shakespeare's Romeo + Juliet (1996)	Drama|Romance
3578	4539	Salsa (1988)	Musical|Romance
111	122	Boomerang (1992)	Comedy|Romance
1136	1397	Bastard Out of Carolina (1996)	Drama
527	594	Snow White and the Seven Dwarfs (1937)	Animation|Children|Drama|Fantasy|Musical
2858	3575	Defying Gravity (1997)	Drama
1148	1413	Whole Wide World, The (1996)	Drama
50	52	Mighty Aphrodite (1995)	Comedy|Drama|Romance
2571	3198	Papillon (1973)	Crime|Drama

g·able

Prediction Accuracy

```
p1 <- predict(r,
              getData(e, "known"),
              type="ratings")
calcPredictionAccuracy(
    p1,
    getData(e, "unknown"),
    given=15,
    goodRating=5)
```

```
      RMSE       MSE       MAE
1.0189389 1.0382364 0.7864003
```

User-Based Collaborative Filtering

Good

- It gives recommendations that can be complements to the item the user was interacting with.
- This might be a stronger recommendation than what a content-based recommender can provide as users might not be looking for direct substitutes to a movie they had just viewed or previously watched.

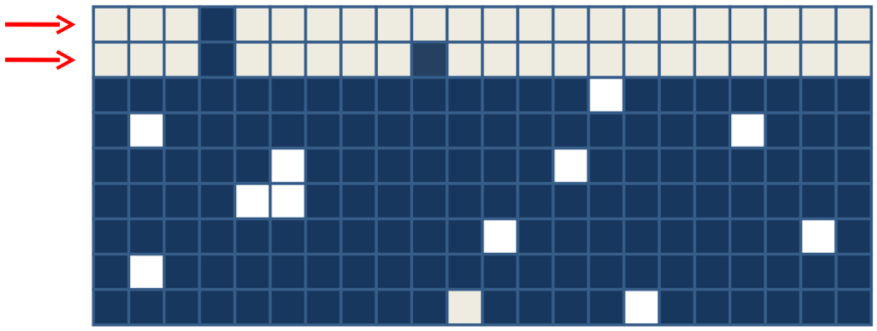
Bad

- Not scalable.
- For millions of users, this computation would be very time consuming.
- User-based collaborative filtering relies on past user choices to make future recommendations. The implications of this is that it assumes that a user's taste and preference remains more or less constant over time, which might not be true and makes it difficult to pre-compute user similarities offline.

BIG DATA EXPERIENCE

Problem with UBCF (1)

- User Cold-Start problem
  not enough known about new user to decide who is similar (and perhaps no other users yet..)



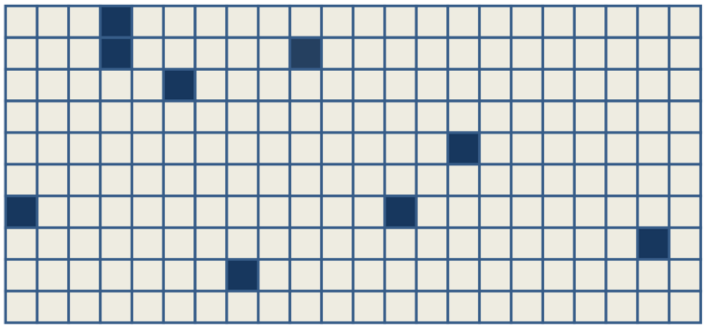
- Need way to motivate early rater

g·able

BIG DATA EXPERIENCE

Problem with UBCF (2)

- Sparsity
  when recommending from a large item set, users will have rated only some of the items
  (makes it hard to find similar users)



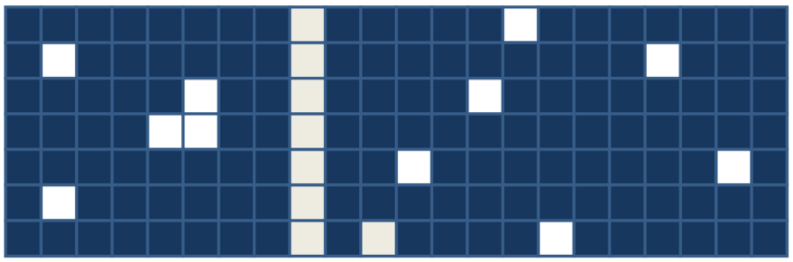
g·able

BIG DATA EXPERIENCE

g·able

Problem with UBCF (3)

