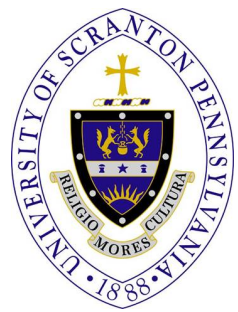
Recommender Systems
Part 2: Collaborative Filtering

Aram Balagyozyan

Department of Operations and Information Management
Kania School of Management
The University of Scranton

July 23, 2020



1. Collaborative Filtering

Introdution

- Collaborative filtering is a very popular recommender algorithm that is based on the notions of identifying relevant items for a specific user from the very large set of items ("filtering") by considering preferences of many other users ("collaboration").
- One of the best known examples is Amazon that uses collaborative filtering to not only provide personalized product recommendations, but also for customizing the entire website interface for each user.
- As this Forbes.com June 30, 2012, article describes, "At root, the retail giant's recommendation system is based on a number of simple elements: what a user has bought in the past, which items they have in their virtual shopping cart, items they've rated and liked, and what other customers have viewed and purchased. Amazon calls this homegrown math "item-to-item collaborative filtering," and it's used this algorithm to heavily customize the browsing experience for returning customers."

Data Type and Format Used in Collaborative Filtering

- Collaborative filtering requires availability of all item–user information.
- Specifically, for each item–user combination, we should have some measure of the user's preference for that item. Preference can be a numerical rating or a binary behavior such as a purchase, a 'like', or a click.
- For n users (U_1, U_2, ..., U_n) and p items (I_1, I_2, ... I_p), we can think of the data as an $n \times p$ matrix of n rows (users) by p columns (items). Each cell in the matrix includes the rating or the binary event corresponding to the user's preference of the item.

	Item ID			
User ID	I_1	I_2	...	I_p
U_1	$r_{1,1}$	$r_{1,2}$	...	$r_{1,p}$
U_2	$r_{2,1}$	$r_{2,2}$	...	$r_{2,p}$
$\vdots$				
U_n	$r_{n,1}$	$r_{n,2}$	...	$r_{n,p}$

Data Type and Format Used in Collaborative Filtering

- Typically, not every user purchases or rates every item, and therefore a purchase matrix will have many zeros (it is sparse), and a rating matrix will have many missing values.
- While non-missing values may reflect low consumer ranking for an item, non-missing values in general may be indicative of interest (as opposed to items with missing values that sometimes convey "uninterested."
- When both n and p are large, it is not practical to store the preferences data ($r_{u,i}$) in an $n \times p$ table. Instead, the data can be stored in many rows of triplets of the form (U_u, I_i, $r_{u,i}$), where each triplet contains the user ID, the item ID, and the preference information.
- To illustrate one possible use of collaborative filtering, the the textbook (Schmueli et al., 2017) presents a famous example on the Netflix Prize Contest; in 2006, Netflix, announced a one million USD contest www.netflixprize.com for the purpose of improving its recommendation system called *Cinematch*.

Data Type and Format Used in Collaborative Filtering

- The table below presents a sample of records from the Netflix Prize Contest, for a subset of 10 customers (rows) and 9 movies (columns). The real contest was based on a dataset with around 480,000 users and 18,000 movies.

	Movie ID								
Customer ID	1	5	8	17	18	28	30	44	48
30878	4	1			3	3	4	5	
124105	4								
822109	5								
823519	3		1	4		4	5		
885013	4	5							
893988	3						4	4	
1248029	3					2	4		3
1503895	4								
1842128	4						3		
2238063	3								

Method 1: User-Based Collaborative Filtering: "People Like You"

- One approach to generating personalized recommendations for a user using collaborative filtering is based on finding users with similar preferences, and recommending items that they liked but the user hasn't purchased. The algorithm has two steps:
  1. Find users who are most similar to the user of interest (neighbors). This is done by comparing the preference of our user to the preferences of other users.
  2. Considering only the items that the user has not yet purchased, recommend the ones that are most preferred by the user's neighbors.
- The second step is more or less trivial, so for now we will concentrate on the first step only.

