# ASSIGNMENT 5

## ENGIN 242 - Applications in Data Analytics

Aditya Peshin

Enrollment Number: 3035280249

**Problem 1: Collaborative Filtering for Recommending Songs to Music Listeners**

(A) Load the data into R. How many songs are in this dataset? How many users? What is the range of values that the ratings take on?

Ans. There are 807 songs in the songs dataset, and 2421 users in the users dataset. The ratings take values in the range [1 , 3.433]

(B) Let X denote the “complete" ratings matrix, i.e., $X_{i,j}$ denotes either the observed rating if user i actually rated song j or the “hypothetical" such rating if user i has not yet rated song j. We are interested in predicting the values of $X_{i;j}$ that are not observed. Let us first consider the following model:

$$X_{i;j} = \alpha_i + \beta_j + \in_{i;j} \qquad (1)$$

where $\alpha_i$ is a coefficient that depends only on the particular row i (i.e., user), $\beta_j$ is a coefficient that depends only on the particular column j (i.e., song), and $\in_{i;j}$ is a noise term.
i) (5 points) For this dataset, how many total parameters are included in model (1)? How many observations do we have to train the model with?

Ans. For this model, we have a total of 2421(nusers) + 807(nsongs) = **3228 parameters** to be trained. Since the model depends on the various values of $\alpha_i$ and $\beta_j$ the total number of parameters will be the sum total of the number of users and the number of songs.
The total number of observations in the training dataset is = nrow(train) = **243,104 obs**

ii) Using the output of the biScale function (check the documentation to see precisely what this function returns, and you may need to look up the attr function), what are the three most popular songs after removing for the bias due to users’ affinity for rating songs highly (or lowly)? Provide the song ID numbers as well as the song and artist names for these three songs. Explain, in your own words, how your answer relates to model (1).

Ans. The three most popular songs according to the model are:

| Rank | Song ID | Song Name | Song Artist |
|---|---|---|---|
| 1 | 54 | You're The One | Dwight Yoakam |
| 2 | 26 | Undo | Bjork |
| 3 | 439 | Secrets | OneRepublic |

Here, these songs are the ones with the largest value of $\beta_j$ overall. By arranging the songs list in decreasing order of beta($\beta_j$), we know that the songs at the top are those that are highly rated by all the users who listen to them.

iii) Likewise, which three users are the most enthused about songs after removing for the bias due to the effect of the popularity of songs? Provide the user ID numbers of these three users. No need to provide an explanation.

Ans. Here, the users who are most enthused about music are:
These ranks were obtained by taking the users with the highest values of $\alpha_i$

| Rank | UserID |
|---|---|
| 1 | 1540 |
| 2 | 838 |
| 3 | 1569 |

iv) What is the out-of-sample performance of the fitted model on the previously constructed test set? Report the test set MAE, RMSE, and the OSR2 values.

Ans. Here, the out of sample performance (on the test set) can be summarised as:

| Simple Additive Model | |
|---|---|
| MAE | 0.04084026 |
| RMSE | 0.05582044 |
| $OSR^2$ | 0.2799879 |

(C) Now let's consider the following model:

