MOVIE RATING PREDICTION

EST 508 FINAL PROJECT II

XINGYU BU 110679523

EST 508 FINAL PROJECT

Description

This is a project analyzing the movie and user data provided by Kaggle and predict some users' ratings on some movies. There are more than 6000 movies data which includes their id, issued year and types. There are more than 2000 users only get some basic personal information, the other 4000 users we also have their ratings on some movies.

The evaluation on Kaggle is: For each user, scoring metric will select the 5% of movies that would be most highly rated by that user as predicted. It then looks at the actual ratings (in the test data) that the user gave those movies. The score is the average of those ratings.

Thus, for an algorithm to score well, it only needs to identify which movies a user is likely to rate most highly (so the absolute accuracy of your ratings is less important than the rank ordering).

Method

In this project I used three statistical methods.

The first method is called Slope one method[1], which is popular used in movie recommendation today. Its principal is: If a group of persons have seen same movies with the predicted man, and they also rated the predicted movie, then we can judge the predicted rating by the difference between the group and predicted man's ratings on other movies.

	Predicted Man	Other Persons
Predicted Movie	2.5	3
Other Movies	4	4.5

For example, if the predicted man's average ratings on other movies is 4, other people's is 4.5 and other person's average on predicted movie is 3, then we can predict the predicted rating is 3 – (4.5 - 4) = 2.5. Such method is convenient to calculate. However, it has some weakness, firstly it need a large volume of data, secondly the predicted man must have seen other movies, which means he should be an old user.

The second method is K-means. We group movies and users into several groups. Assuming that a group of persons will give almost same rates on a group of movies. Thus what we only need to do is build a map, which can tell us the rating of one group of persons

[1] Daniel Lemire, Anna Maclachlan, Slope One Predictors for Online Rating-Based Collaborative Filtering, In SIAM Data Mining (SDM'05), Newport Beach, California, April 21–23, 2005.

grading on one group of movies. Then we classify the predicted man and movie are on which groups to decide the rating.

The third method is more simple, since the evaluation only select the 5% best rated movie of one user to consider, which in this sample means each person only choose 2-3 movies. We can assume the high rated movies are popular in among all users, so for one predicted man's predicted movies, we just choose 3 that most popular and give them the highest ratings. Actually this method is quickly than two method before and get a higher score!

Data Source

The data is in:

