Sistema de Recomendación | Evaluación de Modelos

Josseline Perdomo

Mayo 2016

Introducción

Para esta Tarea 4 se nos dio la labor de realizar los siguientes apartados:

1. Implementar un sistema de recomendación para un periódico digital, recibiendo las transacciones de los usuarios en su portal web.
2. Implementar una función generadora de curvas ROC de la siguiente forma:

```
generate_ROC <- function(scores, real, target){
  # Aqui algoritmo para generar curva
}
```

Con el proposito de realizar un estudio práctico de los **Sistemas de Recomendación** y la **Evaluación de Modelos**.

Sistema de Recomendación

En el primer escenario, un periodico virtual nos da un dataset limpio con la información de las transacciones de su portal y nos solicita las siguientes demandas a resolver:

1. Un análisis exploratorio pertinente para detectar las transacciones bot que los clientes sospechan de su existencia.
2. Modificar el dataset para que los items pertenecientes a las transacciones fueran por su tipo de contenido.
3. Recomendar a un nuevo usuario que ingresa al portal un articulo.
4. Conocer los tipos de usuarios que ingresan al portal.
5. Conocer las 10 visitas con mayor y las 10 con menor tiempo de estadía en el portal.
6. Conocer las 10 transacciones con mayor número de apariciones en el dataset.

Todos estos requerimientos se realizan a continuación:

Instalando los paquetes necesarios

Se implementó una función auxiliar en caso de que algún paquete adicional no esté instalado.

```
include <- function(packages){
    for(pkg in packages){
        # Si ya está instalado, no lo instala.
        if(!require(pkg, character.only = T)){
            install.packages(pkg, repos = "https://cran.rstudio.com", dependencies = T)
            if(!require(pkg, character.only = T))
```

```
                stop(paste("load failure:", pkg))
        }
        library(pkg, character.only = T)
    }
}

include("arules")
```

Inicializando estructuras base

```
setwd("..") # Directorio padre del proyecto
# Cargando dataset provisto
dataset <- read.csv2("dat/periodico.csv", header = T,
                     sep = ",",
                     colClasses = "character",
                     nrows = 131300,
                     comment.char = "",
                     stringsAsFactors = F)
# Áreas de información
subjects <- c("deportes", "politica",
              "variedades", "internacional",
              "nacionales", "sucesos",
              "comunidad", "negocios",
              "opinion")
head(dataset, n = 5)
```

```
##   X     ID               entry                exit              articles
## 1 1 trans1 2016-05-02 22:39:52 2016-05-02 22:49:08  {item1,item9,item63}
## 2 2 trans2 2016-05-02 17:38:55 2016-05-02 17:53:29   {item1,item2,item3}
## 3 3 trans3 2016-05-01 06:57:57 2016-05-01 07:00:44 {item9,item43,item57}
## 4 4 trans4 2016-05-01 09:10:07 2016-05-01 09:15:16 {item2,item14,item72}
## 5 5 trans5 2016-05-01 00:28:29 2016-05-01 01:01:16              {item11}
```

A pesar de que el dataset se encuentra libre de datos erróneos, realizamos un **preprocesamiento** para la preparación de los datos de acuerdo a los requerimientos a cumplir.

Preprocesamiento

Transformando *X* de character a integer:

```
dataset$X <- as.integer(dataset$X)
```

Transformando columnas *entry* y *exit*

Es necesario tranformarlas a **tipo date con formato (YYYY-MM-DD hh-mm-ss)**, para poder operar sobre estas columnas.

```
dataset$entry <- strptime(dataset$entry, "%Y-%m-%d %H:%M:%S")
dataset$exit <- strptime(dataset$exit, "%Y-%m-%d %H:%M:%S")
```

Creando columna *time*

Diferencia de *entry* - *exit* en segundos.

```
dataset$time <- difftime(dataset$exit, dataset$entry, unit = "secs")
```

Definiendo ID de las transacciones

Hay 2 columnas que pueden ser el id univoco de las transacciones: *ID* y *X*, cada transacción (a pesar de tener el mismo valor) son entradas distintas del dataset, por lo que de acuerdo al contexto del problema (Web log) no deben haber entradas repetidas (con mismo id).

Viendo la cantidad de transacciones y cuantas de ellas trasacciones son únicas:

```
rows <- nrow(dataset)
nrow(unique(dataset[,c("entry","exit","articles")]))
```

```
## [1] 131300
```

```
length(unique(dataset$ID)) == rows
```

```
## [1] FALSE
```

```
length(unique(dataset$X)) == rows
```

```
## [1] TRUE
```

Como la cantidad de transacciones usando la columna ID no es igual al total, podemos concluir que la columna *X* es la columna de los id unívocos, por lo que cambiamos su nombre:

```
colnames(dataset)[1] <- "tid"
```

Elminando columnas que se van a utilizar más durante la implementación.

