Project 3:Evaluation of 2 recommender algorithms

Wenqing Wei(wenqing8),Wei Wang(weiw8)

Import Data

```r
library(recommenderlab)
library(reshape2)
library(ggplot2)
library(Rmisc)
library(lattice)
library(plyr)
#set seed
set.seed(5482)
#Import data
## rating data
rates = read.table('ratings.dat', sep = ":")
rates = rates[,-c(2,4,6)]#remove invalid cols
names(rates) = c("UserID","MovieID","Rating","Timestamp")#set names

## user data
users = read.table('users.dat', sep = ":")
users = users[,-c(2,4,6,8)]#remove invalid cols
names(users) = c("UserID","Gender","Age","Occupation","Zip")#set names

## movie data
movies_lines = strsplit(readLines('movies.dat'),'::',fixed = TRUE, useBytes = TRUE)
movies = data.frame(
  MovieID = numeric(),
  Title = numeric(),
  Genres = numeric()
)
for(i in 1:length(movies_lines)){
  movies[i,"MovieID"] = as.integer(movies_lines[[i]][1])
  movies[i,"Title"] = movies_lines[[i]][2]
  movies[i,"Genres"] = movies_lines[[i]][3]
}
#split genres as logical variable
genres = c("Action","Adventure","Animation","Children's","Comedy","Crime",
           "Documentary","Drama","Fantasy","Film-Noir","Horror","Musical",
           "Mystery","Romance","Sci-Fi","Thriller","War","Western")
for (genre in genres){
  movies[genre] = grepl(genre,movies$Genres, fixed=TRUE)
}
```

Model

In this project, we tried two popular recommender algorithms: item based collaborative filtering(IBCF) and user based collaborative filtering(UBCF). Moreover, we developed a simple scheme to deal with cold start problem, namely, how to predict rating for an entirely new user, who never rated any movies before, or for a new movie, which has never been rated by any users before. And based on this scheme, we developed a naive user based model. We called it 'quasi UBCF model' in this report.

Model1:UBCF

In model 1, the User-Based Collaborative Filtering is used to predict users' ratings of unrated movies. With the assumption that users with similar preference (reflected by their prior rating) may have similar rating of terms, KNN is used to gather similar users. Specifically, UBCF searches through database to group the nearest 30 neighbors with respect to their similarity, measured by Cosine Similarity, and then generates an aggregated estimated rating by averaging ratings in the neighborhood, except active user, for each unrated item.

Also, the data is normalized before being applied to the recommender. This is necessary because the user bias, which is that some parts of users tend to give high rating for all movies, while some tend to give reserve rating, may affect the averaging ratings. To extract the users' bias, the user-item matrix is centered by rows.

However, the UBCF model is easy to build but not scalable, as it need to store all users' data in memory - common drawback for KNN algorithms, and to scan the whole dataset to find neighbors can be improved by gird base algorithms. Also, the similarity assumption implies that the users' preference is constant over time, which may not be true in reality but is still the best assumption we could make.

Model2:IBCF

In model2, the Item-Based Collaborative Filtering is used to predict users' rating of unrated movies. With the assumption that a user may have preference to some specific kinds of movies so that similar movies may receive similar ratings from a user. We used Cosine similarity to measure the similarity between two items: the larger the Cosine similarity is, the more similar these two items are. To predict user i's rating for item j, we chose k (in our model, 30) items which are most similar to the item of interest and calculated a similarity weighted average of the user i's ratings for these related items as the prediction.

Still, rating of each user are centered at the very beginning to reduce user bias.

Generally, IBCF doesn't perform as well as UBCF, which is consistent with the results of our projects. And Because it only takes the active user's existing ratings into account when it trys to predict ratings, it's more likely to suffer from generalized 'cold start' problem, i.e. there is no available existing ratings so that you can't even make a prediction.

Cold Start Problem and Quasi UBCF Model

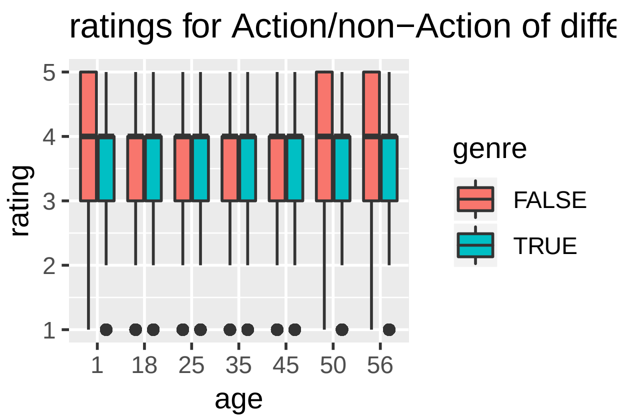
As we stated in the last section, it's possible that we can't obtain any available historical ratings. For example, for UBCF, all the similar users didn't rate the target movie ever. For IBCF, the active user never rated any similar movies to the target movie. In this case, our original models will throw out 'NA'. How to deal with this issue, or even more 'severe' issue - the active user didn't have any historical ratings or the target movie was never rated by any users before - which is also known as cold start problem.

