STAT542 Final Project Code

Ilia's Code Below:

Creating the incomplete TripAdvisor matrix

```
rm(list=ls())
setwd('~/Desktop/stat542_final_report_code')
library(recommenderlab); library(tidyr); library(tictoc)

train_data <- as.matrix(read.csv('tripadvisor_review.csv'))
ntotal <- nrow(train_data)*ncol(train_data)
set.seed(100); ind <- sample(x = 1:ntotal, size = round(ntotal/2), replace = FALSE)
train_data_incomplete <- train_data
train_data_incomplete[ind] <- NA

write.csv(as.data.frame(train_data), file =
            'train_data.csv', row.names = FALSE)
write.csv(as.data.frame(train_data_incomplete), file =
            'train_data_incomplete.csv', row.names = FALSE)
```

Running different algorithms on TripAdvisor data

Naive

```
ratings_data <- as.matrix(read.csv("train_data_incomplete.csv"))
ratings_data <- as(ratings_data, "realRatingMatrix")

fullmat <- as.matrix(read.csv("train_data.csv"))
testmat <- as(ratings_data, "matrix")
naive_pred <- mean(testmat, na.rm = TRUE)
mse_naive <- sum(((naive_pred-fullmat)[which(is.na(testmat))])^2)/(sum(1-is.na(testmat)))
mse_improve <- NULL
```

POPULAR

```
method <- "POPULAR"
set.seed(100)
rec <- Recommender(ratings_data, method = method)
pre <- predict(rec, ratings_data,  type = "ratingMatrix")
compmat <- as(pre, "matrix")
mse_method <- sum(((compmat-fullmat)[which(is.na(testmat))])^2)/(sum(1-is.na(testmat)))
mse_improve <- c(mse_improve, 1-(mse_method/mse_naive))
```

ALS

(The following section is executed in about 3 minutes)

```
tic()
method <- "ALS"
set.seed(100)
mse_improve_sub <- NULL
for(lambda in seq(0.01,0.2,0.01)){
  rec <- Recommender(ratings_data, method = method, param = list(lambda=lambda))
  pre <- predict(rec, ratings_data,  type = "ratingMatrix")
  compmat <- as(pre, "matrix")
  mse_method <- sum(((compmat-fullmat)[which(is.na(testmat))])^2)/(sum(1-is.na(testmat)))
  mse_improve_sub <- c(mse_improve_sub, 1-(mse_method/mse_naive))
}
toc()
```

```
# mse_improve_sub_als <- mse_improve_sub
best_lambda <- seq(0.01,0.2,0.01)[which.max(mse_improve_sub_als)]
mse_improve <- c(mse_improve, max(mse_improve_sub_als))
```

LIBMF

The following section is executed in about 30 seconds

```
tic()
method <- "LIBMF"
set.seed(100)
mse_improve_sub <- NULL
for(costp in seq(0.005,0.155,0.005)){
  for(costq in seq(0.005,0.155,0.005)){
    rec <- Recommender(ratings_data, method = method,
                       param = list(costp_l2=costp, costq_l2=costq))
    pre <- predict(rec, ratings_data,  type = "ratingMatrix")
    compmat <- as(pre, "matrix")
    mse_method <- sum(((compmat-fullmat)[which(is.na(testmat))])^2)/(sum(1-is.na(testmat)))
    mse_improve_sub <- c(mse_improve_sub, 1-(mse_method/mse_naive))
  }
}
toc()
```

```
mse_improve <- c(mse_improve, max(mse_improve_sub_libmf))
```

SVD

```
method <- "SVD"
set.seed(100)
mse_improve_sub <- NULL
for(k in 2:5){
  rec <- Recommender(ratings_data, method = method, param=list(k=k))
  pre <- predict(rec, ratings_data,  type = "ratingMatrix")

  compmat <- as(pre, "matrix")
  mse_method <- sum(((compmat-fullmat)[which(is.na(testmat))])^2)/(sum(1-is.na(testmat)))

  mse_improve_sub <- c(mse_improve_sub, 1-(mse_method/mse_naive))
}
best_rank <- which(mse_improve_sub==max(mse_improve_sub, na.rm = TRUE))
```

```
mse_improve <- c(mse_improve, max(mse_improve_sub, na.rm = TRUE))
```

