Matrix and Tensor Factorization Tutorial taught by Prof. Dr. Panagiotis Symeonidis

A. System’s Components and other Material

1. install R (https://cran.r-project.org/)

2. install RStudio (https://www.rstudio.com/products/rstudio/download/)

3. You have to download the rrecsys package (downloadable from CRAN with the command install.package(“rrecsys”) on the R console).

4. You have also to install 3 more packages i.e, for Non-negative Matrix Factorization, for CUR decomposition, and for Tensor decomposition by using the following commands, respectively.

```
install.packages("NMF")
library("NMF")

install.packages("rCUR")
library("rCUR")

install.packages("rTensor")
library("rTensor")
```

5. Download my book «Matrix and Tensor decomposition in Recommender Systems»

   a. You can download it from the following SpringerLink website:

      http://link.springer.com/book/10.1007%2F978-3-319-41357-0

   b. Moreover, you can download the slides that accompany the book and can be used for lecturing purposes from the following link:

      http://www.inf.unibz.it/~symeonidis/slidesForBook.pdf

Commands for Matrix and Tensor Factorization in R (TUTORIAL)

```
getwd()
```

```
setwd("/Users/psymeonidis/Desktop/Summer School 2017")
```

(in my Book, toy example data, Figure 1.2 page 5)

```
mymatrix <- matrix(scan('toyexample.txt'), nrow=3, byrow=TRUE)
```

~~mymatrix2 <- matrix(scan('toyexample2.txt', na.strings = "NA"), nrow=4, byrow=TRUE)~~

Non-Negative Matrix Factorization

(in my book the solution can be found in Figure 2.2 page 24)

```
install.packages("NMF")
```

```
library("NMF")
```

```
myfactors <- nmf(mymatrix, 2)
```

```
w <- myfactors@fit@W
```

```
h <- myfactors@fit@H
```

```
approximation_matrix_nmf <- w %*% h
```

Singular Value Decomposition

(in my book the solution can be found in Figure 3.2 page 37)

```
mymodel <- svd(mymatrix)

 original_matrix_svd <- mymodel$u %*% diag(mymodel$d) %*% t(mymodel$v)

variance.explained = prop.table(svd(mymatrix)$d^2)

approximation_matrix_svd <- mymodel$u[, 1:2] %*% diag(mymodel$d[1:2]) %*% t(mymodel$v[, 1:2])
```

CUR matrix decomposition

(My book, Solution Figure 2.3 page 29)

```
mymodel <- CUR(mymatrix, 2, 2 )

approximation_matrix_cur <- mymodel@C %*% mymodel@U %*% mymodel@R
```

rrecsys package

```
install.packages("rrecsys")

library(rrecsys)

data("ml100k")



d <- defineData(ml100k)

d
summary(d)
dataChart(d)
histogram(d)
```

```
e <- evalModel(d, folds = 2)

e
```

Evaluating User-based knn and Item-based knn

```
evalPred(e, "ubknn", simFunct = "cos", neigh = 10)
evalPred(e, "ibknn", simFunct = "cos", neigh = 10)

resUB <- evalRec(e, "ubknn", simFunct = "cos", neigh = 10, positiveThreshold = 3,
topN = 10, topNGen = "mf")

resIB <- evalRec(e, "ibknn", simFunct = "cos", neigh = 10, positiveThreshold = 3,
topN = 10, topNGen = "mf")
```

Evaluating Matrix Factorization (Simon Funk SVD)

```
evalPred(e, "funk", k = 10, steps = 100, regCoef = 0.0001, learningRate = 0.001,
biases = F)

evalPred(e, "funk", k = 20, steps = 100, regCoef = 0.0001, learningRate = 0.001,
biases = F)

evalPred(e, "funk", k = 100, steps = 100, regCoef = 0.0001, learningRate = 0.001,
biases = F)


resFunk10 <- evalRec(e, "funk", k = 10, steps = 100, regCoef = 0.001,
learningRate = 0.01, biases = F, positiveThreshold = 3, topN = 10)

resFunk100 <- evalRec(e, "funk", k = 100, steps = 100, regCoef = 0.001,
learningRate = 0.01, biases = F, positiveThreshold = 3, topN = 10)

resFunk300 <- evalRec(e, "funk", k = 300, steps = 100, regCoef = 0.001,
learningRate = 0.01, biases = F, positiveThreshold = 3, topN = 10)
```

