Nicolas Kardous
IND 242 HW5
Paul Grigas

Homework Assignment #5

Part a)

There are 807 songs in the dataset. There are 2421 users. The range of values that the ratings take on are 1 to 3.432969.

```
Songs = read.csv("Songs.csv")
MusicRatings = read.csv("MusicRatings.csv")
Users = read.csv("Users.csv")

min(data_MusicRatings$rating)
max(data_MusicRatings$rating)

# a)

set.seed(345)

train.ids <- sample(nrow(MusicRatings), 0.84*nrow(MusicRatings))
train.mr <- MusicRatings[train.ids,]
test <- MusicRatings[-train.ids,]

# split testing into real testing and validation set

test.ids <- sample(nrow(test), 0.5*nrow(test))
test.mr <- test[test.ids,]
validation <- test[-test.ids,]

# Split validation into validation A and validation B

val.ids <- sample(nrow(validation), 0.5*nrow(validation))
validationA <- validation[val.ids,]
validationB <- validation[-val.ids,]
```

Part b)

i)

In the dataset, the parameters included in model 1, is the sum of the alpha terms and the beta terms. There are 2421 alpha terms and 807 beta terms, thus the total number of parameters is 2421+807 = 3228. From our training set, we have 243,103 observations to train the model with.

ii)

We look for the songs that have the highest beta values. These are listed below.

songID	songName	Artist	Beta value
54	You're the One	Dwight Yoakam	1.709812
26	Undo	Bjork	1.691173
439	Secrets	OneRepublic	1.644809

Our answer relates to model 1 because model 1 is the equation, $X_{i,j} = \alpha_i + \beta_j + \varepsilon_{i,j}$, and our answer gives the value of beta. Because Xij is the value of the rating, the higher the beta value gives us a higher rating. Thus the answer we have computed the beta values, and the highest beta values will give us the highest Xij in the model.

```
# ii)

mat.train.centered <- biScale(mat.train, maxit = 10000, row.scale = FALSE, col.scale = FALSE)
# mat.train.centered is X_ij - alpha_i - beta_j
alpha <- attr(mat.train.centered, "biScale:row")$center
beta <- attr(mat.train.centered, "biScale:column")$center

Users$alpha <- alpha
Songs$beta <- beta

X_ij = matrix(nrow=length(alpha),ncol=length(beta))

for(row in 1:length(alpha)) {
  for(col in 1:length(beta)) {
    X_ij[row, col]=alpha[row]+beta[col]
  }
}

Decreasing_Song <- sort(Songs$beta, decreasing = TRUE)
Decreasing_Song[1]
Decreasing_Song[2]
Decreasing_Song[3]
```

iii)

The users listed below are the users that are most enthused about the songs.

User ID	Alpha value
1540	0.5951995
838	0.5011018
1569	0.4867033

```
#iii)

Decreasing_Users <- sort(Users$alpha, decreasing = TRUE)
Decreasing_Users[1]
Decreasing_Users[2]
Decreasing_Users[3]
```

iv)

Out of sample performance of the fitted model on the previously constructed test set is shown in the table below.

MAE	0.1805888758
RMSE	0.23576713869
OSR^2	0.27859227500

```
#iv)

OSR2 <- function(predictions, train, test) {
  SSE <- sum((test - predictions)^2)
  SST <- sum((test - mean(train))^2)
  r2 <- 1 - SSE/SST
  return(r2)
}

X_ij = Users$alpha[test.mr$userID] + Songs$beta[test.mr$songID]

Partb_MAE <- mean(abs(X_ij - test.mr$rating))
Partb_RMSE <- sqrt(mean((X_ij - test.mr$rating)^2))
Partb_OSR2 <- OSR2(X_ij, train.mr$rating, test.mr$rating)
```

Part c)

i)

