Clasificación ordinal

Juan Ignacio Isern Ghosn
Universidad de Granada
Minería de datos: Aspectos avanzados

07/02/2019

Contents

Clasificación ordinal

En este documento se presenta la primera parte de la práctica del tema *Non-standard problems* de la asignatura de Minería de datos: Aspectos avanzados. Concretamente, en este documento se trata la clasificación ordinal.

Carga de librerías necesarias

Se cargan aquellas librerías necesarias para nuestro análisis:

```
library(RWeka)
library(plyr)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(xgboost)
```

Carga de los conjuntos de datos a utilizar

Se cargan los dataset para los cuales se realiza el análisis en este documento:

```
dataset_esl <- RWeka::read.arff("../Material/esl.arff")
dataset_lev <- RWeka::read.arff("../Material/lev.arff")
```

Etiquetas de las clases

Se guardan las etiquetas de cada una de las clases para cada dataset:

```
labels_esl <- range(dataset_esl[,ncol(dataset_esl)])[1]:range(dataset_esl[,ncol(dataset_esl)])[2]
labels_lev <- range(dataset_lev[,ncol(dataset_lev)])[1]:range(dataset_lev[,ncol(dataset_lev)])[2]
labels_esl
```

```
## [1] 1 2 3 4 5 6 7 8 9
```

```
labels_lev
```

```
## [1] 0 1 2 3 4
```

Se establecen como factor la variable correspondiente a las etiquetas

```
dataset_esl[,ncol(dataset_esl)] <- factor(dataset_esl[,ncol(dataset_esl)])
dataset_lev[,ncol(dataset_lev)] <- factor(dataset_lev[,ncol(dataset_lev)])
```

Creación de conjuntos de train y test

Para evaluar nuestros modelos, se divide nuestro dataset en train y test

```
# Para conjunto esl
index_esl <- createDataPartition(dataset_esl[,ncol(dataset_esl)], p = 0.7, list = FALSE)
train_esl <- dataset_esl[index_esl,]
test_esl <- dataset_esl[-index_esl,]
# Para conjunto lev
index_lev <- createDataPartition(dataset_lev[,ncol(dataset_lev)], p = 0.7, list = FALSE)
```

```
train_lev <- dataset_lev[index_lev,]
test_lev <- dataset_lev[-index_lev,]
```

Etiquetas binarias para clasificación ordinal

Se deben generar nuevas etiquetas binarias para entrenar cada uno de los modelos, pues actualmente tenemos una única variable que guarda todas las etiquetas, así, generamos la siguiente función:

```
ordinal_labels <- function(data, class_indx){
    data[,class_indx] <- as.integer(revalue(as.factor(data[,class_indx])))
    classes <- sort(unique(data[,class_indx]))
    data_aux <- data[,-class_indx]
    for(class in classes[-length(classes)]){
        data_aux <- cbind(data_aux, "Target" = ifelse(data[, class_indx] <= class,0,1))
        names(data_aux)[length(names(data_aux))] <- paste0("Target", class)
    }
    return(data_aux)
}
```

El resultado obtenido es el siguiente:

```
data_lab_esl <- ordinal_labels(train_esl, ncol(train_esl))
data_lab_lev <- ordinal_labels(train_lev, ncol(train_lev))
head(data_lab_lev)
```

```
##   In1 In2 In3 In4 Target1 Target2 Target3 Target4
## 1   4   2   3   0       1       1       1       0
## 2   3   3   0   3       1       1       1       0
## 4   2   1   2   3       1       1       0       0
## 6   0   1   1   0       0       0       0       0
## 7   1   2   2   1       1       1       0       0
## 8   4   2   4   3       1       1       1       0
```

Modelos de clasificación binarios

Del mismo modo, es necesario generar modelos para clasificar cada una de las nuevas etiquetas binarias generadas. Así, se genera la siguiente función que devuelve un conjunto de modelos del algoritmo indicado, de tamaño igual al número de clases menos uno:

```
ordinal_models <- function(data, nlab, alg, form){
    models <- list()
    lab_vars <- paste0("Target",1:(nlab-1))
    for(lab_var in lab_vars){
        data_aux <- cbind(data[,!(names(data) %in% lab_vars)],"C"=factor(data[,lab_var]))
        tc <- trainControl(method = "none")
        model <- caret::train(form, data_aux, method = alg, trControl = tc)
        models[[lab_var]] <-  model
    }
    models
}
```

Se generan los conjuntos de modelos necesarios para cada uno de los datasets. En nuestro caso, escogeremos el algoritmo random forest (rf en caret), pudiendo elegir cualquier otro presente en la librería caret:

```r
models_esl <- ordinal_models(data_lab_esl, length(labels_esl), "rf", C~.)
models_lev <- ordinal_models(data_lab_lev, length(labels_lev), "rf", C~.)
```

Predicción

Para llevar a cabo la predicción de esta clasificación ordinal, se sigue el algoritmo facilitado en el material de la asignatura. Así, la función resultante se muestra a continuación:

```r
ordinal_predict <- function(models, newdata, labels=c(1:(length(models)+1))){
    Pr1 <- predict(models[[1]], newdata, type="prob")$`0`
    res <- data.frame(matrix(0, ncol = 0, nrow = nrow(newdata)))
    res <- cbind(res, Pr1)
    for(i in 2:(length(models))){
        PrI_1 <- predict(models[[i-1]], newdata, type="prob")$`1`
        PrI <- predict(models[[i]], newdata, type="prob")$`0`
        PrI <- PrI_1*PrI
        res <- cbind(res, PrI)
    }
    PrK <- predict(models[[length(models)]], newdata, type="prob")$`1`
    res <- cbind(res, PrK)
    apply(res, MARGIN = 1, function(x){
        labels[which.max(x)]
    })
}
```

Así, se predice para cada dataset:

```r
pred_esl <- ordinal_predict(models_esl, test_esl[,-ncol(test_esl)], labels = labels_esl)
pred_lev <- ordinal_predict(models_lev, test_lev[,-ncol(test_lev)], labels = labels_lev)
head(pred_esl, 30)
```

```
##  [1] 4 3 5 1 5 5 6 5 4 6 6 4 6 5 6 3 6 5 5 7 5 4 6 6 5 5 5 4 7 7
```

Resultados

Los resultados concretos de la ejecución de todos los pasos anteriores se muestran a continuación, mostrándose primero para el dataset **esl**:

```r
sum(pred_esl == test_esl[,ncol(test_esl)])/nrow(test_esl)
```

```
## [1] 0.6478873
```

Y para el dataset **lev**:

```r
sum(pred_lev == test_lev[,ncol(test_lev)])/nrow(test_lev)
```

```
## [1] 0.647651
```