Machine Learning Homework 3

Jin Miao

March 7, 2018

Task 1

1.1 Load Data

Go to the Book-Crossing Database website http://www2.informatik.uni-freiburg.de/~cziegler/BX/ (click on the hyperlink), download and import the BX-Book-Ratings dataset in R (make sure to acknowledge and reference appropriately in your paper).

```
mlrec = read.csv(file = "M:/A Master of Science in Marketing Sciences/MS Machine Learning/H
omework3/BX-Book-Ratings.csv", sep = ";", header = TRUE)
```

1.2 Data Selection

Use table() and order() to select the 100 "most active" (i.e. those with the most ratings) users. Then subset the full dataset to select the ratings of these most active users only. How many unique items/book ratings are there in this reduced data set?

```
rater = unique(mlrec$User.ID)
count = c(0,length(rater))
for (i in 1:length(rater))
{
   count[i] = sum(mlrec$User.ID == rater[i])
}

select = order(count,decreasing = TRUE)[1:100]
selectrater = rater[select]

active = subset(mlrec, User.ID %in% selectrater)
```

Answer:

There are 85118 unique items/book ratings in this reduced dataset.

Task 2

2.1 Create Training and Test Sets

Split this reduced dataset into a "training set" consisting of 100000 observations and a "test set" using the sample() function. Use set.seed(1) before running the split so that you get the same results as everyone else.

```
set.seed(1)
train = sample(nrow(active),100000)
training = active[train, ]
test = active[-train,]
```

The number of unique ISBN in the training set is 67726; the number of unique ISBN in the test set is 28530. However, there are only 11138 unique ISBN that exist in both the training and test sets. On top of that, when we evaluate the performance of the recommendaiton algorithm on the test set, we can only use the coexisting 11138 books because we do not have any information about the other books from the training set.

The number of unique User IDs in the training set is 100; the number of unique User IDs in the test set is 100. However, there are exactly 100 unique User IDs that exist in both the training and test sets.

Task 3

3.1 Convert Data into Ratings Matrix

```
# install.packages("reshape2")
library(reshape2)
newactive = dcast(active, User.ID ~ ISBN, value.var = "Book.Rating")
newactive = newactive[,-1]
TrainRating = dcast(training, User.ID ~ ISBN, value.var = "Book.Rating")
rownames(TrainRating) = TrainRating[,1]
TrainRating = TrainRating[,-1]
TestRating = dcast(test, User.ID ~ ISBN, value.var = "Book.Rating")
rownames(TestRating) = TestRating[,1]
TestRating = TestRating[,-1]
```

Task 4

4.1 Single SVD

Apply the single Singular Value Decomposition (SVD) technique to build a recommendation system. Estimate the test error of your recommendation system using the test set. How well does the SVD recommendation system work?

I use 2 latent factors to recreate the rating matrix from Singular Value Decomposition. I also tried this model with more latent factors and found that the contributions from adding more latent factors are quite negligible. Thus, I will only report the results from SVD with two latent factors.