Measuring Similarity Between Users: Pearson Correlation

- Measuring user similarity in Step 1 requires choosing a distance (or proximity) metric to measure the distance between each given user and the other users.
- Once the distances are computed, we can use a threshold on the distance or on the number of required neighbors to determine the nearest neighbors to be used in Step 2. This approach is called "user-based top-N recommendation."
- A popular proximity measure between two users is the Pearson correlation between their ratings.
- We denote the ratings of items $I_1, I_2 \ldots, I_p$ by user U_1 as $r_{1,1}$, $r_{1,2}, \ldots, r_{1,p}$ and their average by $\bar{r}_1$. Similarly, the ratings by user U_2 as $r_{2,1}, r_{2,2}, \ldots, r_{2,p}$ and their average by $\bar{r}_2$.
- The correlation proximity between the two users is defined by:

$$Corr(U_1, U_2) = \frac{\sum (r_{1,i} - \bar{r}_1)(r_{2,i} - \bar{r}_2)}{\sum \sqrt{(r_{1,i} - \bar{r}_1)^2} \sqrt{(r_{2,i} - \bar{r}_2)^2}}$$

Measuring Similarity Between Users: Pearson Correlation

- As an example, let us calculate the correlation between customer 30878 and customer 823519 in the Netflix sample in the table above. We'll assume that the data shown in the table is the entire information. First, we compute the average rating by each of these users:

$$\bar{r}_{30878} = (4 + 1 + 3 + 3 + 4 + 5)/6 = 3.3$$
$$\bar{r}_{823519} = (3 + 1 + 4 + 4 + 5)/5 = 3.4$$

- Note that the average is computed over a different number of movies for each of these customers, because they each rated a different set of movies. The average for a customer is computed over all the movies that a customer rated.

Measuring Similarity Between Users: Pearson Correlation

- Although the averages are computed over a different number of movies for each of these customers, the calculations for the correlation involve the departures from the average, but only for the items that they co-rated. In this case, the co-rated movie IDs are 1, 28, and 30:

$$Corr(U_{30878}, U_{823519}) = \frac{(4-3.3)(3-3.4)+(3-3.3)(4-3.4)+(4-3.3)(5-3.4)}{\sqrt{(4-3.3)^2+(3-3.3)^2+(4-3.3)^2}\sqrt{(3-3.4)^2+(4-3.4)^2+(5-3.4)^2}}$$
$$= 0.6/0.75 = 0.34$$

- The same approach can be used when the data are in the form of a binary matrix (e.g., purchased or didn't purchase.).

Measuring Similarity Between Users: Cosine Similarity

- Another popular measure is a variant of the Pearson correlation called *cosine similarity*. It differs from the correlation formula by not subtracting the means.
- Subtracting the mean in the correlation formula adjusts for users' different overall approaches to rating—for example, a customer who always rates highly vs. one who tends to give low ratings.
- For example, the cosine similarity between the two Netflix customers is:

$$\begin{aligned}\text{Cos Sim}(U_{30878}, U_{823519}) =& \frac{4 \times 3 + 3 \times 4 + 4 \times 5}{\sqrt{4^2 + 3^2 + 4^2}\sqrt{3^2 + 4^2 + 5^2}} \\ =& 44/45.277 = 0.972\end{aligned}$$

- Note that when the data are in the form of a binary matrix, say, for purchase or no-purchase, the cosine similarity must be calculated over all items that either user has purchased; it cannot be limited to just the items that were co-purchased.

Making Recommendation

- Using a correlation, cosign similarity, or another measure, we can compute each user's similarity to each other user.
- Then we can can look at k nearest users, and among all the other items that they rated/purchased, we chose the best one and recommend it to our user.
- What is the best item? Fir binary purchase data, it is the item most purchased. For rating data, it could be the highest rated, most rated, or a weighting of the two.

Method 2: Item-Based Collaborative Filtering: "Items That Go Together"

- When the number of users is much larger than the number of items, it is computationally cheaper to find similar items rather than similar users.
- Specifically, when a user expresses interest in a particular item, the item-based collaborative filtering algorithm has two steps:
  1. Find the items that were co-rated, or co-purchased, (by any user) with the item of interest.
  2. Recommend the most popular or correlated item(s) among the similar items.
- Similarity is now computed between items, instead of users. For example, in our small Netflix sample, the correlation between movie 1 (I_1 with average $\bar{r}_1 = 3.7$) and movie 5 (I_5 with average $\bar{r}_2 = 3$) is:

$$Corr(I_1, I_5) = \frac{(4-3.7)(1-3)+(4-3.7)(5-3)}{\sqrt{(4-3.7)^2+(4-3.7)^2}\sqrt{(1-3)^2+(5-3)^2}} = 0$$

Method 2: Item-Based Collaborative Filtering: "Items That Go Together"

- The zero correlation above is due to the two opposite ratings of movie 5 by users 30878 and 885013. One user rated it 5 stars and the other gave it a 1 star.
- In like fashion, we can compute similarity between all the movies. This can be done offline. In real time, for a user who rates a certain movie highly, we can look up the movie correlation table and recommend the movie with the highest positive correlation to the user's newly rated movie.
- One big advantage of item-based collaborative filtering is that the algorithm can produce recommendations in real-time and can scale to massive data sets.
- The disadvantage is that there is less diversity between items (compared to users' tastes), and therefore, the recommendations are often obvious.

