Ajustes de hiperparámetros a los algoritmos UBCF y SVD

José Miguel Moya

17/12/2019

Introducción

Como parte de las observaciones realizadas por ustedes en lo referente a los algoritmos de recomendación, me dediqué a realizar un ajuste de los hiperparámetros de los algoritmos UBCF y SVD. Este ajuste se llevó a cabo con el objetivo de tratar de mejorar la precisión y el recall de ambos algoritmos con los datos que tenemos. Los resultados obtenidos serán presentados a continuación.

Esquema de evaluación

Para todas las pruebas que se realizaron se seleccionó un esquema de evaluación de tipo cross-validation con 4 folds. La configuración es la mostrada a continuación:

```
esquema <- evaluationScheme(rating_matrix, method = "cross",
                            k=4,
                            given = 10,
                            goodRating = 4)
```

Ajuste de hiperparámetros para el algoritmo UBCF

El algoritmo UBCF cuenta con cuatro hiperparámetros:

- **nn**: Número de vecinos más cercanos.
- **method**: El método para calcular la similitud entre los vecinos más cercanos que se seleccionen. Este cuenta con dos opciones, *jaccard* y *cosine*.
- **normalize**: Para seleccionar el tipo de normalización de los datos. La librería `recommenderlab` cuenta con tres métodos de normalización, *center*, *Z-score*, y uno basado en boostrap. Para esta selección se probaron *center*(por default) y *Z-score*.
- **sample**: Este es un parámetro que por defecto viene deshabilitado en la implementación del UBCF. Para este ajuste no se utilizó pues lo que hace es tomar una muestra de los datos para realizar la construcción del algoritmo.

En base a estos hiperparámetros y sus posibles valores se trabajó con el objetivo de alcanzar la mayor precisión y recall.

Probando diferentes valores de *nn* con los demás hiperparámetros por defecto

```
esq_eva_nn <- evaluationScheme(rating_matrix, method = "cross",
                            k=4,
                            given = 10,
                            goodRating = 4)

algoritmos_nn <- list(
        "UBCF20" = list(name = "UBCF", param = list(nn=20)),
        "UBCF25" = list(name = "UBCF", param = list(nn=25)),
```

```
        "UBCF30" = list(name = "UBCF", param = list(nn=30)),
        "UBCF35" = list(name = "UBCF", param = list(nn=35)),
        "UBCF40" = list(name = "UBCF", param = list(nn=40)),
        "UBCF45" = list(name = "UBCF", param = list(nn=45)),
        "UBCF50" = list(name = "UBCF", param = list(nn=50)),
        "UBCF55" = list(name = "UBCF", param = list(nn=55)),
        "UBCF60" = list(name = "UBCF", param = list(nn=60)),
        "UBCF65" = list(name = "UBCF", param = list(nn=65)),
        "UBCF70" = list(name = "UBCF", param = list(nn=70))
)

evaluacion_nn <- as(
        evaluate(esq_eva_nn, algoritmos_nn, type = "topNList", n=5),
        "list"
)
```

Funciones que servirán para obtener el promedio de la precisión y el recall de cada uno de los 4 folds

```
##################
# Precision
##################

cal_mean_prec <- function(lista){
        acumulado <- 0
        for (i in (1:4)) {
                acumulado <- acumulado + lista[[i]][5]
        }
        return(acumulado/4)
}

mean_prec_list <- function(lista){
        precisiones <- numeric()
        for (i in (1:11)) {
                precisiones[i] <- cal_mean_prec(getConfusionMatrix(lista[[i]]))

        }

        return(precisiones)

}

###################
# Recall
##################
cal_mean_recall <- function(lista){
        acumulado <- 0
        for (i in (1:4)) {
                acumulado <- acumulado + lista[[i]][6]
        }
        return(acumulado/4)
}
```

```
mean_recall_list <- function(lista){
        recal <- numeric()
        for (i in (1:11)) {
                recal[i] <- cal_mean_recall(getConfusionMatrix(lista[[i]]))

        }

        return(recal)

}
```

Uniendo los resultados en una matriz

```
########################################
# Juntando los resultados en una matriz
########################################

prec_recall_nn <- matrix(
        c(mean_prec_list(evaluacion_nn), mean_recall_list(evaluacion_nn)),
        ncol = 2,
        dimnames = list(c("UBCF20","UBCF25","UBCF30","UBCF35",
                          "UBCF40","UBCF45","UBCF50","UBCF55",
                          "UBCF60","UBCF65","UBCF70"),
                        c("Precision","Recall"))
); prec_recall_nn
```

Probando diferentes valores de *method* con los demás hiperparámetros por defecto

```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Evaluacion del hiperparámetro method
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

esq_eva_method <- evaluationScheme(rating_matrix, method = "cross",
                                   k=4,
                                   given = 10,
                                   goodRating = 4)

algoritmos_method <- list(
        "UBCF_cosine" = list(name = "UBCF", param = list(method = "cosine")),
        "UBCF_jaccard" = list(name = "UBCF", param = list(method = "jaccard"))
)

evaluacion_method <- as(
        evaluate(esq_eva_method, algoritmos_method, type = "topNList", n=5),
        "list"
)
```

Las funciones para calcular el promedio de la precisión y el recall solo presentan pequeñas variaciones y su funcionalidad es análoga a las presentadas con anterioridad. Para más detalles ver el Script `Configuraciones UBCF.R`.