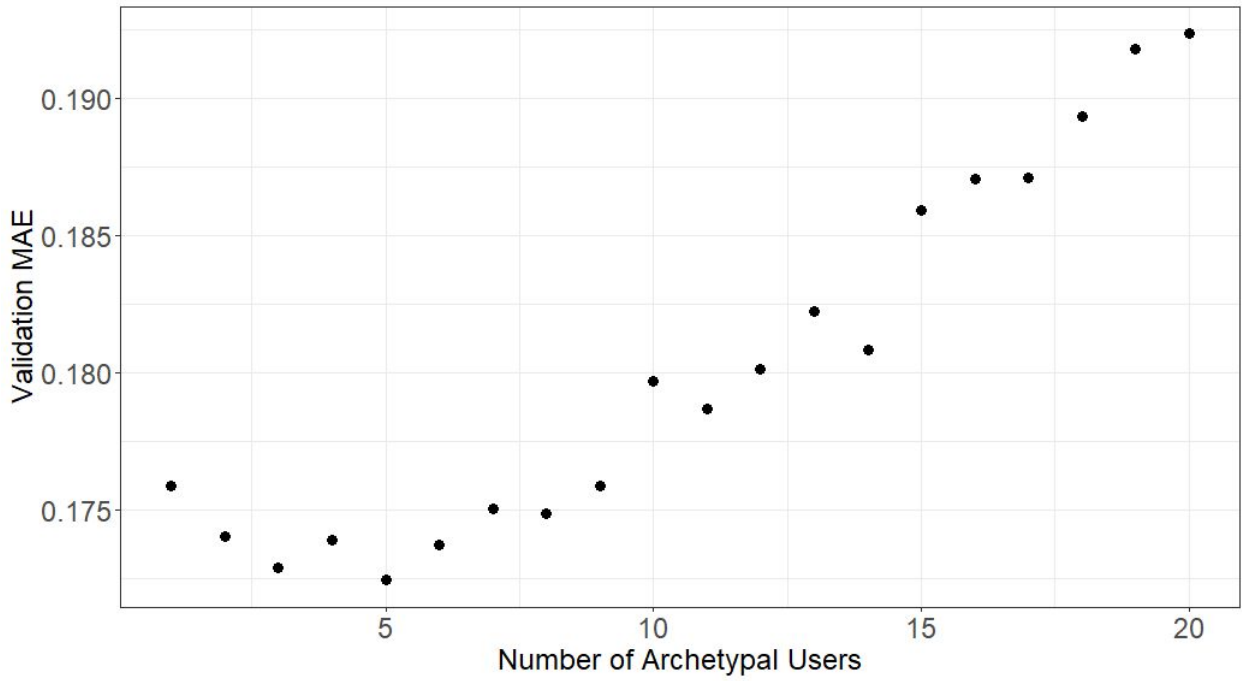
$$X_{i;j} = Z_{i;j} + \alpha_i + \beta_j + \in_{i;j} ; (2)$$

which is the same as part (B) except for an additional term Zi;j. Here, Z represents the low-rank collaborative filtering model that we discussed in lecture.
(i) For this dataset, how many total parameters are included in model (2) (the answer should depend on k)? How many observations do we have to train the model with?

Ans. If we consider the number of archetypes to be 'k', then the number of parameters in this model will be 2421 + 807 + W + S In addition to the $\alpha_i$ and $\beta_j$ parameters that we originally had, we will also need the weights of the archetypal ratings for each user, and the archetypal song ratings themselves. Here, W represents the vector of weights indicating how similar a particular user is to a certain archetype. S indicates the ratings of each archetype for all the songs.
W - > Weights of each obs wrt archetype i.e. 2421 * k
S -> Archetype ratings for each movie i.e. 807 * k
Thus, the total number of parameters is = 2421 + 807 + 2421 * k + 807 * k = **3228(k+1)** params.
The total number of observations in the training dataset is = nrow(train) = **243,104 obs**
However, when we look at the centered matrix, we see that there are 807*2421 = **1953747 obs**

(ii) We will fit the model (2) by using the previously computed estimates of α and β from part (b). That is, letting α and β denote these estimates, we will use the softImpute function to fit a collaborative filtering model on the incomplete training set matrix of residuals $X^{C}_{i,j} := X_{i,j} - \alpha_i - \beta_j$. Thankfully, this object has already been returned to us by the biScale function in part (b). Using the previously constructed validation set A, determine the value of k, i.e., the number of archetypes, that should be selected. Use a plot to justify and explain the value of k that you selected

Ans. I tried various values of k, from 1 to 20, and plotted the validation set MAE against k on a graph. Also, with respect to unscaling the predictions from the basic collaborative model, I set one of the parameters of the impute function called “unscale” to TRUE in order to reverse the centering effects and get an accurate estimate. As we can see from the graph below, the best value of ‘k’ that I got was 5, as it has the lowest validation set MAE.
NOTE : While building this model, and used the value of range obtained in Part (A) [1,3.433] as upper and lower limit on the impute function.