- Scalability
  - with millions of ratings, computations become slow
- Item Cold-Start problem
  - Cannot predict ratings for new item till some similar users have rated it [No problem for content-based]



BIG DATA EXPERIENCE

g·able

Item-Based Collaborative Filtering

- User is likely to have the same opinion for similar items [same idea as in Content-Based Filtering]
- Similarity between items is decided by looking at how other users have rated them
  [different from Content-based, where item features are used]
  Star Wars = [Action, Sci-fi...]
  Star Wars = [User1:8, User2:3, User3:7...]
- Advantage (compared to user-based CF):
  - Prevents User Cold-Start problem
  - Improves scalability (similarity between items is more stable than between users)

BIG DATA EXPERIENCE

Example: IBCF

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1 	8	1	?	2	7
User 2 	2	?	5	7	5
User 3 	5	4	7	4	7
User 4 	7	1	7	3	8
User 5 	1	7	4	6	5
User 6 	8	3	8	3	7

g·able

BIG DATA EXPERIENCE

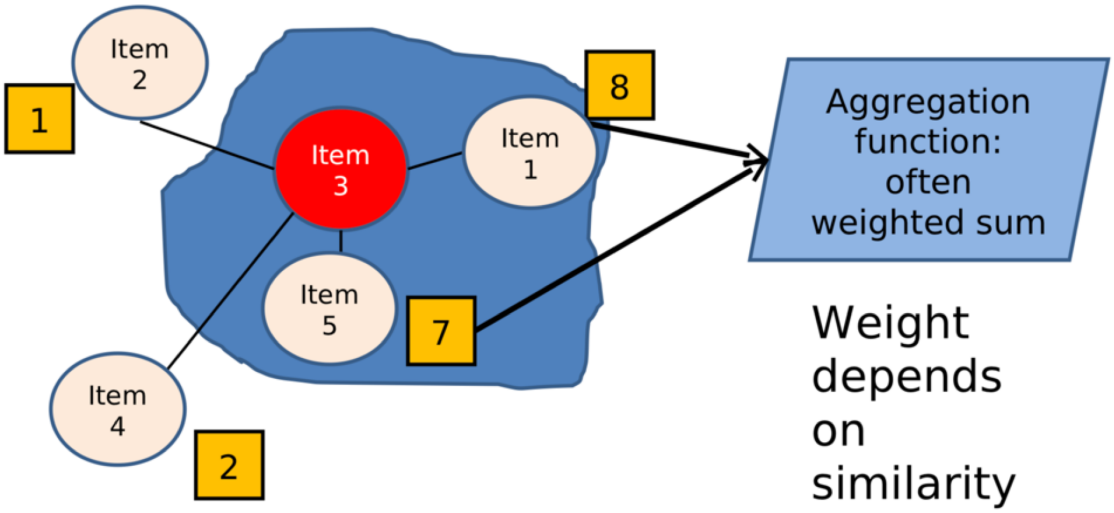
Similarity between items

Item 3	Item 4
?	2
5	7
7	4
7	3
4	6
8	3

- Only consider users who have rated both items
- For each user:
  - Calculate difference in ratings for the two items
  - Take the average of this difference over the users

$$\text{Sim}(\text{Item 3}, \text{Item 4}) = \frac{\Sigma_j \,|\, \text{rating (User j, Item 3)} - \text{rating (User j, Item 4)} \,|}{\text{Number of Users}}$$

g·able



R: creating IBCF

```
ratingmat_ib <- as(
  rating_matrix[,1:500],
  "realRatingMatrix"
)
r <- Recommender(
  ratingmat_ib,
  "IBCF"
)
```

BIG DATA EXPERIENCE

g·able

R: predicting with IBCF

```
p <- predict(
        r,
        ratingmat_ib[1],
        type="topNList")
p.list <- getList(p)
movies[as.integer(p.list[[1]]),]
```

BIG DATA EXPERIENCE

g·able

Thank you

Question?

BIG DATA EXPERIENCE CENTER