Probando diferentes valores de *normalize* con los demás hiperparámetros por defecto

```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Evaluacion del hiperparámetro normalize
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

esq_eva_normalize <- evaluationScheme(rating_matrix, method = "cross",
                                      k=4,
                                      given = 10,
                                      goodRating = 4)

algoritmos_normalize <- list(
        "UBCF_center" = list(name = "UBCF", param = list(normalize = "center")),
        "UBCF_Z_score" = list(name = "UBCF", param = list(normalize = "Z-score"))
)

evaluacion_normalize <- as(
        evaluate(esq_eva_normalize, algoritmos_normalize, type = "topNList", n=5),
        "list"
)
```

Resultados preliminares

Las pruebas individuales arrojaron que los mejores hiperparámetros son *nn=70*, *method="jaccard"* y *normalize="Z-score"*

Probando diferentes combinaciones de hiperparámetros

A continuación se empleran diferentes combinaciones de hiperparámetros para determinar la combinación que arroje mejores resultados.

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Probando diferentes configuraciones
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

esq_eva_configuraciones <- evaluationScheme(rating_matrix, method = "cross",
                                            k=4,
                                            given = 10,
                                            goodRating = 4)
algoritmos_configuraciones <- list(
        "UBCF1" = list(name = "UBCF", param = list(nn=70, method="jaccard",normalize = "center" )),
        "UBCF2" = list(name = "UBCF", param = list(nn=70, method="jaccard",normalize = "Z-score" )),
        "UBCF3" = list(name = "UBCF", param = list(nn=70, method="cosine",normalize = "center" )),
        "UBCF4" = list(name = "UBCF", param = list(nn=70, method="cosine",normalize = "Z-score" )),
        "UBCF5" = list(name = "UBCF", param = list(nn=60, method="jaccard",normalize = "center" )),
        "UBCF6" = list(name = "UBCF", param = list(nn=60, method="jaccard",normalize = "Z-score" )),
        "UBCF7" = list(name = "UBCF", param = list(nn=60, method="cosine",normalize = "center" )),
        "UBCF8" = list(name = "UBCF", param = list(nn=60, method="cosine",normalize = "Z-score" )),
        "UBCF9" = list(name = "UBCF", param = list(nn=80, method="jaccard",normalize = "center" )),
        "UBCF10" = list(name = "UBCF", param = list(nn=80, method="jaccard",normalize = "Z-score" )),
        "UBCF11" = list(name = "UBCF", param = list(nn=80, method="cosine",normalize = "center" )),
        "UBCF12" = list(name = "UBCF", param = list(nn=80, method="cosine",normalize = "Z-score" ))
)
```

```
evaluacion_configuraciones <- as(
        evaluate(esq_eva_configuraciones, algoritmos_configuraciones,
                 type = "topNList", n=5),
        "list"
)
```

Conclusiones sobre el algoritmo UBCF

La combinación que arroja mejores resultados es UBCF2 con hiperparámetros *nn=70*, *method = "jaccard"* y *normalize="Z-score"*. Como se puede observar sí difiere de la que anteriormente se utilizó para crear el recomendador UBCF.

Los detalles sobre la implementación y los resultados pueden ser obtenidos corriendo el Script `Configuraciones UBCF.R` el cual se encuentra disponibe en el repositorio GitHub del proyecto.

Ajuste de hiperparámetros para el algoritmo SVD

El algoritmo SVD cuenta con tres hiperparámetros.

- **k**: Es el rango de aproximación.
- **maxiter**: Es el número máximo de iteraciones SVD.
- **normalize**: Análogo al descrito para el algoritmo UBCF.

En base a estos hiperparámetros y sus posibles valores se trabajó con el objetivo de alcanzar la mayor precisión y recall.

Probando diferentes valores de *k* con los demás hiperparámetros por defecto

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

algotithms <- list(
        "SVD10" = list(name = "SVD", param = list(k=10)),
        "SVD20" = list(name = "SVD", param = list(k=20)),
        "SVD30" = list(name = "SVD", param = list(k=30)),
        "SVD40" = list(name = "SVD", param = list(k=40)),
        "SVD50" = list(name = "SVD", param = list(k=50))
)

evaluacion <- evaluate(esq_eval, algotithms, type = "topNList", n=5)

evaluacion_list <- as(evaluacion, "list")
```

Las funciones para calcular el promedio de la precisión y el recall son análogas a las empleadas en la descripción del algoritmo UBCF. Los detalles de la implementación se pueden encontar en el Script `Configuraciones SVD.R`.