(iii) Build a final collaborative filtering model on the training set using the previously determined value of k. What is the out-of-sample performance of the fitted model on the previously constructed test set? Report the test set MAE, RMSE, and the OSR2 values. (Note: the impute function will work as expected since it will detect the α and β attributes associated with the model returned by the softImpute function.)

Ans. Using the best value of ‘k’ (as 5), I built the best collaborative filtering model, and the results are as shown below.

| Collaborative Filtering Model (k=5) | |
|---|---|
| MAE | 0.03464094 |
| RMSE | 0.047334 |
| $OSR^2$ | 0.2892328 |

(D) In this part of the problem, we will additionally incorporate some of the features associated with songs.

(i) First, fit two distinct models to predict the rating based on the following independent variables: (i) genre of the song, (ii) year that the song was released. Be sure to treat (i) and (ii) as factors/categorical variables and to build the model using only the same training set as before. Report the test set MAE, RMSE, and the OSR2 values of your two models

Ans. For the Genre of the song, I used a linear regression model. The out of sample performance of this model can be seen below.

| Rating vs. Genre (Linear Regression) | |
|---|---|
| MAE | 0.0459205 |
| RMSE | 0.05583358 |
| $OSR^2$ | 0.01105538 |

For the Year that the song was released, I used a CART model with cross validation on the cp parameter. The out of sample performance of this model can be seen below.

| Rating vs. Year (CART) | |
|---|---|
| MAE | 0.04559354 |
| RMSE | 0.05548736 |
| $OSR^2$ | 0.02328205 |

These are the out of sample performance metrics on the test set.

(ii) Now, use validation set B to perform blending of the collaborative filtering model (2) trained in part (c) and the two models trained in part (i) above. Report the test set MAE, RMSE, and the OSR2 values of the blended model. Do the additional features associated with songs add a lot of predictive power on top of the collaborative filtering model?

Ans. Using the predictions of the collaborative filtering model, the linear regression on Genre and the CART on the year of release, the following out of sample performance metrics were observed.

| Rating vs. Blended model predictions (Test set) | |
|---|---|
| MAE | 0.04431331 |
| RMSE | 0.05818722 |
| $OSR^2$ | 0.3125895 |

Now, in order to determine if the blended model is useful, we do bootstrapping on the test set to evaluate the confidence intervals of test set metrics of the various models. Doing the bootstrapping via 1000 bootstrapped test sets, we get:

| Bootstrapped Confidence Intervals (95% level) | | | | |
|---|---|---|---|---|
| | CF Model | LR (Genre) | CART (year) | Blended Model |
| MAE | (0.0544, 0.0577) | (0.0758, 0.0800) | (0.0749, 0.0789) | (0.0526, 0.0557) |
| RMSE | (0.1712, 0.1753) | (0.2276, 0.2316) | (0.226, 0.230) | (0.1754, 0.1792) |
| $OSR^2$ | (0.2692, 0.3092) | (0.0074, 0.0146) | (0.0176, 0.0292) | (0.2964, 0.3288) |

To answer the question with respect to improvements in predictive power, I am using $OSR^2$ as a measure of the predictive power of the model. However this can be done with any of the three performance metrics.

Here, at a 95% confidence level, we cannot say that there is a difference in the predictive power of the Blended model and the Collaborative Filtering model, as their confidence intervals for $OSR^2$ overlap with each other. Looking at the MAE and RMSE values, we can see that there is no statistical difference between these two models for MAE, however the Collaborative Filtering model is better than the Blended model in terms of RMSE at the 95% level.

Thus, we cannot say that the collaborative filtering model adds a lot of predictive value to the Collaborative Filtering model.

