Modelos lineales con SGD

Victor Gallego y Roi Naveiro

09/04/2019

```
library(rvw)
library(dplyr)
```

En este ejercicio, utilizaremos la librería Vowpal Wabbit para entrenar variantes de un modelo lineal.

Primero descargamos y leemos los datos desde http://www.grouplens.org/system/files/ml-100k.zip. Nos interesan ua.base (training set) y ua.test (test set). Los datos constan de 4 columnas: usuario, item, rating (de 1 a 5) y timestamp del evento (descartable)

```
system('wget http://www.grouplens.org/system/files/ml-100k.zip')
system('unzip ml-100k.zip')
recs_train <- read.csv('ml-100k/ua.base', sep = '\t',col.names = c('user', 'item', 'rating', 'time'))
recs_train <- select (recs_train,-c(time))

recs_test <- read.csv('ml-100k/ua.test', sep = '\t',col.names = c('user', 'item', 'rating', 'time'))
recs_test <-  select (recs_test,-c(time))
```

Utilizando la función object_size del paquete pryr, calcula lo que ocupa en memoria el training set, antes y después de convertir a variables dummies.

```
library(fastDummies)
# install.packages('pryr')
library(pryr)

recs_dum <- fastDummies::dummy_cols(recs_train, select_columns = c('user', 'item'))

print(object_size(recs_train))
print(object_size(recs_dum))
```

Ajustar un modelo lineal llamando a vw(). Como hiperparámetros escoge: * épocas de SGD: 3 * bits de hashing: 18 * learning rate: 0.005 * regularización L1: 0.0001

Calcular el RMSE (o AUC si convertiste a clasificación).

```
model_lin <- vw(training_data = recs_train, validation_data = recs_test, model = 'rec_in.vw',
                target = "rating", passes=3, b=18, loss='squared',
                link_function = "--link=identity" ,use_perf = FALSE,
                l1=0.0001,learning_rate = 0.005,
                plot_roc = FALSE, do_evaluation = FALSE, extra = '')
rmse_lin <- sqrt(mean((recs_test$rating - model_lin$data)**2))
```

Ahora ajusta un modelo con interacciones cuadráticas y calcula el nuevo RMSE. Compara el tiempo de train con el anterior Utiliza el argumento extra de vw.

```
model_quad <- vw(training_data = recs_train, validation_data = recs_test, model = 'rec_quad.vw',
        target = "rating", passes=3, b=18, loss='squared',
        link_function = "--link=identity" ,use_perf = FALSE,
        l1=0.0001,learning_rate = 0.005,
        plot_roc = FALSE, do_evaluation = FALSE, extra = '-q ::')
rmse_quad <- sqrt(mean((recs_test$rating - model_quad$data)**2))
```

Ahora ajustamos una FM con dimensión latente = 10.

```
model_lrq <- vw(training_data = recs_train, validation_data = recs_test, model = 'rec_lrq2.vw',
                  target = "rating", passes=3, b=18, loss='squared',
                link_function = "--link=identity" ,use_perf = FALSE,
                l1=0.0001,learning_rate = 0.005,
                plot_roc = FALSE, do_evaluation = FALSE, extra = '--lrq ::10')
rmse_lrq <- sqrt(mean((recs_test$rating - model_lrq$data)**2))
```