UBCF

The following section takes some extra time to execute, this is why I load the data (ran it once) and comment out the actual code

```
e1 <- new.env(parent = baseenv())
load("save.RData", envir = e1)
mse_improve_sub_ubcf <- e1$mse_improve_sub_ubcf

# method <- "UBCF"
# set.seed(100)
# mse_improve_sub <- NULL
# for(k in 212:300){
#   rec <- Recommender(ratings_data, method = method, param=list(nn=k))
#   pre <- predict(rec, ratings_data,  type = "ratingMatrix")
#
#   compmat <- as(pre, "matrix")
#   sum(is.na(compmat))
#   mse_method <- sum(((compmat-fullmat)[which(is.na(testmat))])^2)/(sum(1-is.na(testmat)))
#
#   mse_improve_sub <- c(mse_improve_sub, 1-(mse_method/mse_naive))
# }
# mse_improve_sub_ubcf <- mse_improve_sub # to save as Rdata

mse_improve <- c(mse_improve, max(mse_improve_sub_ubcf, na.rm = TRUE))
```

IBCF

IBCF with $k \in [2, 9]$ didn't complete the matrix, so, it's not taken into account for comparison. It's hence commented out

```
# method <- "IBCF"
# mse_improve_sub <- NULL
# nacomp <- NULL
# for(k in 2:9){
#   set.seed(100)
#   rec <- Recommender(ratings_data, method = method
#                      , param=list(k=k)
#                      )
#   pre <- predict(rec, ratings_data,  type = "ratingMatrix")
#
#   compmat <- as(pre, "matrix")
#   nacomp <- c(nacomp,sum(is.na(compmat)))
# }

  # mse_improve_sub <- c(mse_improve_sub, 1-(mse_method/mse_naive))
# }
# mse_improve <- c(mse_improve, max(mse_improve_sub, na.rm = TRUE))
mse_improve <- c(mse_improve,"-")
```

Combined table for comparison of results

```
mse_improve[1:5] <- paste(round(as.numeric(mse_improve[1:5])*100, 2),'%',sep = '')

names(mse_improve) <- c("POPULAR","ALS","LIBMF","SVD","UBCF","IBCF")

# mse_improve
```