In the dataset, the parameters included in model 2, is the archetype value k multiply by the alpha and beta terms, plus the sum of the alpha and beta terms. This shown as an equation is k(a+b)+(a+b). There are 2421 alpha terms and 807 beta terms, thus the total number of parameters is k(2421+807)+(2421+807) = 3228k + 3228 = 3228(k+1). From our training set, we have 243,103 observations to train the model with. In part iii), we found k to be 4, thus 3228(4+1) = 16140

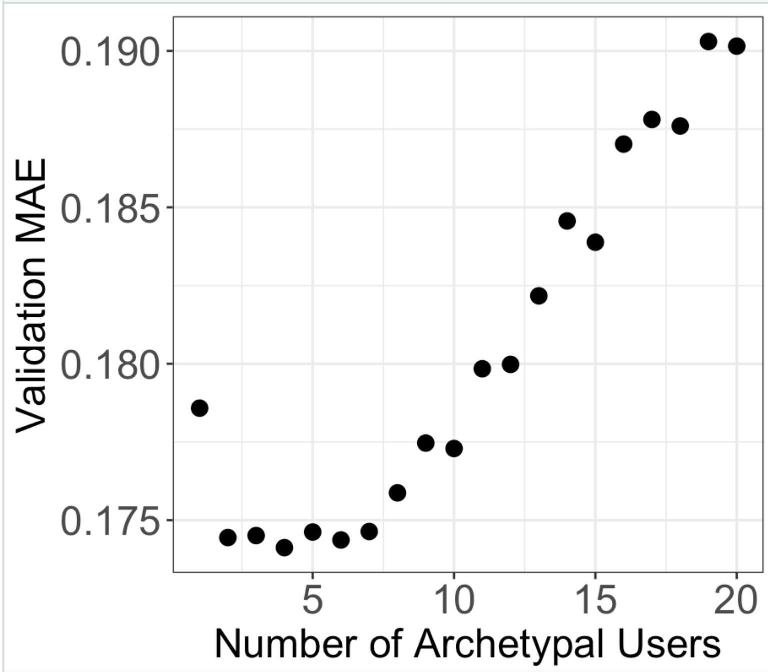
ii)

```
# ii)

# compute validation set MAE for rank = 1,2,...,20
mae.vals.cii = rep(NA, 20)
for (rnk in seq_len(20)) {
  print(str_c("Trying rank.max = ", rnk))
  mod.cii <- softImpute(mat.train.centered, rank.max = rnk, lambda = 0, maxit = 1000)
  preds.cii <- impute(mod.cii, validationA$userID, validationA$songID) %>% pmin(5) %>% pmax(1)
  mae.vals.cii[rnk] <- mean(abs(preds.cii - validationA$rating))
}

mae.val.cii.df <- data.frame(rnk = seq_len(20), mae = mae.vals.cii)
ggplot(mae.val.cii.df, aes(x = rnk, y = mae)) + geom_point(size = 3) +
  ylab("Validation MAE") + xlab("Number of Archetypal Users") +
  theme_bw() + theme(axis.title=element_text(size=18), axis.text=element_text(size=18))
```

Looking at the plot we have above, we see the lowest validation MAE we get is at a value of 4. Thus, the number of archetypal users, and the value of k is equal 4.




```
> mae.vals.cii
 [1] 0.1785801 0.1744447 0.1745082 0.1741219 0.1746208 0.1743671 0.1746373 0.1758732 0.1774647
[10] 0.1772902 0.1798396 0.1799789 0.1821709 0.1845643 0.1838845 0.1870210 0.1878090 0.1876005
[19] 0.1903004 0.1901530
> min(mae.vals.cii)
[1] 0.1741219
```

When we compute the minimum MAE value, we get 0.1741219. From this, we are selecting our number of archetypes to be 4.

iii)

```
# iii)

# choose k = 4

mod.ciii.final <- softImpute(mat.train.centered, rank.max = 4, lambda = 0, maxit = 1000)
preds.ciii.final <- impute(mod.ciii.final, test$userID, test$songID) %>% pmin(5) %>% pmax(1)

Partc_MAE <- mean(abs(preds.ciii.final - test.mr$rating))
Partc_RMSE <- sqrt(mean((preds.ciii.final - test.mr$rating)^2))
Partc_OSR2 <- OSR2(preds.ciii.final, train.mr$rating, test.mr$rating)
```

Out of sample performance of the fitted model on the previously constructed test set is shown in the table below.