Estas columnas no aportaran información relevante para los venideros requisitos a llevar a cabo.

```
dataset$ID <- NULL
dataset$entry <- NULL
dataset$exit <- NULL
```

Requrimientos

1. Transformando el dataset

En total, existen 81 items, 9 artículos por cada tema. Utilizando expresiones regulares transformaremos de la notación **item{N}** a **{subject}/articulo{n}**:

```
items_id <- seq(1, 81)
items <- sprintf("%s/articulo%d", subjects[(items_id-1)%/%9+1], (items_id-1)%%9+1)
for(i in items_id)  dataset$articles <- gsub(sprintf("item%d(?=[,}])", i),
                                             items[i],
                                             dataset$articles,
                                             perl = T)
# Eliminando {}
dataset$articles <- gsub("\\{|\\}", "", dataset$articles)

head(dataset$articles, n = 5)
```

```
## [1] "deportes/articulo1,deportes/articulo9,comunidad/articulo9"
## [2] "deportes/articulo1,deportes/articulo2,deportes/articulo3"
## [3] "deportes/articulo9,nacionales/articulo7,comunidad/articulo3"
## [4] "deportes/articulo2,politica/articulo5,negocios/articulo9"
## [5] "politica/articulo2"
```

2. Transacciones Bot

El periódico acepta que una transacción no es realizada por un *Bot*, cuando una persona dura más de 20 segundos en un artículo. Dado que no podemos asegurar con nigún conocimiento previo que durante el tiempo de la transacción la persona vio por mas de 20 segundos cada articulo, se toma el caso promedio. Por tanto, para verificar que una transacción no proviene de un bot, ésta debe durar al menos 20 segundos por la cantidad de articulos, es decir,

$$x > articles \times 20$$

Para llevar esto acabo, se hizo lo siguiente:

Se crea una lista de transacciones como vectores y se verifica la fórmula anteriormente descrita.

```
trans <- strsplit(dataset$articles, ",")
tol <- 20 # Cantidad de tiempo en segundos mínimo para no ser considerado bot
bots <- dataset$time <= (lengths(trans)*tol)
```

Luego de tener cuales no cumplen la condición, se cuenta las transaciones que son bot en el dataset.

```
nrow(dataset[bots,])
```

```
## [1] 6599
```

```
length(unique(dataset[bots, "articles"]))
```

```
## [1] 5445
```

El periódico no dice que se tenía que hacer con las transacciones bot, pero se asume que la opción mas viable es eliminarlas para que no perturben el sistema de recomendación que se creará.

```
dataset <- dataset[bots == F,]
trans <- trans[bots == F]
```

Evaluación de Modelo: Curva ROC

Este generador fue desarrollado de acuerdo a la exlicación del paper *An introduction to ROC analysis* por *Tom Fawcett*.

Parámetros

1. Los scores por instancia (no necesariamente ordenados).
2. La verdadera clase de las instancias.
3. La clase target. En el caso de que nclass > 2 entonces el enfoque es *1 vs all*.

Algoritmo

```
generate_ROC <- function (scores, real, target){
    data <- data.frame(y = real, score = scores)
    # Ordenando instancias por score descendientemente
    data <- data[with(data, order(score, decreasing = T)),]
    # Matriz de confusión, columna de valores positivos
    p_confusion <- setNames(c(0,0), c("TP", "FP"))
    # Cantidad total de valores de la clase P y N
    target_entries <- data$y == target
    real_v <- setNames(c(length(data$y[target_entries]),
                         length(data$y[!target_entries])),
                       c("P", "N"))
    prev <- Inf
    roc_curve <- data.frame(x = double(), y = numeric(), score = numeric())
    indices <- seq(1, nrow(data))

    for(i in indices){
        if(data$score[i] != prev){
            roc_curve[nrow(roc_curve)+1,] <- c(p_confusion["FP"]/real_v["N"],
                                               p_confusion["TP"]/real_v["P"],prev)
            prev <- data$score[i]
        }
        if(data$y[i] == target)
          p_confusion["TP"] <- p_confusion["TP"] + 1
        else
            p_confusion["FP"] <- p_confusion["FP"] + 1
    }
    # Insertando último punto (1,1)
    roc_curve[nrow(roc_curve)+1,] <- c(p_confusion["FP"]/real_v["N"],
                                       p_confusion["TP"]/real_v["P"],prev)

    return(roc_curve)
}

# Probando función
y <- c(2, 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 1, 1, 1, 2, 1, 1, 1)
scores <- c(0.9, 0.8, 0.7, 0.6, 0.55, 0.54, 0.53, 0.52, 0.5, 0.5, 0.5,
            0.5, 0.38, 0.37, 0.36, 0.35, 0.34, 0.33, 0.30, 0.1)
target <- 2 # Clase considerada "positiva"
```

```r
curve <- generate_ROC(scores, y, target)

# Graficando la curva retornada
plot(curve$x, curve$y,
     type ="l",
     lty = 2,
     xlab = "FP-Rate",
     ylab = "TP-Rate",
     main = "ROC Curve")
abline(0, 1, lty = 2, col = "darkgray")
points(curve$x, curve$y, col = "red", pch = 19)
text(curve$x, curve$y, curve$score, cex = 0.85, pos = 4)
```