Informe 9/12/2019

José Miguel Moya

9/12/2019

Evaluación de los algoritmos recomendadores para las listas top-N

Para le evaluación de estos algoritmos que brinda la librería `recommenderlab` utilicé los algoritmos más significativos en base a pruebas anteriores. Estos fueron:

- IBCF
- UBCF
- Popular items
- Random items
- ALS Implicit
- SVD

Según el paper de `recommenderlab` existen implementados en esta librería varios protocolos de retención que se pueden emplear en el esquema de evaluación, el cual se crea con la función `evaluationScheme()`. Estos protocolos son:

- Given 2
- Given 5
- Given 10
- All but x

Durante las pruebas, el protocolo Given x presenta el algoritmo con solo x elementos elegidos al azar para el usuario de la prueba, y el algoritmo se evalúa según la capacidad de predecir los elementos retenidos. Para All but x, una generalización de All-but-1, el algoritmo ve todas las clasificaciones retenidas excepto x para el usuario de prueba. Para el esquema de evalución, `given` controla estos protocolos. Los enteros positivos dan como resultado un protocolo dado x, mientras que los valores negativos producen un protocolo All but x.

```
set.seed(2019)

source("Create_Rating_Matrix.R")

library(recommenderlab)

ratin_matrix <- readRDS("rating_matrix.RDS")


#####################################################################
# Esquema cross validation
#####################################################################

###############
# Given 2
###############

esq_crossVal_given2 <- evaluationScheme(ratin_matrix, method = "cross",
                                        k=4,
                                        given = -2,
                                        goodRating = 4)
```

```r
algorithms <- list(
        "IBCF" = list(name = "IBCF", param = NULL),
        "UBCF" = list(name = "UBCF", param = list(nn=50)),
        "popular items" = list(name = "POPULAR", param = NULL),
        "random items" = list(name = "RANDOM", param = NULL),
        "ALS_implicit" = list(name = "ALS_implicit", param = NULL),
        "SVD approximation" = list(name = "SVD", param = list(k=50))

)

crossVal_given2 <- evaluate(esq_crossVal_given2, algorithms, type = "topNList",
                    n=c(1,3,5,10,15,20))

saveRDS(crossVal_given2,"./Esquemas/crossVal_given2.RDS")


###############
# Given 5
###############

esq_crossVal_given5 <- evaluationScheme(ratin_matrix, method = "cross",
                                        k=4,
                                        given = -5,
                                        goodRating = 4)

algorithms <- list(
        "IBCF" = list(name = "IBCF", param = NULL),
        "UBCF" = list(name = "UBCF", param = list(nn=50)),
        "popular items" = list(name = "POPULAR", param = NULL),
        "random items" = list(name = "RANDOM", param = NULL),
        "ALS_implicit" = list(name = "ALS_implicit", param = NULL),
        "SVD approximation" = list(name = "SVD", param = list(k=50))

)

crossVal_given5 <- evaluate(esq_crossVal_given5, algorithms, type = "topNList",
                            n=c(1,3,5,10,15,20))

saveRDS(crossVal_given5,"./Esquemas/crossVal_given5.RDS")



###############
# Given 10
###############

esq_crossVal_given10 <- evaluationScheme(ratin_matrix, method = "cross",
                                        k=4,
                                        given = -10,
                                        goodRating = 4)

algorithms <- list(
        "IBCF" = list(name = "IBCF", param = NULL),
```

```
        "UBCF" = list(name = "UBCF", param = list(nn=50)),
        "popular items" = list(name = "POPULAR", param = NULL),
        "random items" = list(name = "RANDOM", param = NULL),
        "ALS_implicit" = list(name = "ALS_implicit", param = NULL),
        "SVD approximation" = list(name = "SVD", param = list(k=50))

)

crossVal_given10 <- evaluate(esq_crossVal_given10, algorithms, type = "topNList",
                             n=c(1,3,5,10,15,20))

saveRDS(crossVal_given10,"./Esquemas/crossVal_given10.RDS")


###############
# All but 2
###############

esq_crossVal_all_but2 <- evaluationScheme(ratin_matrix, method = "cross",
                                          k=4,
                                          given = 2,
                                          goodRating = 4)

algorithms <- list(
        "IBCF" = list(name = "IBCF", param = NULL),
        "UBCF" = list(name = "UBCF", param = list(nn=50)),
        "popular items" = list(name = "POPULAR", param = NULL),
        "random items" = list(name = "RANDOM", param = NULL),
        "ALS_implicit" = list(name = "ALS_implicit", param = NULL),
        "SVD approximation" = list(name = "SVD", param = list(k=50))

)

crossVal_all_but2 <- evaluate(esq_crossVal_all_but2, algorithms, type = "topNList",
                              n=c(1,3,5,10,15,20))

saveRDS(crossVal_all_but2,"./Esquemas/crossVal_all_but2.RDS")


###############
# All but 5
###############

esq_crossVal_all_but5 <- evaluationScheme(ratin_matrix, method = "cross",
                                            k=4,
                                            given = 5,
                                            goodRating = 4)

algorithms <- list(
        "IBCF" = list(name = "IBCF", param = NULL),
        "UBCF" = list(name = "UBCF", param = list(nn=50)),
        "popular items" = list(name = "POPULAR", param = NULL),
        "random items" = list(name = "RANDOM", param = NULL),
```

```
        "ALS_implicit" = list(name = "ALS_implicit", param = NULL),
        "SVD approximation" = list(name = "SVD", param = list(k=50))

)

crossVal_all_but5 <- evaluate(esq_crossVal_all_but5, algorithms, type = "topNList",
                              n=c(1,3,5,10,15,20))

saveRDS(crossVal_all_but5,"./Esquemas/crossVal_all_but5.RDS")


###############
# All but 10
###############

esq_crossVal_all_but10 <- evaluationScheme(ratin_matrix, method = "cross",
                                           k=4,
                                           given = 10,
                                           goodRating = 4)

algorithms <- list(
        "IBCF" = list(name = "IBCF", param = NULL),
        "UBCF" = list(name = "UBCF", param = list(nn=50)),
        "popular items" = list(name = "POPULAR", param = NULL),
        "random items" = list(name = "RANDOM", param = NULL),
        "ALS_implicit" = list(name = "ALS_implicit", param = NULL),
        "SVD approximation" = list(name = "SVD", param = list(k=50))

)

crossVal_all_but10 <- evaluate(esq_crossVal_all_but10, algorithms, type = "topNList",
                              n=c(1,3,5,10,15,20))

saveRDS(crossVal_all_but10,"./Esquemas/crossVal_all_but10.RDS")
```

