Project 3

Max Wagner

June 25, 2016

Libraries

```
library(reshape2)
library(dplyr)
library(recommenderlab)
library(NMF)
library(ggplot2)
```

Rec Sys

The MovieLens data was put into a smaller, more compact version that includes only user and movies with suffient information by the University of Minnesota. One of the problems with the data set is that time is not taken into consideration when looking at user's recommendations. The set provides a timestamp for each recommendation, abit in a strange format. The timestamp is in number of seconds from midnight on January 1, 1970. Using this information, it will be possible to weight ratings that are closer to present day higher than older ratings.

Data

The first step is to load in the data from the small data set, and then separate the dates into a separate data frame, and convert the seconds to years. We can then preview the movies, ratings, and dates tables to make sure they appear normal.

```
movies <- read.csv("https://raw.githubusercontent.com/maxwagner/643/master/ml-latest-small/movies.csv",
                   stringsAsFactors = FALSE)
ratings <- read.csv("https://raw.githubusercontent.com/maxwagner/643/master/ml-latest-small/ratings.csv
                    stringsAsFactors = FALSE)
```

```
movies <- read.csv("ml-latest-small/movies.csv", stringsAsFactors = FALSE)
ratings <- read.csv("ml-latest-small/ratings.csv", stringsAsFactors = FALSE)

dates <- as.vector(ratings[,4])
ratings <- ratings[,-4]

dates.year <- dates/60/60/24/365 + 1970
dates <- as.data.frame(cbind(dates,dates.year))
colnames(dates) <- c("seconds", "year")

knitr::kable(head(movies))
```

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

```
knitr::kable(head(ratings))
```

userId	movieId	rating
1	16	4.0
1	24	1.5
1	32	4.0
1	47	4.0
1	50	4.0
1	110	4.0

```
summary(dates$year)
```

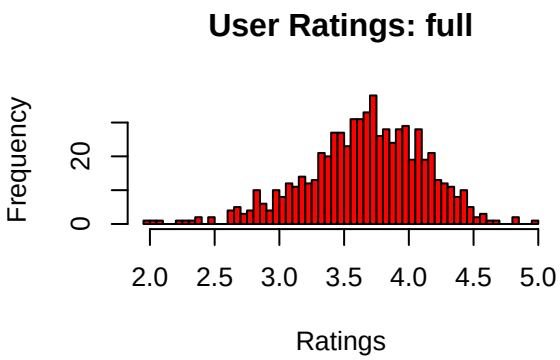
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1996    2001    2005    2006    2010    2016
```

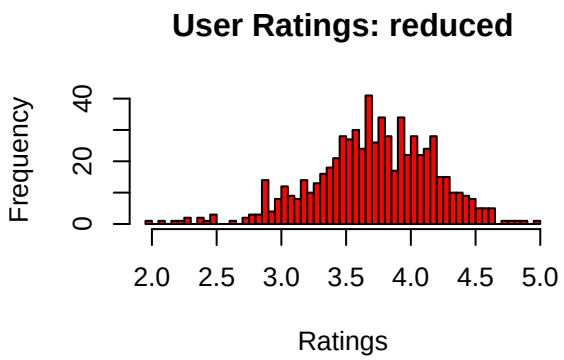
The above all seems reasonable for the information we have so far. Next, the ratings table is put into a real ratings matrix format using `recommenderlab`. In partial thanks to Sreejaya Nair and Suman K Polavarapu for the cleaner way to get to a ratings matrix with *acast*.

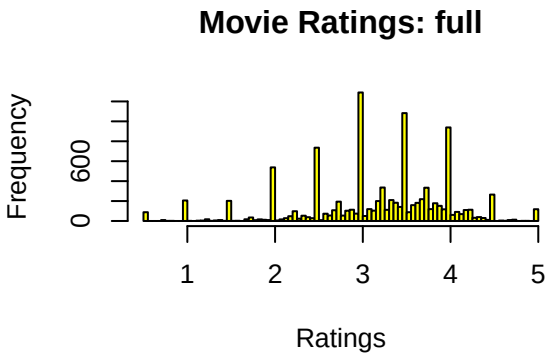
```
ratings_hori <- acast(ratings, userId ~ movieId, value.var="rating")
ratings_rm <- as(ratings_hori, "realRatingMatrix")
ratings_rm_r <- ratings_rm[, colCounts(ratings_rm) > 20]
```

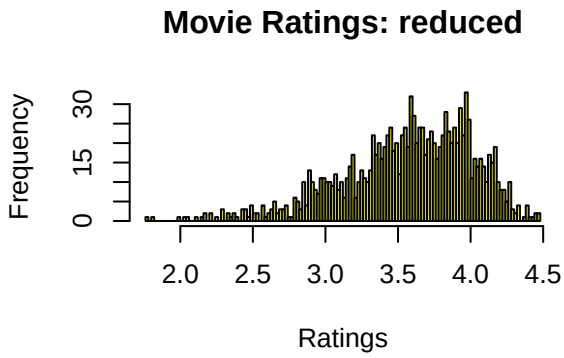
We can compare what it looks like to use all ratings, versus a smaller subset of movies that have at least 20 ratings.

