DATA-643 Assignment - 03

Mohamed Elmoudni
Shazia Khan
Senthil Dhanapal

Contents

Requirements:

Project 03

The goal of this assignment is give you practice working with Singular Value Decomposition. Your task is implement a matrix factorization method-such as singular value decomposition (SVD) or Alternating Least Squares (ALS)-in the context of a recommender system. You may approach this in a large number of ways. You are welcome to start with an existing recommender system written by yourself or someone else (always citing your sources, so that you can be graded on what you added, not what you found). Here is one example. Suppose you start with (or create) a collaborative filtering system against (a subset of) the MovieLens database or our toy dataset. You could create a content-based system, where you populate your item profiles by pulling text information for specific movies from a source like imdb, applying text processing techniques (like TF-IDF), then using SVD and topic modeling to create a set of features derived from the text. An extra intermediate step could be to take text that was pre-classified, e.g. "fighting" or "singing" and build out two "explainable" features. SVD builds features that may or may not map neatly to movie genres or news topics. You may work in a small group (2 or 3 people) on this assignment.

Introduction

Recommendation Using SVD

The goal of CF-based recommendation algorithms is to suggest new products or to predict the utility of a product for a customer, based on the customer's previous behavior and other similar customers 'opinions. However, these systems have some problems like sparsity, scalability, and synonymy. The weakness of CF algorithms for large, sparse databases led the researchers to alternative ways. In order to remove noise data from a large and sparse database, some dimensionality reduction techniques are proposed. Latent Semantic Indexing (LSI), which is a dimensionality reduction technique that used in information retrieval (IR), is a widely used technique to reduce the dimensionality of user-item ratings matrix. LSI, which uses singular value decomposition (SVD) as its underlying dimension reduction algorithm, maps nicely into the collaborative filtering recommender algorithm challenge

Toy Dataset

First let's show an example of dimension reduction, consider the rating matrix A

Table 1: Original Matrix

	Movie1	Movie2	Movie3	Movie4	Movie5
User1	1	5	0	5	4
User2	5	4	4	3	2
User3	0	4	0	0	5
User4	4	4	1	4	0
User5	0	4	3	5	0
User6	2	4	3	5	3

Applying SVD to Toy matrix A

Table 2: SVD, U Matrix

-0.4599561	0.3964320	0.3000847	-0.4318831	0.3225994	-0.4599561
-0.4608062	-0.3069318	-0.6474524	0.2836982	0.0197989	-0.4608062
-0.2487966	0.7546451	-0.2790170	0.1562937	-0.4645854	-0.2487966
-0.3834392	-0.3454649	-0.1318702	-0.6833112	-0.3218441	-0.3834392
-0.3762169	-0.2443748	0.6212002	0.3802641	-0.5020945	-0.3762169
-0.4750090	-0.0089592	0.0981056	0.3115279	0.5692235	-0.4750090

Table 3: Diagonal Matrix, S

16.46644	0.000000	0.000000	0.000000	0.000000
0.00000	6.210013	0.000000	0.000000	0.000000
0.00000	0.000000	4.399085	0.000000	0.000000
0.00000	0.000000	0.000000	2.903364	0.000000
0.00000	0.000000	0.000000	0.000000	1.584456
0.00000	0.000000	0.000000	0.000000	0.000000

Table 4: SVD, V transpose Matrix

-0.3186943	-0.6119629	-0.2903081	-0.5752356	-0.3297887
-0.4086956	0.2218704	-0.3757148	-0.2555815	0.7597749
-0.7429835	0.0327995	-0.1281545	0.5971956	-0.2717235
-0.3869928	-0.1258970	0.8703205	-0.2006657	0.1914731
0.1720872	-0.7478965	-0.0260415	0.4548327	0.4510941

Matrix U (6x6), matrix S (6x5), and matrix V (5x5) are calculated. Now, we will collapse this matrix from a (6x5) space into a 2-Dimensional one. To do this, we simply take the first two columns of U, S and V. The end result

For a recommender system based on SVD, here is one very simple strategy: find the most similar user using the 2-Dimensional matrixes above with one of the similarity calculation algorithms and compare his/her items against that of the new user; take the items that the similar user has rated and the new user has not and return them for the new user. Similar to this, for a new item, find the most similar item using the 2-Dimensional matrixes above with one of the similarity calculation algorithms and compare the users rated similar item against the new item; take the users that rate similar item but not the new item and return the ratings for the new item.