Resultados de las evaluaciones

Ver el script Graficar los Esquemas topN

Evaluación de los algoritmos recomendadores para los Ratings

Similar a cómo se evaluan los algoritmos para las listas top-N, se evaluan los algoritmos para los ratings. La idea es solo establecer el parámetro `type = "ratings"`.

```
set.seed(2019)

library(recommenderlab)

ratin_matrix <- readRDS("rating_matrix.RDS")

esq_eval <- evaluationScheme(ratin_matrix, method = "split",
                             train = 0.9,
                             given = -5,
```

```
                              goodRating = 4)

algorithms <- list(
        "random items" = list(name = "RANDOM", param = NULL),
        "popular items" = list(name = "POPULAR", param = NULL),
        "user-based CF" = list(name = "UBCF", param = list(nn=50)),
        "SVD approximation" = list(name = "SVD", param = list(k=50)),
        "ALS_implicit" = list(name = "ALS_implicit", param = NULL),
        "IBCF" = list(name = "IBCF", param = NULL)
)

results <- evaluate(esq_eval, algorithms, type = "ratings")

saveRDS(results, "./Esquemas/Ratings.RDS")

results <- readRDS("./Esquemas/Ratings.RDS")

lista_resultados <- as(results, "list")

RANDOM = getConfusionMatrix(lista_resultados[[1]])
POPULAR = getConfusionMatrix(lista_resultados[[2]])
UBCF = getConfusionMatrix(lista_resultados[[3]])
SVD_aprox = getConfusionMatrix(lista_resultados[[4]])
ALS_implicit = getConfusionMatrix(lista_resultados[[5]])
IBCF = getConfusionMatrix(lista_resultados[[6]])

error <- rbind(
        RANDOM = c(RANDOM[[1]][1], RANDOM[[1]][2], RANDOM[[1]][3]),
        POPULAR = c(POPULAR[[1]][1], POPULAR[[1]][2], POPULAR[[1]][3]),
        UBCF = c(UBCF[[1]][1], UBCF[[1]][2], UBCF[[1]][3]),
        SVD_aprox = c(SVD_aprox[[1]][1], SVD_aprox[[1]][2], SVD_aprox[[1]][3]),
        ALS_implicit = c(ALS_implicit[[1]][1], ALS_implicit[[1]][2], ALS_implicit[[1]][3]),
        IBCF = c(IBCF[[1]][1], IBCF[[1]][2], IBCF[[1]][3])

)

colnames(error)<- c("RMSE", "MSE", "MAE")
```