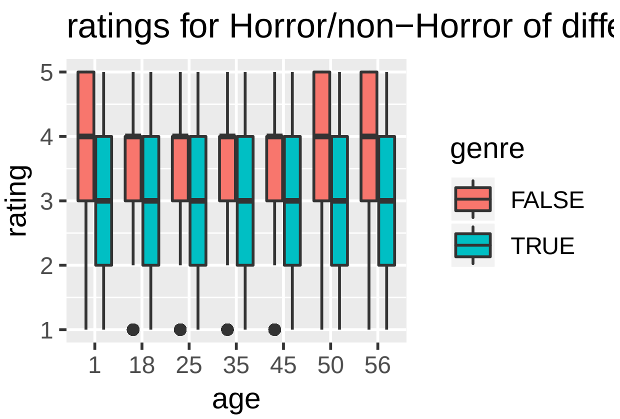
To solve this issue, we developed a scheme as the complement to our original model:

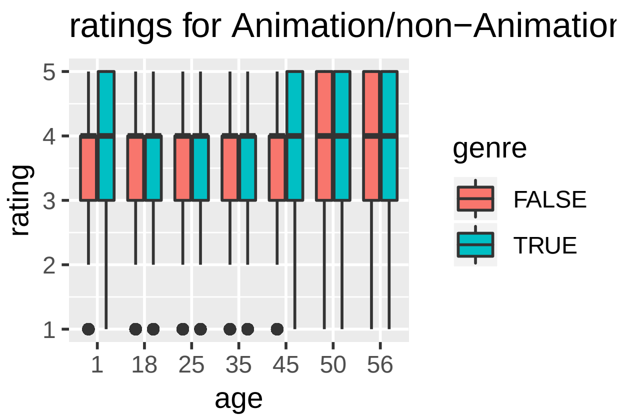
1. If the movie is not new but the original model didn't provide applicable rating prediction, we turn to the ratings of the target movie from similar users, using the average rating as the prediction. In this context, similarity is measured by users' age, occupation as well as gender. If two users are within the same age group, the same occupation group, and the same gender group, then they are similar users.

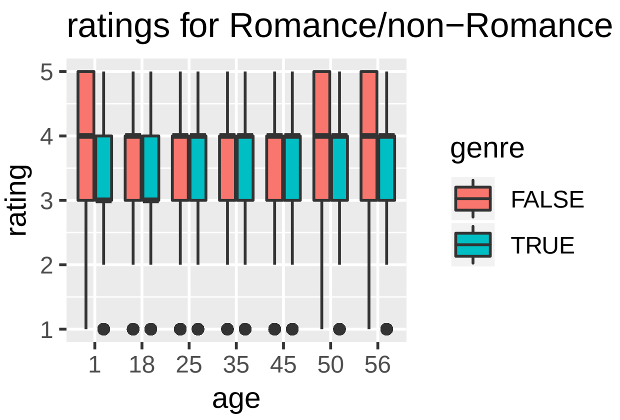
we binned age as 4 groups: under 18, 18-44,45-49, after 50.

It's because we've found that people within these 4 age groups have similar movie preference. Some selected box plots used to explore the preference to a specific movie genre of people from different age groups are as follows:









We could see that for different types of movies, rating patterns of people aged from 18 to 44 are always almost the same, showing that for differnt movie genres, they probably consistently have similar preference. Moreover, rating patterns of people aged after 50 are almost the same over these two graphs as well. Thus, to simpligy our model, to make the most of information that can be used to predict ratings, and to avoid overfitting, we binned ages into 4 groups.

```r
#function to bin age
binage <- function(x){
  if ((x==18)|(x==25)|(x==35)) return(2)
  if (x==45) return(3)
  if (x==1) return(1)
  return(4)
}
users$Age=sapply(users$Age,binage)#bin age
```

As for the occupations, because box plots indicate there are not any two occupation groups people from which consistently show similar preference to different movie genres. Lawyers and artists may both be lenient with Action movies. But when they evaluate a Romance movie, artists may be more strict. Therefore, we didn't bin occupations further.