Probando diferentes valores de *maxiter* con los demás hiperparámetros por defecto

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Evaluando el hiperparámetro maxiter
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

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

algotithms_maxiter <- list(
        "SVD100" = list(name = "SVD", param = list(maxiter=100)),
        "SVD200" = list(name = "SVD", param = list(maxiter=200)),
        "SVD300" = list(name = "SVD", param = list(maxiter=300)),
        "SVD400" = list(name = "SVD", param = list(maxiter=400)),
        "SVD500" = list(name = "SVD", param = list(maxiter=500))
)

evaluacion_maxiter <- evaluate(esq_eval_maxiter, algotithms_maxiter,
                               type = "topNList", n=5)

evaluacion_list_maxiter <- as(evaluacion_maxiter, "list")
```

Probando diferentes valores de *normalize* con los demás hiperparámetros por defecto

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Evaluando el hiperparámetro normalize
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

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

algotithms_normalize <- list(
        "SVD_center" = list(name = "SVD", param = list(normalize="center")),
        "SVD_Z_score" = list(name = "SVD", param = list(normalize="Z-score"))

)

evaluacion_normalize <- evaluate(esq_eval_normalize, algotithms_normalize,
                               type = "topNList", n=5)

evaluacion_list_normalize <- as(evaluacion_normalize, "list")
```

Resultados preliminares

Las pruebas individuales arrojaron que los mejores hiperparámetros son *k=40*, *maxiter=300* y *normalize="Z-score"*

Probando diferentes combinaciones de hiperparámetros

A continuación se empleran diferentes combinaciones de hiperparámetros para determinar la combinación que arroje mejores resultados.

```
esq_eval <- evaluationScheme(ratin_matrix, method = "cross",
                             k=4,
                             given = 10,
                             goodRating = 4)
algoritmos <- list(
        "SVD1" = list(name = "SVD", param = list(k=40, maxiter=300, normalize="center")),
        "SVD2" = list(name = "SVD", param = list(k=40, maxiter=300, normalize="Z-score")),
        "SVD3" = list(name = "SVD", param = list(k=50, maxiter=300, normalize="center")),
        "SVD4" = list(name = "SVD", param = list(k=50, maxiter=300, normalize="Z-score")),
        "SVD5" = list(name = "SVD", param = list(k=60, maxiter=300, normalize="center")),
        "SVD6" = list(name = "SVD", param = list(k=60, maxiter=300, normalize="Z-score")),
        "SVD7" = list(name = "SVD", param = list(k=70, maxiter=300, normalize="center")),
        "SVD8" = list(name = "SVD", param = list(k=70, maxiter=300, normalize="Z-score")),
        "SVD9" = list(name = "SVD", param = list(k=40, maxiter=200, normalize="center")),
        "SVD10" = list(name = "SVD", param = list(k=40, maxiter=200, normalize="Z-score"))
)

evaluacion <- as(
        evaluate(esq_eval, algoritmos, type = "topNList", n=5),
        "list"
)
```

Conclusiones sobre el algoritmo SVD

La combinación que arroja mejores resultados es *k=40*, *maxiter=200/300* y *normalize="Z-score"*. Como se puede observar sí difiere de la que anteriormente se utilizó para crear el recomendador SVD.

Los detalles sobre la implementación y los resultados pueden ser obtenidos corriendo el Script `Configuraciones SVD.R` el cual se encuentra disponibe en el repositorio GitHub del proyecto.

Cambios a la función recomendadora de películas

En base a las sugerencias realizadas por ustedes se actualizó el Script `DEMO Objetivo.R`. Este contaba con dos funciones `Seleccionar_peliculas()` y `Recomendar_Pelicula`. Estas funciones fueron actualizadas con el objetivo de recibir ahora 5 películas y realizar las recomendaciones pertinentes. Tengan en cuenta que se contruyeron dos recomendadores *UBCF* y *SVD_aprox*. Estos recomendadores ya están construidos con la combinación de hiperparámetros que arrojó los mejores resultados en cuanto a precisión y recall. Los recomendadores pueden ser cargados desde la carpeta raíz del repositorio como sigue:

```
UBCF <- readRDS("UBCF.RDS")
SVD_aprox <- readRDS("SVD_aprox.RDS")
```

Sugerencias y/o cambios

En base a los cambios y las actualizaciones realizadas me gustaría que volvieran a testear el recomendador y volver a envíar los resultados con observaciones y/o cambios que se deban hacer.

La aplicación Shiny ya está montada en un DEMO, obviamente aún no está con estos nuevos recomendadores sino que está empleando el SVD_aprox antiguo, con solo tres películas de entrada. Quedo a la espera de sus observaciones e indicaciones para definir el siguiente paso.