Resultados de las evaluaciones

Ver Script Evaluador de ratings

Recomendador

A continuación el procedimiento seguido para crear un recomendador de películas basado en la selección de tres películas aleatorias. El recomendador toma como entrada:

- Un objeto `Recommender`
- Una semilla para seleccionar la muestra de las películas
- Número de películas de la muestra (fijado a 3)
- Los Ratings asignados a esas películas (objeto tipo vector)

Esto es con la idea de que el usuario seleccione tres películas de su preferencia y las califique y en base a eso recomendarle 5 películas.

Cargar todos los datos necesarios

```
library(recommenderlab)

source("Read Files of Movies.R")

movies <- ReadFiles()$Movies
moviesId <- movies$movieId

rating_matrix <- readRDS("rating_matrix.RDS")

}
```

Leer los Recomendadores previamente construidos

```
recomendadores <- readRDS("./Recomendadores/Recomendadores.RDS")


SVD_aprox <- recomendadores$SVD_aprox
IBCB <- recomendadores$IBCF
UBCF <- recomendadores$UBCF
POPULAR <- recomendadores$POPULAR
RANDOM <- recomendadores$RANDOM
ALS_implicit <- recomendadores$ALS_implicit
```

Crear la función que muestrea las películas

```
muestreador <- function(semilla, n=3, estrellas){
        set.seed(semilla)
        muestra <- movies[sample(nrow(movies), size=n),]
        muestra$userId <- c(611, 611, 611)
        muestra$rating <- estrellas
        return(muestra)
}
```

Crear la función recomendadora

```
Recomendar_Pelicula <- function(recomendador, semilla, n, estrellas){

        prueba <- muestreador(semilla, n, estrellas)

        matriz <- matrix(NA, nrow = 1, ncol = 9724)

        matriz[1,which(moviesId==prueba$movieId[1])]<- prueba$rating[1]
        matriz[1,which(moviesId==prueba$movieId[2])]<- prueba$rating[2]
        matriz[1,which(moviesId==prueba$movieId[3])]<- prueba$rating[3]

        rating_prueba <- as(matriz, "realRatingMatrix")

        pred <- predict(recomendador, rating_prueba, n=5)

        p_ratings <- predict(recomendador, rating_prueba, type = "ratings")
        p_matrix_rating <- as(p_ratings, "matrix")
```

```
        id_movies <- as.integer(as(pred, "list")[[1]])

        ratings_list <- numeric()
        ratings_list[1] <-  p_matrix_rating[1,which(id_movies[1]==moviesId)]
        ratings_list[2] <-  p_matrix_rating[1,which(id_movies[2]==moviesId)]
        ratings_list[3] <-  p_matrix_rating[1,which(id_movies[3]==moviesId)]
        ratings_list[4] <-  p_matrix_rating[1,which(id_movies[4]==moviesId)]
        ratings_list[5] <-  p_matrix_rating[1,which(id_movies[5]==moviesId)]

        library(dplyr)
        top5_movies <- movies %>%
                select(movieId, title, genres) %>%
                filter( (movieId== id_movies[1]) | (movieId==id_movies[2]) |
                                (movieId== id_movies[3]) | (movieId== id_movies[4]) |
                                (movieId== id_movies[5])) %>%
                mutate(ratings = ratings_list)

        top5_movies_ordenados <- top5_movies[order(match(top5_movies[,1], id_movies)),]

        print("En base a la elección y los Ratings de estas películas")
        print("")
        print(prueba[,c(2,3,5)])
        print("")
        print("Las recomendaciones para ud según sus gustos son:")
        top5_movies_ordenados

}
```

Problemas detectados hasta el momento

1. A la hora de recomendar películas con los métodos IBCF y UBCF **si los ratings introducidos son todos iguales** no devuelve resultado alguno. El error al parecer está a la hora de utilizar la función `predict(IBCF/UBCF, newData)`. Según mis observaciones cuando guardo esta predicción en una variable y luego la convierto a lista no devuelve los ID de las películas recomendadas.
2. Como habíamos comentado el método **ALS implicit** parece ser uno de los mejores que se desempeña, pero su tiempo de predicción es *demasiado* para la idea que se tiene de una aplicación interactiva en línea.