https://inclass.kaggle.com/c/movie-recommendation/data

Coding(R)

```
#For "new users", we find out their groups by same age, occupation, gender and first_zipcode
group_new_user <- function(userinfo,users2){
  sim_users = list()
  for (i in users2[2783:6040,1]){
    if (users2[i,2] == userinfo[2]){
      sim_users = append(sim_users, users2[i,1])}}
  return(sim_users)}
#For old users, find sinilar group by K-means of occupation.... and similar movie ratings
similar_old_users <- function(MU_mat, groups){
  kold_users <- kmeans(MU_mat, groups,iter.max=10000, algorithm="MacQueen")
  mat <- cbind(c(2783:6040),row.names(fitted(kold_users)))
  groups_old <- matrix(NA,groups,table(mat[,2])[which.max(table(mat[,2]))])
  for (i in 1:length(mat[,1])){
    g <- as.integer(mat[i,2])
    groups_old[g,which(is.na(groups_old[g,]))[1]] <- as.integer(mat[i,1])}
  return(groups_old)}
```

```
#All movies that have been rated by a group of users with their id, total and times.
movies_rated <- function(sim_users, training2){
  raw <- list()
  for (i in 1:length(data[,1])){
    if (data[i,1] %in% sim_users){
      raw = rbind(raw, data[i,][1:3])}}
  dup <- !duplicated(raw[,2])
  movieid <- raw[dup,][,2]
  mat <- matrix(0, nrow = length(movieid), ncol = 3)
  mat[,1] <- movieid
  matchtable <- list()
  for (i in 1:length(movieid)){
    matchtable[movieid[i]] <- i}
  for (i in 1:length(raw[,1])){
    p = as.integer(matchtable[raw[i,2]])
    mat[p,2] = mat[p,2] + raw[i,3]
    mat[p,3] = mat[p,3] + 1}
  return(mat)}
average_rating <- function(data){
  average <- colMeans(data)
  movieid <- data[list,2]
  mat <- matrix(0, nrow = length(movieid),ncol = 4)
  mat[,1] <- movieid
  for (i in 1:length(data[,1])){
    p = which(mat[,1] == data[i,2])
    mat[p,2] <- mat[p,2] + data[i,3]
    mat[p,3] <- mat[p,3] + 1}
  mat[,4] <- mat[,2] / mat[,3]
  return(cbind(mat[,1],mat[,4]))
}
```

```
#divide movies into several groups by kmeans
similar_movies <- function(movies, groups){
  row.names(movies) <- movies[,1]
  movies[,2] <- movies[,2]/1000
  movies <- movies[,-1]
  kmovies <- kmeans(movies, groups,iter.max=15000, algorithm="MacQueen")
  mat <- cbind(row.names(movies),row.names(fitted(kmovies)))
  groups_movies <- matrix(NA,groups,table(mat[,2])[which.max(table(mat[,2]))])
  for (i in 1:length(mat[,1])){
    g <- as.integer(mat[i,2])
    groups_movies[g,which(is.na(groups_movies[g,]))[1]] <- as.integer(mat[i,1])
  }
  return(groups_movies)
}
MU_mean <- function(user, movie, training2,movies_match){
  rate = 0
  n = 0
  for (k in user) {
    for (m in movie){
      g <- unlist(movies_match[m])
      if(!is.na(training2[k,g])){
        rate = rate + training2[k,g]
        n = n + 1
      }
    }
  }
  if (n != 0){
    return(rate/n)
  }else{
   return(3.6031)
  }
}
```

```
k_mean_map <- function(groups_old,groups_movies,training2,movies_match){
  mat <- matrix(3.60,length(groups_old[,1]),length(groups_movies[,1]))
  for (i in 1:length(groups_old[,1])){
    for(j in 1:length(groups_movies[,1])){
      user <- groups_old[i,];user <- user[!is.na(user)];movie <- groups_movies[j,];movie <- movie[!is.na(movie)]
      mat[i,j] <- MU_mean(user,movie,training2,movies_match)
      print(paste(i,j,':',mat[i,j]))}}
  return(mat)}
ratingSlopeOne <- function(uid, mid, groups_movies, groups_old,training2,mapk,movies_match,users2){
  lo <- length(groups_old[,1])
  lm <- length(groups_movies[,1])
  if (uid < 2783){
    userinfo <- users2[uid,]
    group <- group_new_user(userinfo,users2)
    a <- training2[unlist(group),unlist(movies_match[mid])]
    a <- a[!is.na(a)]
    if (length(a) > 5){
      rate <- mean(a);print(paste('<2783, seen, >5:',uid,mid,rate));return(rate)}else{
      g <- match(mid,groups_movies)%%lm
      if (g == 0){
        g <- lm}
      simovies <- groups_movies[g,];simovies <- simovies[!is.na(simovies)]
      rate <- MU_mean(group,simovies,training2,movies_match)
      print(paste('<2783, seen,<5:',uid,mid,rate))
      return(rate)}}else{
    g <- match(uid,groups_old)%%lo
    if(g == 0){ g <- lo}
    group <- groups_old[g,];group <- group[!is.na(group)]
    group <- unlist(group);mid_group <- training2[group,unlist(movies_match[mid])]
    l <- length(mid_group[is.na(mid_group)])
    if (length(mid_group) != l){
      m <- mean(mid_group,na.rm = TRUE)
      a <- training2[uid,]
```

```
uid_seen <- which(!is.na(a))
if (length(uid_seen)==0){
  userinfo <- users2[uid,]
  group <- group_new_user(userinfo,users2)