Testing using SVD in Toy dataset

Selecting k = 2

Table 5: SVD for k =2

1.4075937	5.181121	1.2737945	3.727541	4.3682179
3.1971956	4.220580	2.9189424	4.851947	1.0542144
-0.6096677	3.546850	-0.5714007	1.158876	4.9116525
2.8889890	3.387872	2.6390068	4.180278	0.4522691

2.5945209	3.454378	2.3686189	3.951421	0.8900136
2.5154726	4.774251	2.2916088	4.513545	2.5372394

New user with rating: 0,0,2,0,4

```
##             [,1]       [,2]
## [1,] -0.1504557 0.08419174
```

Selecting k = 3

Table 6: SVD for k =3

0.4267829	5.224419	1.1046180	4.5158972	4.0095163
5.3133595	4.127161	3.2839519	3.1510159	1.8281366
0.3022846	3.506591	-0.4141014	0.4258669	5.2451711
3.3199999	3.368845	2.7133503	3.8338400	0.6098981
0.5641610	3.544009	2.0184095	5.5833848	0.1474716
2.1948197	4.788406	2.2363005	4.7712798	2.4199704

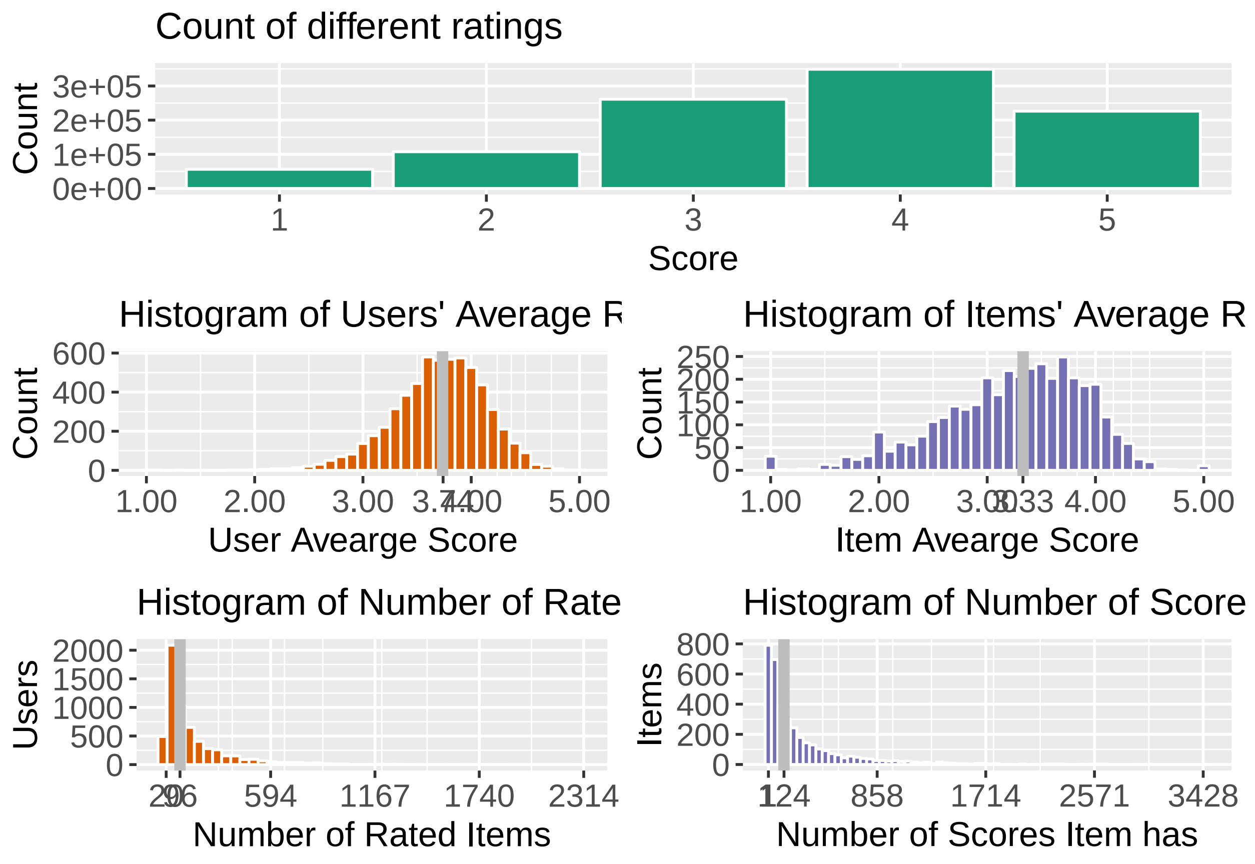
New user with rating: 0,0,2,0,4

```
##             [,1]       [,2]     [,3]
## [1,] -0.1504557 0.08419174 0.460294
```