MAE	0.25282208
RMSE	0.33805224
OSR^2	-1.9662782495

Part d)

i)

We first create a new copy of our training and testing set. On our new copies, we use the inner_join function to join our training and testing set to the "Songs" data frame. This way, for our training and testing set, for each songID, we have a year and genre associated with it. We also make year and genre into factors/categorical variable. This is shown in the code below.

```
# Part d)

# i)

# Create a copy of training and testing set, and
# add genre and year to the training set and testing set

train.mr.new <- train.mr
train.mr.new <- inner_join(train.mr.new,Songs,by='songID')

test.mr.new <- test.mr
test.mr.new <- inner_join(test.mr.new,Songs,by='songID')

# Treat as factors/categorical variables

train.mr.new$year = as.factor(train.mr.new$year)
train.mr.new$genre = as.factor(train.mr.new$genre)
```

The first model that we implement is a linear regression model. This is shown in the code below.

```
# Train Linear Regression model

LR <- lm(rating ~ year + genre, data=train.mr.new)
summary(LR)
LRPredictions <- predict(LR, newdata=test.mr.new)

Partd_LR_MAE <- mean(abs(LRPredictions - test.mr.new$rating))
Partd_LR_RMSE <- sqrt(mean((LRPredictions - test.mr.new$rating)^2))
Partd_LR_OSR2 <- OSR2(LRPredictions, train.mr.new$rating, test.mr.new$rating)
```

Out of sample performance of the fitted model on the previously constructed test set is shown in the table below.

MAE	0.224412789
RMSE	0.273162627
OSR^2	0.0315954868

The next model we train is a random forest model. This is shown in the code below.

```
# Train Random Forest model

RF = randomForest(rating ~ year + genre, data=train.mr.new)
RFPredictions = predict(RF, newdata = test.mr.new)
table(test.mr.new$rating, RFPredictions)

Partd_RF_MAE <- mean(abs(RFPredictions - test.mr.new$rating))
Partd_RF_RMSE <- sqrt(mean((RFPredictions - test.mr.new$rating)^2))
Partd_RF_OSR2 <- OSR2(RFPredictions, train.mr.new$rating, test.mr.new$rating)
```

MAE	0.2236007665
RMSE	0.2715465625
OSR^2	0.043019999

ii)

To perform blending of collaborative filtering model (2), linear regression model and random forest model is shown in the code below.

```r
# ii)

validationB <- inner_join(validationB,Songs,by='songID')
validationB$year = as.factor(validationB$year)
validationB$genre = as.factor(validationB$genre)

# Blending

val.preds.cf <- impute(mod.ciii.final, validationB$userID, validationB$songID)
val.preds.lm <- predict(LR, newdata = validationB)
val.preds.rf <- predict(RF, newdata = validationB)

# Build validation set data frame
val.blending_df = data.frame(rating = validationB$rating, cf_preds = val.preds.cf,
                             lm_preds = val.preds.lm, rf_preds = val.preds.rf)

# Train blended model
blend.mod = lm(rating ~ . -1, data = val.blending_df)
summary(blend.mod)
# Test blended model
test.blending_df = data.frame(rating = test.mr.new$rating, cf_preds = preds.ciii.final,
                              lm_preds = LRPredictions, rf_preds = RFPredictions)

test.preds.blend <- predict(blend.mod, newdata = test.blending_df)

Partd_blend_MAE <- mean(abs(test.preds.blend - test.mr.new$rating))
Partd_blend_RMSE <- sqrt(mean((test.preds.blend - test.mr.new$rating)^2))
Partd_blend_OSR2 <- OSR2(test.preds.blend, train.mr.new$rating, test.mr.new$rating)
```

Test set MAE, RMSE and OSR^2 is shown in the table below.