  a <- training2[unlist(group),unlist(movies_match[mid])]
  a <- a[!is.na(a)]
  if (length(a) > 5){rate <- mean(a)
    print(paste('<2783, seen, >5:',uid,mid,rate))
    return(rate)}else{
    g <- match(mid,groups_movies)%%lm
    if (g == 0){g <- lm }
    simovies <- groups_movies[g,];simovies <- simovies[!is.na(simovies)]
    rate <- MU_mean(group,simovies,training2,movies_match)
    print(paste('<2783, seen,<5:',uid,mid,rate))
    return(rate)}}
uid_score <- training2[uid,uid_seen];group_seen <- training2[group,uid_seen]
group_n <- list();group_score <- list()
for (i in 1:length(group_seen[1,])){
  a <- group_seen[,i];a <- a[!is.na(a)];group_n[i] <- length(a)
  if (length(a) != 0){
    group_score[i] <- sum(a)}else{group_score[i] <- 0}}
mat <- matrix(as.numeric(c(group_score,group_n,uid_score)),length(group_score))
delta <- sum(mat[,1]-mat[,2]*mat[,3]) / sum(mat[,2])
rate <- m - delta
print(paste('>2783, slopeone:', uid,mid,rate))
return(rate)}else{k <- match(uid,groups_old)%%lo
if(k == 0){k <- lo}
m <- match(mid,groups_movies)%%lm
if(m == 0){m <- lm}
rate <- mapk[k,m]
print(paste('>2783, kmeans:',rate))
return(rate)}}}
```

```
main<- function(){
 rm(list=ls())
 setwd("C:/Users/Vodka/Documents/SBU/EST 508/Projects/Movies Rating")
 users <- read.csv(file.choose());ratings <- read.csv(file.choose()),movies <- read.csv(file.choose())
 testing <- read.csv(file.choose())
 #construct movies matchtable
 movies_match <- list()
 for (i in 1:length(movies[,1])){movies_match[movies[i,1]] <- i }
 #Transform training data
 training2 <- matrix(NA,length(users[,1]),length(movies[,1]))
 for (i in 1:length(ratings[,1])){training2[ratings[i,1],unlist(movies_match[ratings[i,2]])] <- ratings[i,3]}
 #add average into movies
 average <- colMeans(training2,na.rm = TRUE)
 m_training2 <- mean(training2,na.rm = TRUE)
 average[is.nan(average)] <- m_training2 ;distinct <- list()
 for (i in 1:length(average)){if (average[i] > 4){ distinct[i] <- 5}else{ distinct[i] <- 0}}
 movies <- cbind(movies, as.matrix(distinct))
 groups_movies <- similar_movies(movies,groups = 70)
 #construct a movie_user mat, for dividing old users
 old_uid <- c(2783:6040)
 for (i in 1:length(groups_movies[,1])) {print(i);movie <- groups_movies[i,];movie <- movie[!is.na(movie)]
  group_rate <- training2[2783:6040,unlist(movies_match[movie])]
  if (length(group_rate) == 3258){group_mean <- group_rate}else{
   group_mean <- rowMeans(group_rate,na.rm = TRUE)}
  if (i != 1){MU_mat <- cbind(MU_mat,group_mean)}else{
   MU_mat <- group_mean}}
 MU_mat[is.nan(MU_mat)] <- m_training2
 MU_mat[is.na(MU_mat)] <- m_training2
  otherinf <- users[2783:6040,2:5]/c(1/10,10,4,2);MU_mat <- cbind(MU_mat, otherinf)
 colnames(MU_mat) <- append(c(1:70),c('Gender','Age','Occupation','First_ZipCode'))
 groups_old <- similar_old_users(MU_mat,groups = 75)
 mapk <- k_mean_map(groups_old,groups_movies,training2,movies_match)
 users2 <- cbind(users[,1],users[,2]*10000+users[,3]*100+users[,4])
 ratingSlopeOne(uid,mid,groups_movies,groups_old,training2,mapk,movies_match,users2)}
```

```
ratingkmap <- function(uid, mid, groups_movies, groups_old,training2,mapk,movies_match,users2){
  lo <- length(groups_old[,1])
  lm <- length(groups_movies[,1])
  if (uid < 2783){
    userinfo <- users2[uid,]
    group <- group_new_user(userinfo,users2)
    a <- training2[unlist(group),unlist(movies_match[mid])]
    a <- a[!is.na(a)]
    if (length(a) > 5){
      rate <- mean(a)
      print(paste(uid,mid,rate))
      return(rate)
    }else{
      g <- match(mid,groups_movies)%%lm
      if (g == 0){
        g <- lm}
      simovies <- groups_movies[g,]
      simovies <- simovies[!is.na(simovies)]
      rate <- MU_mean(group,simovies,training2,movies_match)
      print(paste(uid,mid,rate))
      return(rate)}
  }else{
    k <- match(uid,groups_old)%%lo
    if(k == 0){
      k <- lo }
    m <- match(mid,groups_movies)%%lm
    if(m == 0){
      m <- lm}
    rate <- mapk[k,m]
    print(paste(uid,mid,rate))
    return(rate)
    }
`
```

```
choose <- function(a){
  n <- length(a[!is.na(a)])
  if (n < 10){
    return(3.60)
  }else{
    return(mean(a,na.rm = TRUE))
  }
}
total <- apply(training2,2,choose)


testing <- as.matrix(testing)
for (i in 1:length(testing[,1])){
  a <- testing[i,3]
  b <- strsplit(a,'_')
  mid <- as.integer(b$id[2])
  testing[i,2] <- total[unlist(movies_match[mid])]
}
print('Writing csv')
write.csv(testing,"testing5.csv",row.names = FALSE)
```