MovieLense dataset

Now that we have a basic understanding on how to apply SVD in our toy data, let's apply it in the to bigger set such the Movie Lens dataset.
We will be using the MovieLense dataset that comes with the recommenderlab package.

Data Exploration



First,let's convert the data into sparse format.

```
## 10 x 10 sparse Matrix of class "dgCMatrix"
##
## u1  5 . . . . . . . . .
## u2  . . . . . . . . . .
## u3  . . . . . . . . . .
## u4  . . . . . . . . . .
## u5  . . . . . 2 . . . .
## u6  4 . . . . . . . . .
## u7  . . . . . 4 . . . .
## u8  4 . . 3 . . . . . .
## u9  5 . . . . . . . . .
## u10 5 5 . . . . 4 . . .
```

Creating a realRatingMatrix object from sparse matrix:

```
## 6040 x 3706 rating matrix of class 'realRatingMatrix' with 1000209 ratings.
```

Model Creation without SVD

Creating the model:

Prediction of ratings for first 5 users:

```
##          m1       m2       m3       m4       m5
## u1       NA 3.864537 3.748432 3.489864 3.768903
## u2 4.192675 3.389036 3.272931 3.014363 3.293402
## u3 4.381458 3.577818 3.461714 3.203145 3.482184
## u4 4.669973 3.866334 3.750229 3.491661 3.770700
## u5 3.625962 2.822322 2.706218 2.447649 2.726688
```