When calculating the average rating, if the active user is not new, to reduce user bias, we compute the average of normalized ratings at first and then use the average rating of the active user to denormalize it. This denormalized average rating is our final prediction. If the active user is new, we simply take the average of ratings withou normalization as our prediction.

If no similar user has ever rated the target movie, then we compute the average of similar users' ratings for movies of the same genre as the target movie and take it as our prediction. It's possbile that similar users never rate any movies from this genre. Then, we assign 2.5.

2. If the target movie has never been rated before, then we at first turn to the average rating of active user's rating for movies of the same genre as the target movie. If it's not applicable, then we turn to the final step of the last condition.

We also established a naive Quasi UBCF Model using this scheme, which predicts ratings as we describe here.

Codes implementing Model1, Model2, this scheme as well as the Quasi UBCF Model are showed in the next section. Because it will take terribly long time to run, we set `eval=FALSE`.

Codes

```
#funciton to calculate RMSE
calrmse <- function(real,est){
  return(sqrt(mean((real-est)^2)))
}
#function to find rating for new user
findfornewuser=function(usrid,movieid,newmovie=FALSE){
  uindex=which(users$UserID==usrid)
  gender=users$Gender[uindex]
  age=users$Age[uindex]
  occup=users$Occupation[uindex]
  if (! newmovie){

  simuser=users$UserID[(users$Age==age)
                         & (users$Occupation==occup)
                         & (users$Gender==gender)]
  if(usrid %in% rownames(train_norm)){
  rate=colMeans(train_norm[simuser,movieid])
  +getNormalize(train_norm)$row$factors$means[usrid]}else{
    rate=colMeans(train_matrix[simuser,movieid])
  }

  if (is.na(rate)){
  mindex=which(movies$MovieID==movieid)
  genrecand=strsplit(movies$Genres[mindex],"\\|")[[1]]
    ind=which((querytableforcoldstart$age==age)
    &(querytableforcoldstart$occup==occup)
    &(querytableforcoldstart$genders==gender)
    & (querytableforcoldstart$genre %in% genrecand))
if (length(ind)==0){
  rate=2.5
}else{
  tmpmatrix=querytableforcoldstart[ind,c("rating","num")]
 rate=sum(tmpmatrix[,1]*tmpmatrix[,2])/sum(tmpmatrix[,2])
}}}else{
  mindex=which(movies$MovieID==movieid)
  genrecand=strsplit(movies$Genres[mindex],"\\|")[[1]]
  ind=which((querytableforcoldstart$age==age)
            &(querytableforcoldstart$occup==occup)
            &(querytableforcoldstart$genders==gender)
            & (querytableforcoldstart$genre %in% genrecand))
  if (length(ind)==0){
    rate=2.5
  }else{
    tmpmatrix=querytableforcoldstart[ind,c("rating","num")]
    rate=sum(tmpmatrix[,1]*tmpmatrix[,2])/sum(tmpmatrix[,2])
  }
}
    return(rate)
}
#function to find rate for new movie
findfornewmovie <- function(usrid,movieid){
  mindex=which(movies$MovieID==movieid)
  genrecand=strsplit(movies$Genres[mindex],"\\|")[[1]]
  if (length(genrecand)==1){
    rate=mean(mergedtrain$Rating[mergedtrain[,genrecand] & (mergedtrain$UserID==usrid)])
  }else{
    rate=mean(mergedtrain$Rating[apply(mergedtrain[,genrecand],1,sum)>0
                                 & (mergedtrain$UserID==usrid)])}
  if (is.na(rate)){
    rate=findfornewusr(usrid,movieid,newmovie=TRUE)
  }
  return(rate)
}

#split train and test #it's possible that some movies or users in test doesn't appear in train,
which is called cold-start problem in real-world.
#using other methods to estimate ratings for these cases.
sample_idx = sample(1:length(rates$UserID),floor(length(rates$UserID)*0.6))
train = rates[sort(sample_idx),-4]
test = rates[-sample_idx,-4]
test_idx = sample(1:length(test$UserID),floor(length(test$UserID)*0.5))
test = test[sort(test_idx),]

#create realRatingMatrix
train_matrix = acast(train, UserID ~ MovieID, value.var = "Rating", na.rm=FALSE)
mergedtrain=merge(x=train,y=users,by.x="UserID")
mergedtrain=merge(x=mergedtrain,y=movies,by.x="MovieID")

#bin age because some ages seem like have the same preference as the others'
mergedtrain$Age=sapply(mergedtrain$Age,binage)

#generate querytable used to find rate for new users