MAE	0.24346232
RMSE	0.3143511587
OSR^2	-1.5649232209

The additional features associated with songs do add predictive power on top of the collaborative filtering model because our MAE and RMSE values have decreased.

Code:

```
# IND242HW5
# Nicolas Kardous

library(tm)
library(SnowballC)
library(wordcloud)
library(MASS)
library(caTools)
library(dplyr)
library(rpart)
library(rpart.plot)
library(randomForest)
library(caret)
library(tm.plugin.webmining)

library(softImpute)
library(ranger)
library(tidyverse)
library(reshape2)

# Part a

Songs = read.csv("Songs.csv")
MusicRatings = read.csv("MusicRatings.csv")
Users = read.csv("Users.csv")

min(data_MusicRatings$rating)
max(data_MusicRatings$rating)

# a)

set.seed(345)

train.ids <- sample(nrow(MusicRatings), 0.84*nrow(MusicRatings))
train.mr <- MusicRatings[train.ids,]
test <- MusicRatings[-train.ids,]

# split testing into real testing and validation set

test.ids <- sample(nrow(test), 0.5*nrow(test))
test.mr <- test[test.ids,]
validation <- test[-test.ids,]

# Split validation into validation A and validation B

val.ids <- sample(nrow(validation), 0.5*nrow(validation))
validationA <- validation[val.ids,]
validationB <- validation[-val.ids,]

#a)
train.mr
#b))
validationA
#c)
validationB
#d)
test.mr

# Construct an incomplete training set ratings matrix

mat.train <- Incomplete(train.mr$userID, train.mr$songID, train.mr$rating)

# Part b)
```

```
# i)

# In the dataset, there are three parameters included in model 1, the alpha term, beta term, and the noise term. From our training set,
#we have 243,103 observations to train the model with.

# ii)

mat.train.centered <- biScale(mat.train, maxit = 10000, row.scale = FALSE, col.scale = FALSE)
# mat.train.centered is X_ij - alpha_i - beta_j
alpha <- attr(mat.train.centered, "biScale:row")$center
beta <- attr(mat.train.centered, "biScale:column")$center

Users$alpha <- alpha
Songs$beta <- beta

length(alpha)+length(beta)

Decreasing_Song <- sort(Songs$beta, decreasing = TRUE)
Decreasing_Song[1]
Decreasing_Song[2]
Decreasing_Song[3]

#iii)

Decreasing_Users <- sort(Users$alpha, decreasing = TRUE)
Decreasing_Users[1]
Decreasing_Users[2]
Decreasing_Users[3]

#iv)

OSR2 <- function(predictions, train, test) {
  SSE <- sum((test - predictions)^2)
  SST <- sum((test - mean(train))^2)
  r2 <- 1 - SSE/SST
  return(r2)
}

X_ij = Users$alpha[test.mr$userID] + Songs$beta[test.mr$songID]

Partb_MAE <- mean(abs(X_ij - test.mr$rating))
Partb_RMSE <- sqrt(mean((X_ij - test.mr$rating)^2))
Partb_OSR2 <- OSR2(X_ij, train.mr$rating, test.mr$rating)

# Part c)

# i)

# In the dataset, there are four parameters included in model 2, the alpha term, beta term, the noise term, and the Z term. From our training
set,
#we have 243,103 observations to train the model with.

# ii)

# compute validation set MAE for rank = 1,2,...,20
mae.vals.cii = rep(NA, 20)
for (rnk in seq_len(20)) {
  print(str_c("Trying rank.max = ", rnk))
  mod.cii <- softImpute(mat.train.centered, rank.max = rnk, lambda = 0, maxit = 1000)
  preds.cii <- impute(mod.cii, validationA$userID, validationA$songID) %>% pmin(5) %>% pmax(1)
  mae.vals.cii[rnk] <- mean(abs(preds.cii - validationA$rating))