RMSE without SVD

Calculation of RMSE

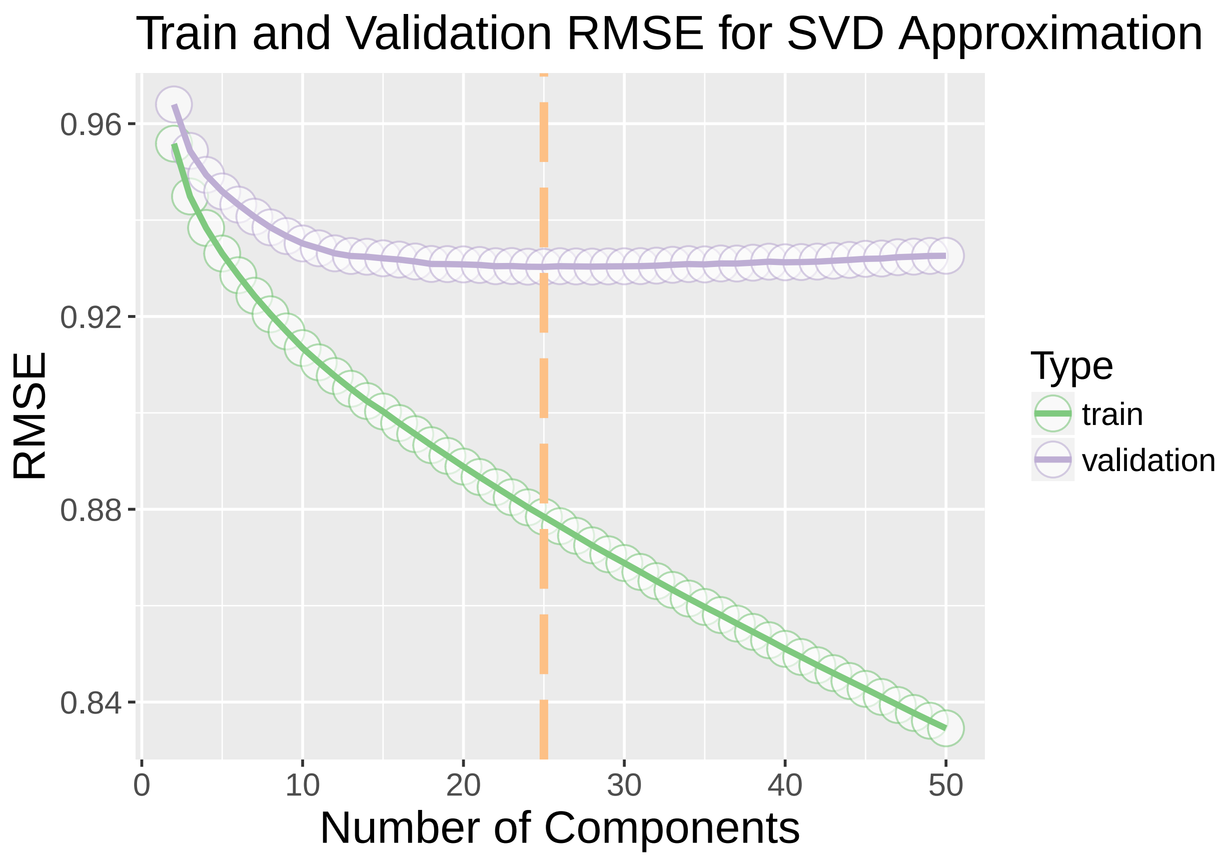
```
##      RMSE
## 0.9610123
```

Therefore the RMSE is 0.9610123

Model creation using SVD

Model tuning for different value of k (r in the R package)

```
## Model is evaluated for r = 2
## Model is evaluated for r = 3
## Model is evaluated for r = 4
## Model is evaluated for r = 5
## Model is evaluated for r = 6
## Model is evaluated for r = 7
## Model is evaluated for r = 8
## Model is evaluated for r = 9
## Model is evaluated for r = 10
## Model is evaluated for r = 11
## Model is evaluated for r = 12
## Model is evaluated for r = 13
## Model is evaluated for r = 14
## Model is evaluated for r = 15
## Model is evaluated for r = 16
## Model is evaluated for r = 17
## Model is evaluated for r = 18
## Model is evaluated for r = 19
## Model is evaluated for r = 20
## Model is evaluated for r = 21
## Model is evaluated for r = 22
## Model is evaluated for r = 23
## Model is evaluated for r = 24
## Model is evaluated for r = 25
## Model is evaluated for r = 26
## Model is evaluated for r = 27
## Model is evaluated for r = 28
## Model is evaluated for r = 29
## Model is evaluated for r = 30
## Model is evaluated for r = 31
## Model is evaluated for r = 32
## Model is evaluated for r = 33
## Model is evaluated for r = 34
## Model is evaluated for r = 35
## Model is evaluated for r = 36
## Model is evaluated for r = 37
## Model is evaluated for r = 38
## Model is evaluated for r = 39
## Model is evaluated for r = 40
## Model is evaluated for r = 41
## Model is evaluated for r = 42
## Model is evaluated for r = 43
## Model is evaluated for r = 44
## Model is evaluated for r = 45
## Model is evaluated for r = 46
## Model is evaluated for r = 47
## Model is evaluated for r = 48
## Model is evaluated for r = 49
## Model is evaluated for r = 50
```



RMSE using SVD

Creating RMSE using SVD:

```
## [1] 0.9313392
```

Therefore the RMSE usng SVD is 0.9313392

Conclusion

From the results above, we clearly see that the performance of the model on the MovieLens dataset using SVD is better than the model without SVD. The RMSE using SVD is 0.9313392 while the RMSE without SVD is 0.9610123 . However, choosing the value of k in the diagonal matrix S needs to be evaluated very carefully.