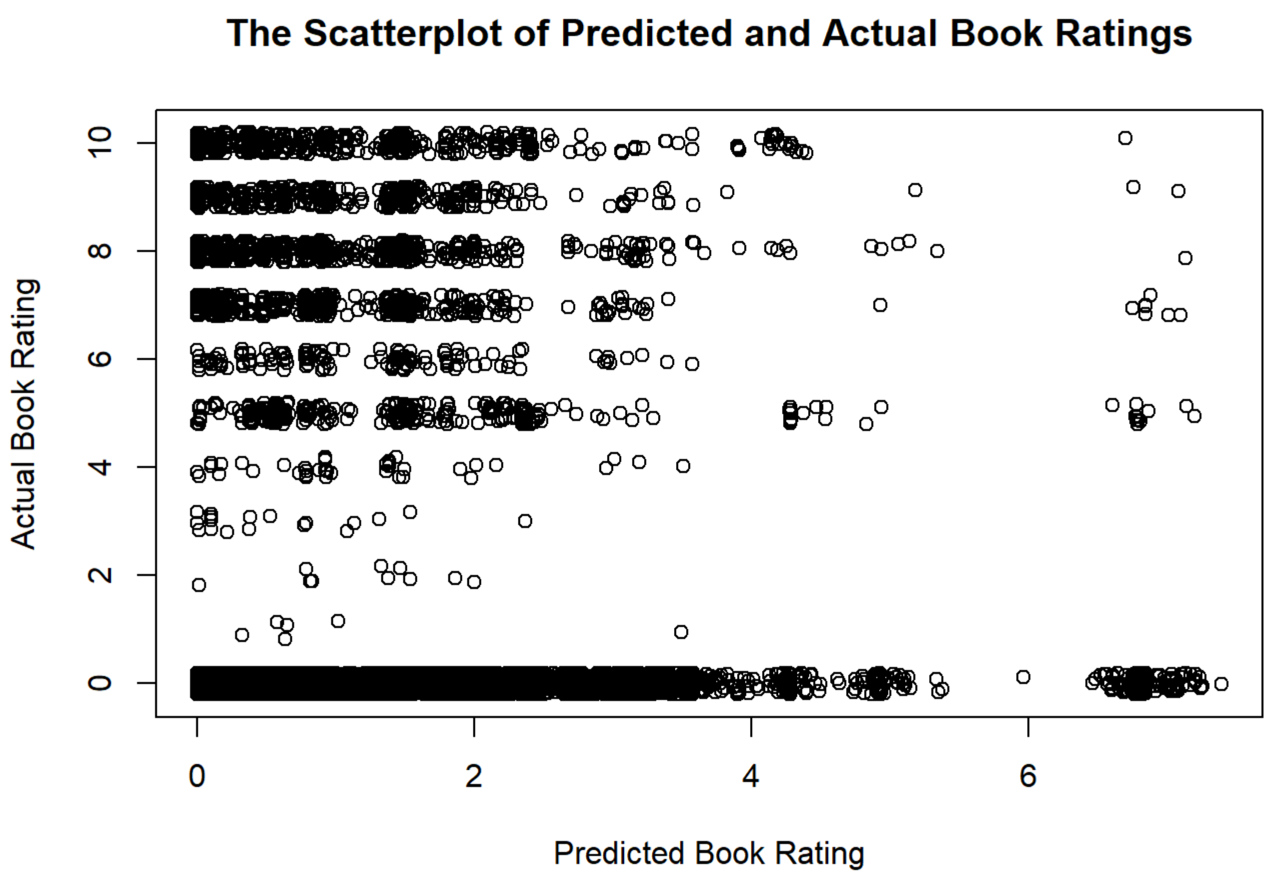
```
## Step 1: Replace Missing Values with Row Means for Training Set

k <- which(is.na(TrainRating), arr.ind=TRUE)
TrainRating1 = TrainRating
TrainRating1[k] <- rowMeans(TrainRating, na.rm=TRUE)[k[,1]]

## Step 2: Singular Value Decomposition

SVD1 <- svd(TrainRating1)
ReTrain1 <- SVD1$u[,1:2] %*% diag(SVD1$d)[1:2,1:2] %*% t(SVD1$v[,1:2])
```

After recreating the training set with SVD as the prediction matrix, I select the subset of this prediction matrix that has exactly the same pair (User.ID, ISBN).

```
rownames(ReTrain1) = rownames(TrainRating1)
colnames(ReTrain1) = colnames(TrainRating1)

newReTrain1 = as.data.frame(ReTrain1)
newReTrain1$User.ID = rownames(ReTrain1)
newR1Prediction <- melt(as.matrix(newReTrain1), id.vars = User.ID,  variable.name = "ISBN",
value.name = "Book.Rating")

newR1Prediction$Book.Rating = as.numeric(as.character(newR1Prediction$Book.Rating))
names(newR1Prediction)[1:2] = c("User.ID","ISBN")
test$index = do.call(paste0, test[c("User.ID","ISBN")])
newR1Prediction$index = do.call(paste0, newR1Prediction[c("User.ID","ISBN")])
mjpredict = subset(newR1Prediction, newR1Prediction$index %in% test$index)
mjpredict = mjpredict[order( mjpredict$index ), ]
mjreality = subset(test, test$index %in% mjpredict$index)
mjreality = mjreality[order( mjpredict$index ), ]
diff = mjpredict$Book.Rating - mjreality$Book.Rating
```

The Mean Square Error for this recommendation system using the test set is 9.1817164, which indicates that, given the ten-point Likert rating scale, the performance of this recommendation is not accurate enough. This is also evidenced from the Scatterplot shown as follows where the predicted book ratings are on the x-axis while the actual rating on the y-axis.

```
plot(mjpredict$Book.Rating , jitter(mjreality$Book.Rating), main = "The Scatterplot of Pred
icted and Actual Book Ratings", xlab = "Predicted Book Rating", ylab = "Actual Book Rating"
)
```