}

mae.val.cii.df <- data.frame(rnk = seq_len(20), mae = mae.vals.cii)
ggplot(mae.val.cii.df, aes(x = rnk, y = mae)) + geom_point(size = 3) +
```

```
  ylab("Validation MAE") + xlab("Number of Archetypal Users") +
  theme_bw() + theme(axis.title=element_text(size=18), axis.text=element_text(size=18))

mae.vals.cii
min(mae.vals.cii)

# iii)

# choose k = 4

mod.ciii.final <- softImpute(mat.train.centered, rank.max = 4, lambda = 0, maxit = 1000)
preds.ciii.final <- impute(mod.ciii.final, test$userID, test$songID) %>% pmin(5) %>% pmax(1)

Partc_MAE <- mean(abs(preds.ciii.final - test.mr$rating))
Partc_RMSE <- sqrt(mean((preds.ciii.final - test.mr$rating)^2))
Partc_OSR2 <- OSR2(preds.ciii.final, train.mr$rating, test.mr$rating)


# Part d)

# i)

# Create a copy of training and testing set, and
# add genre and year to the training set and testing set

train.mr.new <- train.mr
train.mr.new <- inner_join(train.mr.new,Songs,by='songID')

test.mr.new <- test.mr
test.mr.new <- inner_join(test.mr.new,Songs,by='songID')

# Treat as factors/categorical variables

train.mr.new$year = as.factor(train.mr.new$year)
train.mr.new$genre = as.factor(train.mr.new$genre)
test.mr.new$year = as.factor(test.mr.new$year)
test.mr.new$genre = as.factor(test.mr.new$genre)

# Train Linear Regression model

LR <- lm(rating ~ year + genre, data=train.mr.new)
summary(LR)
LRPredictions <- predict(LR, newdata=test.mr.new)

Partd_LR_MAE <- mean(abs(LRPredictions - test.mr.new$rating))
Partd_LR_RMSE <- sqrt(mean((LRPredictions - test.mr.new$rating)^2))
Partd_LR_OSR2 <- OSR2(LRPredictions, train.mr.new$rating, test.mr.new$rating)

# Train Random Forest model

RF = randomForest(rating ~ year + genre, data=train.mr.new)
RFPredictions = predict(RF, newdata = test.mr.new)
table(test.mr.new$rating, RFPredictions)

Partd_RF_MAE <- mean(abs(RFPredictions - test.mr.new$rating))
Partd_RF_RMSE <- sqrt(mean((RFPredictions - test.mr.new$rating)^2))
Partd_RF_OSR2 <- OSR2(RFPredictions, train.mr.new$rating, test.mr.new$rating)


# ii)

validationB <- inner_join(validationB,Songs,by='songID')
validationB$year = as.factor(validationB$year)
validationB$genre = as.factor(validationB$genre)

# Blending
```

```r
val.preds.cf <- impute(mod.ciii.final, validationB$userID, validationB$songID)
val.preds.lm <- predict(LR, newdata = validationB)
val.preds.rf <- predict(RF, newdata = validationB)

# Build validation set data frame
val.blending_df = data.frame(rating = validationB$rating, cf_preds = val.preds.cf,
                             lm_preds = val.preds.lm, rf_preds = val.preds.rf)

# Train blended model
blend.mod = lm(rating ~ . -1, data = val.blending_df)
summary(blend.mod)
# Test blended model
test.blending_df = data.frame(rating = test.mr.new$rating, cf_preds = preds.ciii.final,
                              lm_preds = LRPredictions, rf_preds = RFPredictions)

test.preds.blend <- predict(blend.mod, newdata = test.blending_df)

Partd_blend_MAE <- mean(abs(test.preds.blend - test.mr.new$rating))
Partd_blend_RMSE <- sqrt(mean((test.preds.blend - test.mr.new$rating)^2))
Partd_blend_OSR2 <- OSR2(test.preds.blend, train.mr.new$rating, test.mr.new$rating)
```