Collaborative Filtering Analysis Using the Jester Dataset

- To demonstrate the above ideas, we will use the Jester dataset obtained from Kaggle.com.
- Jester is an online joke recommender system developed by Ken Goldberg and the team at UC Berkeley. Users are presented jokes through an HTML client interface and allowed to rate jokes. Once a user rates all jokes in the gauge set, the system recommends new jokes to the user.

```
library(readr)
jester_items<-read_csv("jester_items.csv")
jester_ratings<-read_csv("jester_ratings.csv")
```

- The dataset contains over 1.7 million continuous ratings (-10.00 to +10.00) of 140 jokes from 59,132 users.
- Each row of `jester_ratings.csv` is a rating of one joke by one user with column names `userId`, `jokeId`, `rating`. While each row of `jester_items.csv` contains information about each joke in `jester_rating.csv` with column names `jokeId` and `jokeText`.

Collaborative Filtering Analysis Using the Jester Dataset

- One thing that you should notice right away is that the `jester_ratings` data frame is not in wide form (as the Netflix ratings in one of the slides above), it is in the long form.
- Thus, we should reshape `jester_ratings` into a wide form. To that end, we will rely on the `dcast()` function of the **reshape2** package.

```
library(reshape2)
cast_jester_ratings<-dcast(jester_ratings,
                           userId~jokeId)
rownames(cast_jester_ratings)<- # rename rows
  paste('u', cast_jester_ratings$userId,sep = '')
# drop the column with user IDs
cast_jester_ratings<-cast_jester_ratings[,-1]
colnames(cast_jester_ratings)<-#rename columns
  paste('j',colnames(cast_jester_ratings),sep='')
head(cast_jester_ratings[,1:7]) # display head
```

Collaborative Filtering Analysis Using the Jester Dataset

- Below is the output of the code above.

```
##         j5     j7     j8    j13    j15    j16    j17
## u1   0.219 -9.281 -9.281 -6.781  0.875 -9.656 -9.031
## u2 -9.688  9.938  9.531  9.938  0.406  3.719  9.656
## u3 -9.844 -9.844 -7.219 -2.031 -9.938 -9.969 -9.875
## u4 -5.812 -4.500 -4.906     NA     NA     NA     NA
## u5  6.906  4.750 -5.906 -0.406 -4.031  3.875  6.219
## u6 -0.031 -9.094 -0.406  7.500 -7.219 -9.438  0.125
```

Refresher: `dcast()` and `melt()` in the **reshape2** Package

- To understand what the `dcast()` function has accomplished, consider the following hypothetical miniature dataset of user ratings:

```
toy_ratings<-data.frame(userID=c(1,2,2),
                        jokeID=c(1,1,2),
                        rating=c(-10,3,7))
toy_ratings # ratings in long format
```

```
##   userID jokeID rating
## 1      1      1    -10
## 2      2      1      3
## 3      2      2      7
```

```
cast_toy_ratings<-dcast(toy_ratings,userID~jokeID)
cast_toy_ratings # ratings in wide format
```

```
##   userID   1  2
## 1      1 -10 NA
## 2      2   3  7
```

Refresher: `dcast()` and `melt()` in the **reshape2** Package

- Since the first user rated only joke 1 but note joke 2 and user 2 rated both jokes 1 and 2, the data in wide form have an `NA`.
- The `melt()` function does the opposite of `dcast()`, it converts data from wide form to long form (it melts wide tables).
- The following code utilizes the `melt()` function to convert our molten `cast_toy_rating` data frame back to its original long form.

```
melt(cast_toy_ratings, id.vars="userID",
 na.rm = TRUE)
```

```
##   userID variable value
## 1      1        1   -10
## 2      2        1     3
## 4      2        2     7
```