Task 5

5.1 Iterative SVD

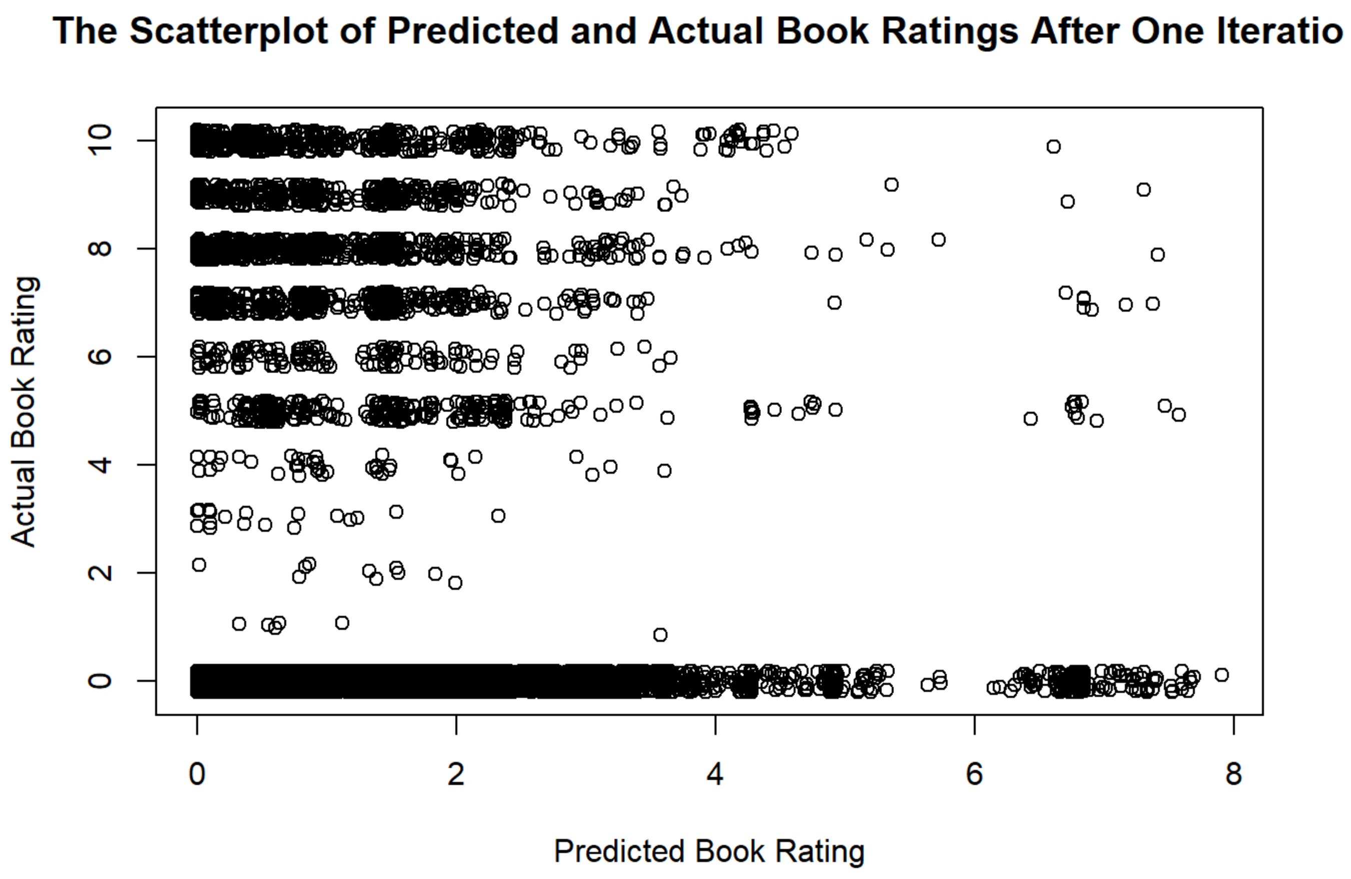
Apply the Iterative SVD technique (with 2 iterations) to potentially improve the performance of your recommendation system. Estimate the new test error and discuss your findings. Does the iterative recommendation system perform better?

```
ReTrain2 = ReTrain1
trueindex = which(!is.na(TrainRating), arr.ind=TRUE)
ReTrain2[trueindex] = TrainRating[trueindex]

SVD2 = svd(ReTrain2)
ReTrain3 <- SVD2$u[,1:2] %*% diag(SVD2$d)[1:2,1:2] %*% t(SVD2$v[,1:2])
## Change the row and column names so as to match the training set
rownames(ReTrain3) = rownames(TrainRating1)
colnames(ReTrain3) = colnames(TrainRating1)

newReTrain3 = as.data.frame(ReTrain3)
newReTrain3$User.ID = rownames(ReTrain3)
```

```
newR2Prediction <- melt(as.matrix(newReTrain3), id.vars = User.ID,  variable.name = "ISBN",
value.name = "Book.Rating")

newR2Prediction$Book.Rating = as.numeric(as.character(newR2Prediction$Book.Rating))
names(newR2Prediction)[1:2] = c("User.ID","ISBN")
newR2Prediction$index = do.call(paste0, newR2Prediction[c("User.ID","ISBN")])

mjpredict2 = subset(newR2Prediction, newR2Prediction$index %in% test$index)
mjpredict2 = mjpredict2[order( mjpredict2$index ), ]
mjreality2 = subset(test, test$index %in% mjpredict2$index)
mjreality2 = mjreality2[order( mjpredict2$index ), ]

diff2 = mjpredict2$Book.Rating - mjreality2$Book.Rating
```