Appendix A: DATA643 Assignment 03 R Code

```
if (!require("ggplot2",character.only = TRUE)) (install.packages("ggplot2",repos = "http://cran.us.r-pro
if (!require("recommenderlab",character.only = TRUE)) (install.packages("ggplot2",repos = "http://cran.u
if (!require("reshape2",character.only = TRUE)) (install.packages("ggplot2",repos = "http://cran.us.r-pr
if (!require("knitr",character.only = TRUE)) (install.packages("ggplot2",repos = "http://cran.us.r-proj

library(recommenderlab)
library(reshape2)
library(ggplot2)
library(knitr)
library(recommenderlab)
library(recosystem)
library(SlopeOne)
library(SVDApproximation)
library(data.table)
library(RColorBrewer)
library(ggplot2)


User1 <- c(1, 5, 0, 5, 4)
User2<- c(5, 4, 4, 3, 2)
User3<- c(0, 4, 0, 0, 5)
User4<- c(4, 4, 1, 4, 0)
User5<- c(0, 4, 3, 5, 0)
User6<- c(2, 4, 3, 5, 3 )

A<- rbind(User1,User2, User3, User4, User5, User6)
colnames(A)<- c("Movie1","Movie2", "Movie3", "Movie4", "Movie5")
kable(A, caption = 'Original Matrix')


A.svd <- svd(A)
U<- matrix(A.svd$u,6,6)
kable(U, caption = 'SVD, U Matrix')
S<- diag(A.svd$d, 6,5)
kable(S, caption = 'Diagonal Matrix, S')
V_t<- t(A.svd$v)
#V_t
kable(V_t, caption = 'SVD, V transpose Matrix')

svd <- svd(A,nu=2,nv=2)
S <- diag(svd$d[1:2])
kable(svd$u %*% S %*% t(svd$v),  caption = 'SVD for k =2')

b <-matrix(c(0,0,2,0,4,1),byrow=T, ncol=6)
S1 <- diag(1/svd$d[1:2])
b %*% svd$u %*% S1


svd <- svd(A,nu=3,nv=3)
S <- diag(svd$d[1:3])
kable(svd$u %*% S %*% t(svd$v),  caption = 'SVD for k =3')
```

```r
b <-matrix(c(0,0,2,0,4,1),byrow=T, ncol=6)
S1 <- diag(1/svd$d[1:3])
b %*% svd$u %*% S1



set.seed(1)
data("MovieLense")



#set.seed(1)
mtx <- split_ratings(ratings_table = ratings,
               proportion = c(0.7, 0.15, 0.15))


visualize_ratings(ratings_table = ratings, color_palett = "Dark2")


sparse_ratings <- sparseMatrix(i = ratings$user, j = ratings$item, x = ratings$rating,
                              dims = c(length(unique(ratings$user)), length(unique(ratings$item))),
                              dimnames = list(paste("u", 1:length(unique(ratings$user)), sep = ""),
                                              paste("m", 1:length(unique(ratings$item)), sep = "")))
sparse_ratings[1:10, 1:10]
real_ratings <- new("realRatingMatrix", data = sparse_ratings)
real_ratings


model <- Recommender(real_ratings, method = "POPULAR", param=list(normalize = "center"))

prediction <- predict(model, real_ratings[1:5], type="ratings")
as(prediction, "matrix")[,1:5]

set.seed(1)
e <- evaluationScheme(real_ratings, method="split", train=0.8, given=-5)
#5 ratings of 20% of users are excluded for testing

model <- Recommender(getData(e, "train"), "POPULAR")
prediction <- predict(model, getData(e, "known"), type="ratings")

rmse_popular <- calcPredictionAccuracy(prediction, getData(e, "unknown"))[1]
rmse_popular

model <- svd_build(mtx)


model_tunes <- svd_tune(model, r = 2:50)
model_tunes$train_vs_valid
rmse_svd <- svd_rmse(model, r = model_tunes$r_best, rmse_type = c("test"))
rmse_svd

##
```