**R Code:**

```
# Homework 5
setwd("D:\\Studies\\Berkeley\\Semester 1\\242\\Hw 5")
library(softImpute)
library(randomForest)
library(ranger)
library(dplyr)
library(tidyverse)
library(reshape2)
library(glmnet)
library(broom)
library(rpart)
library(caret)
library(boot)

mean_squared_error <- function(data, index) {
  responses <- data$response[index]
  predictions <- data$prediction[index]
  MSE <- mean((responses - predictions)^2)
  return(MSE)
}
mean_absolute_error <- function(data, index) {
  responses <- data$response[index]
  predictions <- data$prediction[index]
  MAE <- mean(abs(responses - predictions))
  return(MAE)
}
OS_R_squared <- function(data, index) {
  responses <- data$response[index]
  predictions <- data$prediction[index]
  baseline <- data$baseline[index]
  SSE <- sum((responses - predictions)^2)
  SST <- sum((responses - baseline)^2)
  r2 <- 1 - SSE/SST
  return(r2)
}
all_metrics <- function(data, index) {
  mse <- mean_squared_error(data, index)
  mae <- mean_absolute_error(data, index)
  OSR2 <- OS_R_squared(data, index)
  return(c(mse, mae, OSR2))
}
OSR2 <- function(predictions, train, test) {
```

```
  SSE <- sum((test - predictions)^2)
  SST <- sum((test - mean(train))^2)
  r2 <- 1 - SSE/SST
  return(r2)
}
musicratings <- read.csv("D:\\Studies\\Berkeley\\Semester 1\\242\\Hw 5\\MusicRatings.csv")
songs <- read.csv("D:\\Studies\\Berkeley\\Semester 1\\242\\Hw 5\\Songs.csv")
users <- read.csv("D:\\Studies\\Berkeley\\Semester 1\\242\\Hw 5\\Users.csv")


# PART (A)
str(users)
# 2421 users
str(songs)
# 807 songs
summary(musicratings$rating)
# Range of values of scores are [1,3.433]
# Splitting the data into training and testing (8%)
set.seed(345)
train.ids <- sample(nrow(musicratings), 0.92*nrow(musicratings))
train <- musicratings[train.ids,]
test <- musicratings[-train.ids,]
# split training into real training (84%) and 2 validation sets
# 4% for Collaborative Filtering
val1.ids <- sample(nrow(train), (4/92)*nrow(train))
val1 <- train[val1.ids,]
train <- train[-val1.ids,]
# 4% for Blending
val2.ids <- sample(nrow(train), (4/88)*nrow(train))
val2 <- train[val2.ids,]
train <- train[-val2.ids,]
# Constructing an Incomplete Training set ratings matrix
mat.train <- Incomplete(train$userID, train$songID, train$rating)
summary(train)


# PART (B)
# PART (B.i)
# Here, we will need 2421(nusers) + 807(nsongs) = 3228 parameters to be trained
set.seed(345)
mat.train.centered <- biScale(mat.train, maxit = 1000, row.scale = FALSE, col.scale = FALSE)
```

```
# PART (B.ii)
# 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
# Reordering the songs matrix in order to find the most popular songs
songs_new <- songs
songs_new$BETA <- beta
songs_new <- songs_new[order(-songs_new$BETA),]
head(songs_new,3)
# songID        songName year         artist   genre    BETA
# 54     54 You're The One 1990   Dwight Yoakam Country 1.710318
# 26     26       Undo 2001               Bjork    Rock 1.691155
# 439   439     Secrets 2009   OneRepublic   Rock 1.641377


# PART (B.iii)
# Reordering the users matrix to see which are the most enthused about music
users_new <- users
users_new$ALPHA <- alpha
users_new <- users_new[order(-users_new$ALPHA),]
head(users_new,3)
# userID         ALPHA
# 1540   1540 0.5959218
# 838   838 0.4964753
# 1569   1569 0.4770457


# PART (B.iv)
# Out of Sample Performance of Simple Additive Model
# Creating matrix of prediction values
SAMpreds <- matrix(rep(NA,1953747), nrow = 2421, ncol = 807)