Collaborative Filtering Analysis Using the Jester Dataset

- Now that we understood how `dcast()` and `melt()`work, let's get back to our example.
- For demonstration purposes, let's split our data into training and testing sets, with the former containing 1,000 records and the latter containing the rest.

```
set.seed(1234)
trainrows<-sample(
  1:dim(cast_jester_ratings)[1],1000)
validrows<-setdiff(
  1:dim(cast_jester_ratings)[1],trainrows)
train_jester<-cast_jester_ratings[trainrows,]
valid_jester<-cast_jester_ratings[validrows,]
```

Collaborative Filtering Analysis Using the Jester Dataset

- The default data class of the **recommenderlab** package is `realRatingMatrix` so we need to convert our `cast_jester_ratings` data frame into a realRatingMatrix.
- Unfortunately, there is no direct way of coercing a data frame into a realRatingMatrix class. However, the conversion can be done in two steps, first converting our data frame into a matrix, then the matrix into a RealRatingMatrix.

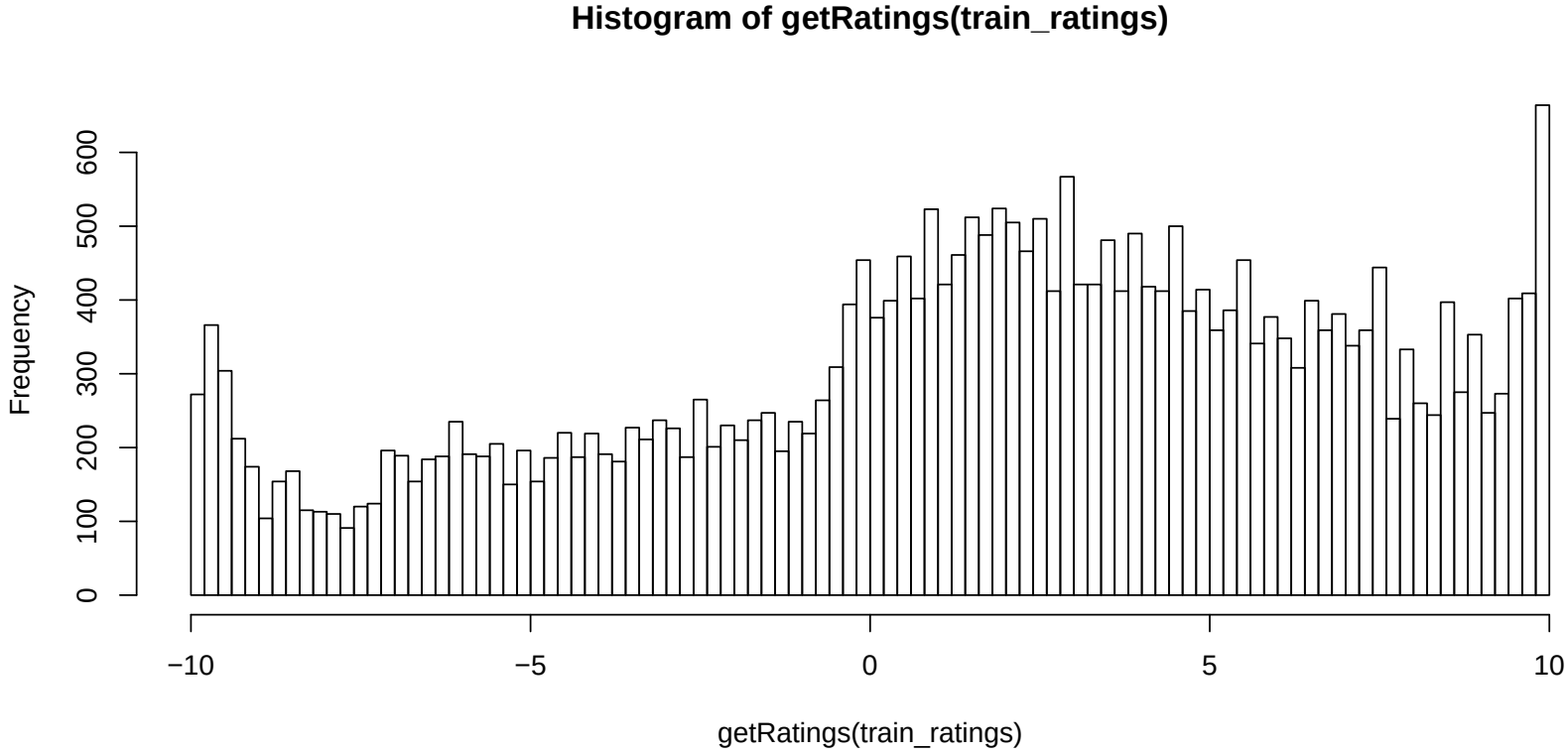
```
library(recommenderlab)
train_ratings <-as(train_jester,"matrix")
train_ratings<-as(train_ratings,"realRatingMatrix")
valid_ratings<-as(valid_jester,"matrix")
valid_ratings<-as(valid_ratings,"realRatingMatrix")
train_ratings
```

```
## 1000 x 140 rating matrix of class
'realRatingMatrix' with 30717 ratings.
```