David's code below

Feedback predictions, evaluating all algorithms on MovieLens, TripAdvisor

```
# Final Predictions

library(MASS)
library(recommenderlab)
setwd("~/R/STAT542")

ratings_data <- read.csv("Feedback.csv")
ratings_data <- ratings_data[,-1] # drop the first column
ratings_data <- data.matrix(ratings_data)

train_real <- as(ratings_data, "realRatingMatrix")
rec <- Recommender(train_real, method = "LIBMF", param=list(costp_l2=0.1, costq_l2=0.1))

pre <- predict(rec, train_real, type = "ratingMatrix")
pre_m <- as(pre, "matrix")

rounded_pre <- round(pre_m)

vec_ratings <- c(ratings_data)
null_indices <- which(sapply(vec_ratings, is.na) == TRUE)

filled_pre <- ratings_data
filled_pre[null_indices] <- rounded_pre[null_indices]

write.matrix(filled_pre,file="predictions.csv")
```

```
# TripAdvisor

library(recommenderlab)
setwd("~/R/STAT542")

ratings_data <- read.csv("train_data.csv")
ratings_data <- data.matrix(ratings_data)

vec_ratings <- c(ratings_data)
non_null_indices <- which(sapply(vec_ratings, is.na) == FALSE)
n_non_null <- length(non_null_indices)

dim(ratings_data)

pred_acc <- rep(0, 12)
```

```
nulls <- rep(0, 12)
iters <- 1

set.seed(100)
for(j in 1:iters) {
  locations = sample(non_null_indices, as.integer(0.4 * n_non_null), replace=FALSE)
  test <- ratings_data[locations]
  train <- ratings_data
  train[locations] <- NA
  train_real <- as(train, "realRatingMatrix")

  naive <- mean(train, na.rm = TRUE)
  pred_acc[1] <- sum((naive - test) ** 2, na.rm=TRUE) / length(test)
  nulls[1] <- 0

  recommenders <- c(Recommender(train_real, method = "UBCF", param=list(nn=150)),
                    Recommender(train_real, method = "UBCF", param=list(nn=200)),
                    Recommender(train_real, method = "UBCF", param=list(nn=250)),
                    Recommender(train_real, method = "UBCF", param=list(nn=300)),
                    Recommender(train_real, method = "IBCF", param=list(k=3)),
                    Recommender(train_real, method = "IBCF", param=list(k=4)),
                    Recommender(train_real, method = "IBCF", param=list(k=5)),
                    Recommender(train_real, method = "ALS"),
                    Recommender(train_real, method = "LIBMF"),
                    Recommender(train_real, method = "SVD", param=list(k=4)),
                    Recommender(train_real, method = "POPULAR"))

  i <- 2
  for(rec in recommenders){
    pre <- predict(rec, train_real, type = "ratingMatrix")
    pre_m <- as(pre, "matrix")
    mse <- mean((pre_m[locations] - test) ** 2, na.rm=TRUE)
    pred_acc[i] = pred_acc[i] + (mse / iters)
    nulls[i] <- sum(is.na(pre_m[locations]))
    i <- i+1
  }
}
```

```
ratings_data <- read.csv("train_data.csv")
ratings_data <- data.matrix(ratings_data)
set.seed(100)

pred_acc <- rep(0, 6)
iters <- 1

# for(j in 1:iters) {
    locations = sample(1:9800, 4900, replace=FALSE) # select random locations to erase
    test <- ratings_data[locations]
    train <- ratings_data
    train[locations] <- NA
    naive_pred <- mean(train, na.rm=TRUE)
    train_real <- as(train, "realRatingMatrix")
    # train_real <- normalize(train_real)
```

```
    # rec <- Recommender(train_real, method = "SVD", param=list(k=4))
    # pre <- predict(rec, train_real, type = "ratingMatrix")
    # pre_m <- as(pre, "matrix")
    mse <- sum((naive_pred - test) ** 2) / length(test)
    pred_acc = pred_acc + mse
# }
```

```
# MovieLens

library(recommenderlab)
setwd("~/R/STAT542")
ratings_data <- read.csv("ratings.csv")
ratings_data <- ratings_data[,-4]
ratings_data <- as(ratings_data, "realRatingMatrix")
ratings_data <- ratings_data[rowCounts(ratings_data) > 50, ]
ratings_data <- ratings_data[, colCounts(ratings_data) > 50]

ratings_data <- as(ratings_data, "matrix")
ratings_data <- data.matrix(ratings_data)