Visualizing User-based knn and Item-based knn behaviour

```
evalChart(resUB, x = "items", y = "num_of_recommendations", y_lim = 60)
evalChart(resIB, x = "items", y = "num_of_recommendations", y_lim = 60)


evalChart(resUB, x = "items", y = "TP", y_label = "TP", y_lim = 60)
evalChart(resIB, x = "items", y = "TP", y_label = "TP", y_lim = 60)


evalChart(resUB, x = "topN", y = "num_of_ratings", y_lim = 200)
evalChart(resIB, x = "topN", y = "num_of_ratings", y_lim = 200)
```

Visualizing Matrix Factorization behaviour

```
evalChart(resFunk10, x = "items", y = "num_of_recommendations", y_lim = 160)
evalChart(resFunk100, x = "items", y = "num_of_recommendations", y_lim = 160)
evalChart(resFun300, x = "items", y = "num_of_recommendations", y_lim = 160)

evalChart(resFunk10, x = "items", y = "TP", y_lim = 60)
evalChart(resFunk100, x = "items", y = "TP", y_lim = 60)
evalChart(resFun300, x = "items", y = "TP", y_lim = 60)

evalChart(resFunk10, x = "topN", y = "num_of_ratings")
evalChart(resFunk100, x = "topN", y = "num_of_ratings")
evalChart(resFunk300, x = "topN", y = "num_of_ratings")
```

Tensor HOSVD Decomposition

(My book, Figure 6.2 page 83)

```
vector1 <- c(1,1,0,0,0,0,0,0,0)

vector2 <- c(0,0,0,0,1,0,0,0,0)

vector3 <- c(0,0,0,0,0,0,0,0,1)
```

```
b <- array(c(vector1,vector2, vector3),dim = c(3,3,3))

print(b)

b<- as.tensor(b)

approximation_tensor_hosvd <- hosvd(b, c(2,3,3))

print(approximation_tensor_hosvd$est@data)
```

Results

```
, , 1

          [,1] [,2] [,3]
[1,] 0.7236068    0    0
[2,] 1.1708204    0    0
[3,] 0.0000000    0    0

, , 2

     [,1]  [,2] [,3]
[1,]    0 0.4472136    0
[2,]    0 0.7236068    0
[3,]    0 0.0000000    0

, , 3

     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
[3,]    0    0    1
```

Step by step execution of Tensor Decomposition Algorithm

~~(Mode1$u %*% diag(Mode1$d) %*% t(Mode1$v))~~

```
vector1 <- c(1,1,0,0,0,0,0,0,0)

vector2 <- c(0,0,0,0,1,0,0,0,0)
```

```
vector3 <- c(0,0,0,0,0,0,0,0,1)


b <- array(c(vector1,vector2, vector3),dim = c(3,3,3))

print(b)

b<- as.tensor(b)


A <- unfold(b, row_idx=1,col_idx=c(2,3))

A<-matrix(vec(A),3)
Mode1 <- svd(A)


B <- unfold(b, row_idx=2,col_idx=c(1,3))

B<-matrix(vec(B),3)
Mode2 <- svd(B)


C <- unfold(b, row_idx=3,col_idx=c(1,2))

C<-matrix(vec(C),3)
Mode3 <- svd(C)

n1 <- ttm(b, t(Mode1$u[,1:2]),1);

n2 <- ttm(n1,t(Mode2$u[,1:3]),2);

n3 <- ttm(n2,t(Mode3$u[,1:3]),3);


S <- unfold(n3,row_idx=1,col_idx=c(3,2));


q1 <- ttm(n3, Mode1$u[,1:2], 1);
q2 <- ttm(q1, Mode2$u[,1:3], 2);
q3 <- ttm(q2, Mode3$u[,1:3], 3);


q3@data
```

Results

```
, , 1

          [,1] [,2] [,3]
[1,] 0.7236068    0    0
```

```
[2,] 1.1708204    0    0
[3,] 0.0000000    0    0

, , 2

     [,1]      [,2] [,3]
[1,]    0 0.4472136    0
[2,]    0 0.7236068    0
[3,]    0 0.0000000    0

, , 3

     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
[3,]    0    0    1
```