for(i in nrow(SAMpreds))
  for(j in ncol(SAMpreds))
          SAMpreds[i,j] = alpha[i] + beta[j]
str(test)
# MAE of the Simple Additive Model
mean(abs(SAMpreds[test$userID,test$songID] - test$rating))/5
# MAE = 0.04084026
# RMSE of the Simple Additive Model
sqrt(mean((SAMpreds[test$userID,test$songID] - test$rating)^2))/5
# RMSE = 0.05582044
# R^2 of the Simple Additive Model
# I use a copy of the test set and add Alpha and Beta columns to it
```

```
testa <- test
testa$alpha <- alpha[testa$userID]
testa$beta <- beta[testa$songID]
# Prediction = alpha[i] + beta[i]
testa$preds <- testa$alpha + testa$beta
# R2
OSR2(testa$preds, train$rating, testa$rating)
# R^2 = 0.2799879


# PART (C) - Collaborative Filtering
# Part (C.i)
# The number of parameters in this model will be 2421 + 807 + W + S
# W - > Weights of each obs wrt archetype i.e. 2421 * k no of weights
# S -> Archetype ratings for each movie i.e. 807 *k


# Part (C.ii)
# Trying different number of archetypes, i.e. rank = 1,2,...,20
CF.mae.vals = rep(NA, 20)
for (rnk in seq_len(20)) {
  print(str_c("Trying rank.max = ", rnk))
  CFmod <- softImpute(mat.train.centered, rank.max = rnk, lambda = 0, maxit = 1000)
  preds <- impute(CFmod, val1$userID, val1$songID, unscale = TRUE) %>% pmin(3.433) %>%
pmax(1)
# here the impute function has a parameter called "unscale" which is set to true
# This parameter, when set to true, reverses the centering and scaling on the predictions.
  CF.mae.vals[rnk] <- mean(abs(preds - val1$rating))
}
CF.mae.val.df <- data.frame(rnk = seq_len(20), mae = CF.mae.vals)
ggplot(CF.mae.val.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))

# Final value of k = 5 from graph


# Part (C.iii)
set.seed(345)
finalCFmod <- softImpute(mat.train.centered, rank.max = 5, lambda = 0, maxit = 1000)
finalCFpreds <- impute(finalCFmod, test$userID, test$songID) %>% pmin(3.433) %>% pmax(1)
# Calculating the MAE
mean(abs(finalCFpreds - test$rating))/5
```

```
# Here, the MAE is 0.03464094
# Calculating the RSS
sqrt(mean((finalCFpreds - test$rating)^2))/5
# Here, the RMSE is 0.047334
# Calculating R^2
OSR2(finalCFpreds, train$rating, test$rating)
# Here, the R^2 is 0.2892328


# PART (D)
# Part (D.i)
# Building a model using only genre of the song
# Preparing the datasets
train2 <- train
train2$genre <- songs$genre[train2$songID]
train2$year <- songs$year[train2$songID]
test2 <- test
test2$genre <- songs$genre[test2$songID]
test2$year <- songs$year[test2$songID]
val2a <- val2
val2a$genre <- songs$genre[val2a$songID]
val2a$year <- songs$year[val2a$songID]
# Building the model
modLM <- lm(rating ~ genre, data = train2)
summary(modLM)
modLMpreds <- predict(modLM, newdata=test2)
# Calculating the MAE
mean(abs(modLMpreds - test2$rating))/5
# Here, the MAE is 0.0459205
# Calculating the RSS
sqrt(mean((modLMpreds - test2$rating)^2))/5
# Here, the RMSE is 0.05583358
# Calculating R^2
OSR2(modLMpreds, train2$rating, test2$rating)
# OSR2 = 0.01105538
# Building a model using only the year the song was released
set.seed(456)
cpVals = data.frame(cp = seq(0, .04, by=.002))
modCART <- train(rating ~ year,
                 data = train2,
                 method = "rpart",
                 tuneGrid = cpVals,
                 trControl = trainControl(method = "cv", number=10))