Collaborative Filtering Analysis Using the Jester Dataset

- `getRatings()` extracts a vector with all non-missing ratings from a rating matrix.

```
hist(getRatings(train_ratings), breaks=100)
```

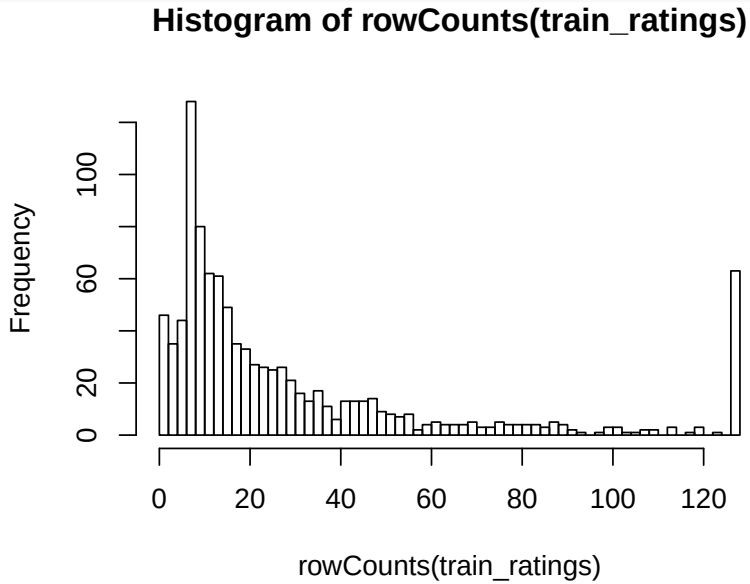


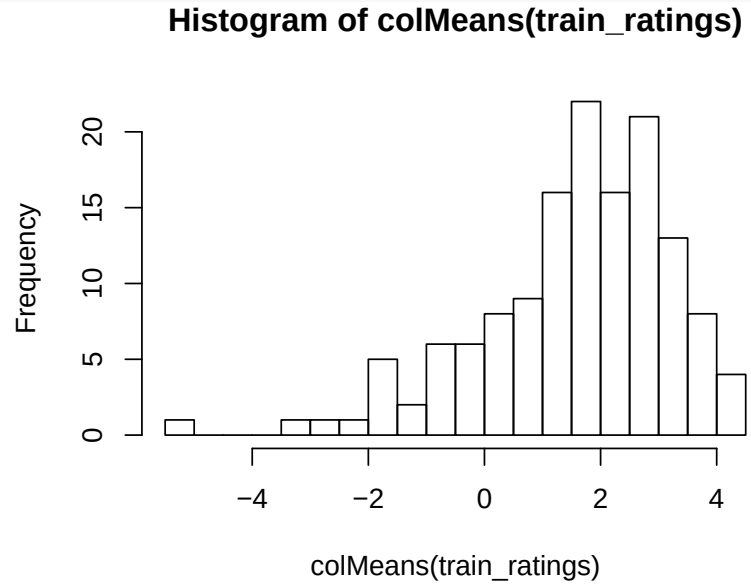
- The number of favorable ratings clearly exceed the number of unfavorable ratings. In addition, there is a concentration of extreme positive and negative ratings.

Number of Rated Jokes and Average Rating Per Joke

- There are additional visual aids that we can call.

```
par(mfrow=c(1,2))#split the plot area into two
hist(rowCounts(train_ratings), breaks=50)
hist(colMeans(train_ratings), breaks=20)
```





- The figures show that there are unusually many users who rated around 10-20 jokes and users who have rated all jokes. The distributions of average ratings per joke is negatively skewed with a mean above 0.