vec_ratings <- c(ratings_data)
non_null_indices <- which(sapply(vec_ratings, is.na) == FALSE)
n_non_null <- length(non_null_indices)

dim(ratings_data)

pred_acc <- rep(0, 16)
nulls <- rep(0, 16)
iters <- 1

set.seed(100)
for(j in 1:iters) {
  locations = sample(non_null_indices, as.integer(0.4 * n_non_null), replace=FALSE)
  test <- ratings_data[locations]
  train <- ratings_data
  train[locations] <- NA
  train_real <- as(train, "realRatingMatrix")

  naive <- mean(train, na.rm = TRUE)
  pred_acc[1] <- sum((naive - test) ** 2, na.rm=TRUE) / length(test)
  nulls[1] <- 0

  recommenders <- c(
    Recommender(train_real, method = "UBCF", param=list(nn=150)),
    Recommender(train_real, method = "UBCF", param=list(nn=200)),
    Recommender(train_real, method = "UBCF", param=list(nn=250)),
    Recommender(train_real, method = "UBCF", param=list(nn=300)),
    Recommender(train_real, method = "IBCF", param=list(k=150)),
    Recommender(train_real, method = "IBCF", param=list(k=200)),
    Recommender(train_real, method = "IBCF", param=list(k=250)),
    Recommender(train_real, method = "IBCF", param=list(k=300)),
    Recommender(train_real, method = "ALS", param=list(lambda=0.1)),
    Recommender(train_real, method = "LIBMF"),
    Recommender(train_real, method = "SVD", param=list(k=5)),
```

```
    Recommender(train_real, method = "SVD", param=list(k=10)),
    Recommender(train_real, method = "SVD", param=list(k=20)),
    Recommender(train_real, method = "SVD", param=list(k=25)),
    Recommender(train_real, method = "POPULAR")
    )

  i <- 2
  for(rec in recommenders){
    pre <- predict(rec, train_real, type = "ratingMatrix")
    pre_m <- as(pre, "matrix")
    mse <- mean((pre_m[locations] - test) ** 2, na.rm=TRUE)
    pred_acc[i] = pred_acc[i] + (mse / iters)
    nulls[i] <- sum(is.na(pre_m[locations]))
    i <- i+1
  }
}
```

```
# MovieLens

library(recommenderlab)
setwd("~/R/STAT542")
ratings_data <- read.csv("ratings.csv")
ratings_data <- ratings_data[,-4]
ratings_data <- as(ratings_data, "realRatingMatrix")
ratings_data <- ratings_data[rowCounts(ratings_data) > 50, ]
ratings_data <- ratings_data[, colCounts(ratings_data) > 50]

ratings_data <- as(ratings_data, "matrix")
ratings_data <- data.matrix(ratings_data)

vec_ratings <- c(ratings_data)
non_null_indices <- which(sapply(vec_ratings, is.na) == FALSE)
n_non_null <- length(non_null_indices)

dim(ratings_data)

pred_acc <- rep(0, 10)
nulls <- rep(0, 10)
iters <- 1

set.seed(100)
for(j in 1:iters) {
  locations = sample(non_null_indices, as.integer(0.4 * n_non_null), replace=FALSE)
  test <- ratings_data[locations]
  train <- ratings_data
  train[locations] <- NA
  train_real <- as(train, "realRatingMatrix")

  recommenders <- c(
    Recommender(train_real, method = "ALS", param=list(lambda=0.01)),
    Recommender(train_real, method = "ALS", param=list(lambda=0.03)),
    Recommender(train_real, method = "ALS", param=list(lambda=0.1)),
    Recommender(train_real, method = "ALS", param=list(lambda=0.3)),
    Recommender(train_real, method = "ALS", param=list(lambda=1)),
```

```
    Recommender(train_real, method = "LIBMF", param=list(costp_l2=0.01, costq_l2=0.01)),
    Recommender(train_real, method = "LIBMF", param=list(costp_l2=0.03, costq_l2=0.03)),
    Recommender(train_real, method = "LIBMF", param=list(costp_l2=0.1, costq_l2=0.1)),
    Recommender(train_real, method = "LIBMF", param=list(costp_l2=0.3, costq_l2=0.3)),
    Recommender(train_real, method = "LIBMF", param=list(costp_l2=1, costq_l2=1))
    )

  i <- 1
  for(rec in recommenders){
    pre <- predict(rec, train_real, type = "ratingMatrix")
    pre_m <- as(pre, "matrix")
    mse <- mean((pre_m[locations] - test) ** 2, na.rm=TRUE)
    pred_acc[i] = pred_acc[i] + (mse / iters)
    nulls[i] <- sum(is.na(pre_m[locations]))
    i <- i+1
  }
}
```