After one iteration, the Mean Square Error for this recommendation system using the test set is 9.1939454, which indicates that, given the ten-point Likert rating scale, the performance of this recommendation is still not accurate enough. This is also evidenced from the Scatterplot shown as follows where the predicted book ratings are on the x-axis while the actual rating on the y-axis.

```
plot(mjpredict2$Book.Rating , jitter(mjreality2$Book.Rating), main = "The Scatterplot of Pr
edicted and Actual Book Ratings After One Iteration", xlab = "Predicted Book Rating", ylab
= "Actual Book Rating")
```



Then I have another iteration with similar procedure:

```
ReTrain4 =  ReTrain3
ReTrain4[trueindex] = TrainRating[trueindex]

SVD4 = svd(ReTrain4)
ReTrain5 <- SVD4$u[,1:2] %*% diag(SVD4$d)[1:2,1:2] %*% t(SVD4$v[,1:2])
## Change the row and column names so as to match the training set
rownames(ReTrain5) = rownames(TrainRating1)
colnames(ReTrain5) = colnames(TrainRating1)

newReTrain4 = as.data.frame(ReTrain5)
newReTrain4$User.ID = rownames(ReTrain5)
newR3Prediction <- melt(as.matrix(newReTrain4), id.vars = User.ID,  variable.name = "ISBN",
value.name = "Book.Rating")

newR3Prediction$Book.Rating = as.numeric(as.character(newR3Prediction$Book.Rating))
names(newR3Prediction)[1:2] = c("User.ID","ISBN")
newR3Prediction$index = do.call(paste0, newR3Prediction[c("User.ID","ISBN")])

mjpredict3 = subset(newR3Prediction, newR3Prediction$index %in% test$index)
```

```
mjpredict3 = mjpredict3[order( mjpredict3$index ), ]
mjreality3 = subset(test, test$index %in% mjpredict3$index)
mjreality3 = mjreality3[order( mjpredict3$index ), ]

diff3 = mjpredict3$Book.Rating - mjreality3$Book.Rating
```

After two iterations, the Mean Square Error for this recommendation system using the test set is 9.214577, which indicates that, given the ten-point Likert rating scale, the performance of this recommendation is still not accurate enough. In addition, the improvement from these two iterations are quite incremental. To conclude, the recommendation algorithm with two iterations do not significantly outperform the SVD algorithm.

Task 6

6.1 Brief Discussion

Given your analysis briely discuss how you could improve the performance of the above recommendation system.

Answer:

The Rating Matrix for the Training Set has 6,773,100 elements in total. However, the training set has only 100,000 observtions, which means that 98.5235712 percent of the elements in the Rating matrix are missing values. What's more, there are only 11138 unique ISBN that exist in both the training and test sets, which means that for the other 17392 books in the test set, we do not have any rating information from the training set.

On top of these constraints, three main methods to improve the performance of this SVD algorithm are

i. Increase the number of iterations. When the number of iterations increases to 30, the MSE could be reduce to less than 9.

ii. Increase the size of the Training Set. In this way, the sparsity of the Rating Matrix will be allevaited so that more accurate predictions could be made.

iii. Make the training set closer to the Test Set in terms of uique ISBNs. We can gain confidence of making precise predictions only when there is sufficiently large amount of relevant information to learn from the training set.