Creating a Recommender Using the **recommenderlab** Package

- Next, we create a recommender which generates recommendations using the user-based collaborative filtering (`UBCF`) method. A recommender is created using the creator function `Recommender()`.

```
r<- Recommender(train_ratings, method = "UBCF")
r
```

```
## Recommender of type 'UBCF' for 'realRatingMatrix'
## learned using 1000 users.
```

Creating a Recommender Using the **recommenderlab** Package

- Now using the recommender, we are ready to make recommendations to the selected users in the validation set. For the sake of this example, I am selecting user 1:

```
recom <- predict(r, valid_ratings[1,])
as(recom, "list")
```

```
## $u1
## [1] "j38" "j129" "j111" "j125" "j117" "j132"
"j56" "j148" "j113" "j90"
```

- In this case, `predict()` produced the top 10 joke recommendations for user 1.

Creating a Recommender Using the **recommenderlab** Package

- Alternatively, we can request the complete rating matrix which includes the original ratings by the user.

```
recom <- predict(r, valid_ratings[1,],
                 type="ratingMatrix")
as(recom, "matrix")[,1:10]
```

```
## j5 j7 j8 j13 j15 j16
## NA -3.994157768 -2.968970041 -1.847281679
0.007234514 -3.466944766
## j17 j18 j19 j20
## -2.934318639 -2.603052661 -0.446672585 NA
```

Creating a 'IBCF' Recommender

- The procedure for creating recommender using user-based collaborative filtering requires changing only the `method` in the `Recommender` command.

```
r<- Recommender(train_ratings, method = "IBCF")
recom <- predict(r, valid_ratings[1,])
as(recom, "list")
```

```
## $u1
## [1] "j43" "j135" "j142" "j140" "j146" "j136"
"j131" "j116" "j100" "j71"
```

- Recommendations based on different recommender systems don't have to be the same.
- One important step in developing an efficient recommender system is determining which recommender method (or which combination of methods) is the most accurate.
- Recommender accuracy assessment methods are outside of the scope of this course.

Advantages and Weaknesses of Collaborative Filtering

- **Advantages:**
  1. One big advantage of collaborative filtering is that it provides useful recommendations (even for the sparsely rated items) if our database contains a sufficient number of similar users (not necessarily many, but at least a few per user).
- **Disadvantages:**
  1. One limitation of collaborative filtering is that it cannot generate recommendations for new users, nor for new items. It is also blind to low-rated or generally unwanted (unpopular) items. Therefore, we can not expect to use it as is for detecting unwanted items.
  2. When the number of users becomes very large, collaborative filtering becomes computationally difficult. Solutions include item-based algorithms, clustering of users, and dimension reduction. The most popular dimension reduction method used in such cases is singular value decomposition (SVD), a computationally superior form of principal components analysis.

Collaborative Filtering vs. Association Rules

- **Frequent itemsets vs. personalized recommendations**
  Association rules look for frequent item combinations and will provide recommendations only for those items. In contrast, collaborative filtering provides personalized recommendations for every item (even for sparsely-rated items) thereby catering to users with unusual taste. Association-based recommendations such as Amazon's "Frequently Bought Together" display the same recommendations to all users searching for a specific item. In contrast, collaborative filtering-based methods such as Amazon's "Customers Who Bought This Item Also Bought..." generate user-specific recommendations.

Collaborative Filtering vs. Association Rules

- **Transactional data vs. user data** Association rules provide recommendations of items based on their co-purchase with other items in many transactions/baskets. In contrast, collaborative filtering provides recommendations of items based on their co-purchase or co-rating by other users. Considering distinct baskets is useful when the same items are purchased over and over again (e.g., in grocery shopping). Considering distinct users is useful when each item is typically purchased/rated once (e.g., purchases of books, music, and movies).
- **Binary data and ratings data** Association rules treat items as binary data (1 = purchase, 0 = nonpurchase), whereas collaborative filtering can operate on either binary data or on numerical ratings.

Collaborative Filtering vs. Association Rules

- **Two or more items** In association rules, the antecedent and consequent can each include one or more items (e.g., IF milk THEN cookies and cornflakes). Hence, a recommendation might be a bundle of the item of interest with multiple items ("buy milk, cookies, and cornflakes and receive 10% discount"). In contrast, in collaborative filtering, similarity is measured between pairs of items or pairs of users. A recommendation will therefore be either for a single item (the most popular item purchased by people like you, which you haven't purchased), or for multiple single items which do not necessarily relate to each other (the top two most popular items purchased by people like you, which you haven't purchased).
- In general, when considering what to recommend to a user who purchased a popular item, association rules and item-based collaborative filtering might yield the same recommendation for a single item. But a user-based recommendation will likely differ.