Results

I submitted my testing online, the results are as below:

Method 1 (using Slope One. Time using: more than 1 hour):

#	Δrank	Team Name	Score	Entries	Last Submission UTC (Best – Last Submission)	
1	—	Mary Petik	4.32963	2	Thu, 21 May 2015 20:27:47 (-0.1h)	
2	—	G Scott	4.19219	2	Thu, 21 May 2015 19:52:49	
3	—	Kush_Greg_Noj_Lav	4.18270	1	Thu, 21 May 2015 23:04:33	
4	—	Jon & Vikram	4.13432	1	Thu, 21 May 2015 23:52:52	
-		**XingyuBu**	**4.12390**	.	**Sat, 07 May 2016 20:34:54**	**Post-Deadline**

Method 2 (using Kmap. Time using: 5 min)：

#	Δrank	Team Name	Score	Entries	Last Submission UTC (Best – Last Submission)	
1	—	Mary Petik	4.32963	2	Thu, 21 May 2015 20:27:47 (-0.1h)	
2	—	G Scott	4.19219	2	Thu, 21 May 2015 19:52:49	
3	—	Kush_Greg_Noj_Lav	4.18270	1	Thu, 21 May 2015 23:04:33	
-		**XingyuBu**	**4.17122**	.	**Sat, 07 May 2016 21:02:01**	**Post-Deadline**
Post-Deadline Entry If you would have submitted this entry during the competition, you would have been around here on the leaderboard.						
4	—	Jon & Vikram	4.13432	1	Thu, 21 May 2015 23:52:52	

Method 3 (choosing popular movies. Time using: 1 min)：

#	Δrank	Team Name	Score	Entries	Last Submission UTC (Best – Last Submission)	
1	—	Mary Petik	4.32963	2	Thu, 21 May 2015 20:27:47 (-0.1h)	
-		**XingyuBu**	**4.30767**	.	**Sat, 07 May 2016 21:03:12**	**Post-Deadline**
Post-Deadline Entry If you would have submitted this entry during the competition, you would have been around here on the leaderboard.						
2	—	G Scott	4.19219	2	Thu, 21 May 2015 19:52:49	
3	—	Kush_Greg_Noj_Lav	4.18270	1	Thu, 21 May 2015 23:04:33	
4	—	Jon & Vikram	4.13432	1	Thu, 21 May 2015 23:52:52	