```
par(mfrow=c(2,2))
hist(rowMeans(ratings_rm), breaks = 100, col = "red",
     main = "User Ratings: full", xlab = "Ratings")

hist(rowMeans(ratings_rm_r), breaks = 100, col = "red",
     main = "User Ratings: reduced", xlab = "Ratings")

hist(colMeans(ratings_rm), breaks = 100, col = "yellow",
     main = "Movie Ratings: full", xlab = "Ratings")

hist(colMeans(ratings_rm_r), breaks = 100, col = "yellow",
     main = "Movie Ratings: reduced", xlab = "Ratings")
```









The advantages of using a reduced version become much more clear with the histogram of the movie ratings. The first version, which shows all ratings, has obvious spikes on int values of 1, 2, 3, 4, and 5. This is likely from movies with low amount of ratings that have inflated results from what they actually should be rated. The second movie hisogram shows the reduced version, where only users/movies with above 20 ratings can be included. The curve is much more to what I had expected, with only a few major spikes, and more of a bell shape. Because of this, the reduced version is preferred.

UBCF Method

As a baseline, we can try out some of the methods from the last few weeks before we try the matrix factorization methods. Train with the first 400 users, and then test on a few different users. We could then match the id's given to the movies they relate to. Each column is a user, and each row is the id of a suggested movie.

```
movies.recom <- Recommender(ratings_rm_r[1:400], method = "UBCF")
movies.predict <- predict(movies.recom, ratings_rm_r[c(500, 525, 600)],n=5)
knitr::kable(as.data.frame(as(movies.predict, "list")))
```

X500	X525	X600
318	296	541
356	318	1641
593	356	1213
527	2028	345
608	913	2686

IBCF Method

Very similar to the above, but with IBCF. IBCF takes significantly longer than UBCF to run.

```
movies.recom <- Recommender(ratings_rm_r[1:400], method = "IBCF")
movies.predict <- predict(movies.recom, ratings_rm_r[c(500, 525, 600)],n=5)
knitr::kable(as.data.frame(as(movies.predict, "list")))
```

X500	X525	X600
5	36	172
7	45	372
16	141	405
36	151	707
58	198	762

Matrix Factorization Method

Using SVD for the matrix factorization method. I decided to fill in 0's for values that were NA. Couldn't find a more elegant approach. One other option is filling in by row average (user average).

```
ratings_rm_r <- normalize(ratings_rm_r)
ratings_rm_r <- as(ratings_rm_r, "matrix")
ratings_rm_r[is.na(ratings_rm_r)] <- 0
ratings_svd <- svd(ratings_rm_r)
```

Estimating k from the SVD `d` value. Then reducing the `u` and `v` matrices based on the estimated k value. Because the matrix was already reduced once based on number of ratings, I do not expect it to reduced as much as the original matrix would have. And finally, create a matrix we can use to recommend things to users.