```

```
modCART$results
modCARTpreds <- predict(modCART, newdata=test2)
# Calculating the MAE
mean(abs(modCARTpreds - test2$rating))/5
# Here, the MAE is 0.04559354
# Calculating the RSS
sqrt(mean((modCARTpreds - test2$rating)^2))/5
# Here, the RMSE is 0.05548736
# Calculating R^2
OSR2(modCARTpreds, train2$rating, test2$rating)
# OSR2 = 0.02328205


# Part (D.ii)
# Using Validation set 2 to build blending model
val2predsCF <- impute(finalCFmod, val2a$userID, val2a$songID) %>% pmin(3.433) %>%
pmax(1)
val2predsLMG <- predict(modLM, newdata=val2a)
val2predsLMY <- predict(modCART, newdata=val2a)
val.blending_df = data.frame(rating = val2a$rating, cf_preds = val2predsCF, lm1_preds =
val2predsLMG , lm2_preds =val2predsLMY)
mod_blend = lm(rating ~ . -1, data = val.blending_df)
summary(mod_blend)
test2CF  <- impute(finalCFmod, test2$userID, test2$songID) %>% pmin(3.433) %>% pmax(1)
test2predsLMG <- predict(modLM, newdata=test2)
test2predsLMY <- predict(modCART, newdata=test2)
test.blending_df = data.frame(rating = test2$rating, cf_preds = test2CF, lm1_preds =
test2predsLMG , lm2_preds =test2predsLMY)
test.preds.blend <- predict(mod_blend, newdata = test.blending_df)
mean(abs(test.preds.blend - test$rating))/4
# MAE = 0.04431331
sqrt(mean((test.preds.blend - test$rating)^2))/4
# RMSE = 0.05818722
OSR2(test.preds.blend, train$rating, test$rating)
# OSR2 = 0.3125895


# Bootstrapping to test confidence intervals of new model
# Bootstrapping Blended model metrics
blend_test_set = data.frame(response = test2$rating, prediction = test.preds.blend , baseline =
mean(train2$rating))
# do bootstrap
set.seed(456)
```

```
blend_boot <- boot(blend_test_set, all_metrics, R = 1000)
# get confidence intervals
boot.ci(blend_boot, index = 1, type = "basic")
boot.ci(blend_boot, index = 2, type = "basic")
boot.ci(blend_boot, index = 3, type = "basic")


# CART(year) bootstrap model metrics
CART_test_set = data.frame(response = test2$rating, prediction = modCARTpreds, baseline =
mean(train2$rating))
# do bootstrap
set.seed(456)
CART_boot <- boot(CART_test_set, all_metrics, R = 1000)
# get confidence intervals
boot.ci(CART_boot, index = 1, type = "basic")
boot.ci(CART_boot, index = 2, type = "basic")
boot.ci(CART_boot, index = 3, type = "basic")


# LR(genre) bootstrap model metrics
LRG_test_set = data.frame(response = test2$rating, prediction = modLMpreds, baseline =
mean(train2$rating))
# do bootstrap
set.seed(456)
LRG_boot <- boot(LRG_test_set, all_metrics, R = 1000)
# get confidence intervals
boot.ci(LRG_boot, index = 1, type = "basic")
boot.ci(LRG_boot, index = 2, type = "basic")
boot.ci(LRG_boot, index = 3, type = "basic")


# CF bootstrap model metrics
CF_test_set = data.frame(response = test2$rating, prediction = finalCFpreds, baseline =
mean(train2$rating))
# do bootstrap
set.seed(456)
CF_boot <- boot(CF_test_set, all_metrics, R = 1000)
# get confidence intervals
boot.ci(CF_boot, index = 1, type = "basic")
boot.ci(CF_boot, index = 2, type = "basic")
boot.ci(CF_boot, index = 3, type = "basic")
```

```
# CF Model
# LR (Genre)
# CART (year)
# Blended Model

# MAE
# (0.0544,  0.0577)
# (0.0758,  0.0800)
# (0.0749,  0.0789)
# (0.0526, 0.0557)

# RMSE
# (0.1712,  0.1753)
# (0.2276,  0.2316)
# (0.226,  0.230)
# (0.1754,  0.1792)

# OSR2
# (0.2692,  0.3092)
# (0.0074,  0.0146)
# (0.0176,  0.0292)
# (0.2964,  0.3288)
```