querytableforcoldstart=data.frame("age"=numeric(),"occup"=numeric(),
                                  "genders"=character(),"genre"=character(),"rating"=numeric(),
                                  "num"=numeric(),stringsAsFactors = FALSE)
ages=unique(mergedtrain$Age)
occups=unique(mergedtrain$Occupation)
genders=unique(mergedtrain$Gender)
i=1
for (age in ages){
  for (occup in occups){
    for (gender in genders){
      for (genre in genres){
        querytableforcoldstart[i,"age"]=age
        querytableforcoldstart[i,"occup"]=occup
        querytableforcoldstart[i,"genders"]=gender
        querytableforcoldstart[i,"genre"]=genre
        rate=mergedtrain$Rating[(mergedtrain$Age==age)
                                & (mergedtrain$Occupation==occup)
                                & (mergedtrain$Gender==gender)
                                & mergedtrain[,genre]]
        querytableforcoldstart[i,"rating"]=mean(rate)
        querytableforcoldstart[i,"num"]=length(rate)
        i=i+1
      }
    }
  }
}
querytableforcoldstart=querytableforcoldstart[querytableforcoldstart$num!=0,] #remove NA


train_matrix = as(train_matrix, "realRatingMatrix")
train_norm=normalize(train_matrix) #center rating by row

#MODEL1
recommender_model = Recommender(train_matrix, method = "UBCF",
                                param=list(method="Cosine",nn=30))
recom = predict(recommender_model, train_matrix, type = 'ratings')
recom_matrix= as(recom, "matrix")#convert recommenderlab object to readable matrix

#MODEL2
recommender_model2=Recommender(train_matrix,method='IBCF')
recom2=predict(recommender_model2,train_matrix,type='ratings')
recom_matrix2=as(recom2,"matrix")
test$est = NA #init rates as NA
test$est2=NA#inits rate as NA for model2

for (u in 1:nrow(test)){

  # Read userid and movieid from columns 2 and 3 of test data
  userid =as.character (test$UserID[u])
  movieid =as.character (test$MovieID[u])
  notnewuser=userid %in% rownames(recom)
  notnewmovie=movieid %in% colnames(recom)
  if (notnewuser & notnewmovie ){
  rating = recom_matrix[userid,movieid]
  if (is.na(rating)){
rating=findfornewuser(userid,movieid)
  }
  test$est[u]=rating}else{
    if (notnewuser){
    test$est[u]=findfornewmovie(userid,movieid)}else{
      if(notnewmovie){
    test$est[u]=findfornewuser(userid,movieid)}else{
      test$est[u]=findfornewuser(userid,movieid,newmovie=TRUE)
    }
```

```r
    }
  }
}
calrmse(test$Rating,test$est)#performance of model 1

for (u in 1:nrow(test)){

  # Read userid and movieid from columns 2 and 3 of test data
  userid =as.character (test$UserID[u])
  movieid =as.character (test$MovieID[u])
  notnewuser=userid %in% rownames(recom)
  notnewmovie=movieid %in% colnames(recom)
  if (notnewuser & notnewmovie ){
    rating = recom_matrix2[userid,movieid]
    if (is.na(rating)){
      rating=findfornewuser(userid,movieid)
    }
    test$est2[u]=rating}else{
      if (notnewuser){
        test$est2[u]=findfornewmovie(userid,movieid)}else{
          if(notnewmovie){
            test$est2[u]=findfornewuser(userid,movieid)}else{
              test$est2[u]=findfornewuser(userid,movieid,newmovie=TRUE)
            }
        }
    }
}

calrmse(test$Rating,test$est2)#performance of model2

##just for fun:Quasi UBCF model
test$est3=NA
for (u in 1:nrow(test)){

  userid =as.character (test$UserID[u])
  movieid =as.character (test$MovieID[u])
  notnewuser=userid %in% rownames(recom)
  notnewmovie=movieid %in% colnames(recom)
  if (notnewuser & notnewmovie ){

      rating=findfornewuser(userid,movieid)
    test$est3[u]=rating}else{
      if (notnewuser){
        test$est3[u]=findfornewmovie(userid,movieid)}else{
          if(notnewmovie){
            test$est3[u]=findfornewuser(userid,movieid)}else{
              test$est3[u]=findfornewuser(userid,movieid,newmovie=TRUE)
            }
        }
    }
}

calrmse(test$Rating,test$est3) #performance of Quasi UBCF model
```

Results

	RMSE of Different Models
UBCF	1.033461
IBCF	1.110966
Quasi UBCF	1.043468

We could see that our quasi UBCF model also performs better than the basic IBCF model. The user based model always performs over the item based model. These results are consistent with common sense.