```
d_cumul <- cumsum(ratings_svd$d) / sum(ratings_svd$d)
k <- min(which(d_cumul >= 0.8));k
```

```
## [1] 336
```

```
d <- diag(sqrt(ratings_svd$d[1:k]))
u <- ratings_svd$u[,1:k]
v <- ratings_svd$v[1:k,]

recom <- data.frame(u%*%d%*%v)
colnames(recom) <- 1:668
```

Using `recom` we can then try to decide what kinds of movies a user likes by comparing them to other users.

```
movie.recs <- data.frame(matrix(NA, nrow = 668, ncol = 668))
for(i in 1:ncol(recom)) {
   movie.recs[i,] <- colnames(recom[i,order(recom[i,],decreasing = TRUE)])
}
```

For instance if we wanted to see the user that is most similar to userid 3, we can take the first value in the third row. And then look at the movies that user likes to watch. In this case, the user most similar to user 3 is user 335, so we can take user 335's favorite movies, and show them to user 3, in hopes they enjoy them. The number of suggestions and minimum ranking can be chosen if desired.

```
user3simi <- movie.recs[3,1]
user.movies <- subset(ratings, userId == user3simi)
user.best <- head(user.movies[order(-user.movies$rating),],3)
user.best <- inner_join(user.best,movies,by = "movieId")
user.best <- user.best[,4:5]
knitr::kable(user.best)
```

title	genres
Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
Pulp Fiction (1994)	Comedy|Crime|Drama|Thriller
Lion King, The (1994)	Adventure|Animation|Children|Drama|Musical|IMAX

There are other options, such as showing which genres a user likes the most. We can use user 3 again.

```
user.movies <- subset(ratings, userId == 3 & rating >= 4)
user.movies <- inner_join(user.movies,movies,by = "movieId")
user.genres <- user.movies[,5]
user.genres.split <- unlist(strsplit(user.genres, split = "|", fixed = TRUE))
user.genres.count <- data.frame(table(user.genres.split))
knitr::kable(head(user.genres.count[order(-user.genres.count$Freq),],5))
```

	user.genres.split	Freq
8	Drama	24
5	Comedy	18
14	Romance	11
16	Thriller	9
6	Crime	7

In this case, we can see that user 3 likes to watch dramas, comedies, romance, thrillers, and action movies. This could be accurate, but could also be an effect of the database itself. We can check the overall count in the movie database to see if the ranking is similar.

```
movies.o <- movies[,3]
genres.split <- unlist(strsplit(movies.o, split = "|", fixed = TRUE))
genres.count <- data.frame(table(genres.split))
knitr::kable(head(genres.count[order(-genres.count$Freq),],10))
```

	genres.split	Freq
9	Drama	5220
6	Comedy	3515
18	Thriller	2187
16	Romance	1788
2	Action	1737

	genres.split	Freq
7	Crime	1440
3	Adventure	1164
12	Horror	1001
17	Sci-Fi	860
15	Mystery	675

If we take a look at the top genres in the whole database, we can see that the suggestions for user 3 and the overall collection end up very similar. It's an effect that should be taken into consideration when doing a content based system. Saying a user watches a lot of dramas or comedies, is not overly telling of what movies they might actually like. A more dividing system where comedies are split further into something like; romantic comedy, dark comedy, or silly comedy would be a much better indicator of what they might like.

Citations

http://econometricsense.blogspot.com/2012/10/nonnegative-matrix-factorization-and.html

http://www.r-bloggers.com/using-the-svd-to-find-the-needle-in-the-haystack/

https://github.com/srajeev1/MSDA-IS643/blob/master/projects/Project2/DATA643_Project_2.Rmd