Trabajo Regresión kNN

Raúl Varela Ferrando

15-02-2023

Antes de comenzar sería conveniente señalar que hemos duplicado todas las tablas que aparecen en el trabajo para poder formatearlas en html y en latex, utilizando el argumento "eval =" para identificar cuál estamos usando dependiendo si estamos renderizando en .pdf o en .html.

1. - Determinación de un clasificador basado en kNN ponderado para la variable objetivo "target" con los atributos o variables explicativas previamente seleccionadas, considerando la distancia euclídea.

1.1 Selecciona el mejor núcleo y el mejor k (entre 1 y 30) a través de validación cruzada.

```
# Carga y estandarización de los datos

khan <- read.table("khantrainadd2.csv", header=TRUE, sep=";")
dim(khan)
```

1.2 Obtener la estimación del error (por validación cruzada).

```
## [1]   64 2322
```

```
# El primer paso sería estandarizar las variables, empezando por las variables
# numéricas:

var.num=c(3:2310)
khan.est=as.data.frame(scale(khan[,var.num]))


# Dado que nos piden eliminar las variables Gen* que terminen en los mismos dos dígitos
# que nuestro DNI, en mi caso tendré que eliminar las terminadas en 50

var1.elim=paste("Gen", c(1:22), "50", sep="")
var.elim=list("Gen50", var1.elim)
drop = names(khan.est) %in% var.elim
khan.est=khan.est[,!drop]


# Las variables binarias dado que ya tienen valores 0 o 1 no hace falta cambiarlas,
# solo aplicarles la función scale()
```

```r
khan.est$codlog01=scale(khan$codlog01, center = FALSE)
khan.est$codlog02=scale(khan$codlog02, center = FALSE)
khan.est$codlog03=scale(khan$codlog03, center = FALSE)
khan.est$codlog04=scale(khan$codlog04, center = FALSE)
khan.est$codlog05=scale(khan$codlog05, center = FALSE)
khan.est$codlog06=scale(khan$codlog06, center = FALSE)
khan.est$codlog07=scale(khan$codlog07, center = FALSE)

# Las variables nominales son la variable "class" y la variable "cat1", ambas
# con 4 clases, por lo que las transformaremos en 4 variables dummy cada una, es decir,
# 8 variables dummy en total.

khan.est$cat1=0
khan.est$cat1[khan$cat1==1] <- 1
khan.est$cat2=0
khan.est$cat2[khan$cat1==2] <- 1
khan.est$cat3=0
khan.est$cat3[khan$cat1==3] <- 1
khan.est$cat4=0
khan.est$cat4[khan$cat1==4] <- 1

sc1=sqrt((var(khan.est$cat1)+var(khan.est$cat2)+var(khan.est$cat3)+var(khan.est$cat4))/4)
khan.est$cat1=khan.est$cat1/sc1
khan.est$cat2=khan.est$cat2/sc1
khan.est$cat3=khan.est$cat3/sc1
khan.est$cat4=khan.est$cat4/sc1

# Para la varibale "class" estableceremos las variables "class1", "class2", "class3"
# y "class4" para los casos  BL-NHL, EWS, NB y RMS respectivamente:

khan.est$class1=0
khan.est$class1[khan$class=="BL-NHL"] <- 1
khan.est$class2=0
khan.est$class2[khan$class=="EWS"] <- 1
khan.est$class3=0
khan.est$class3[khan$class=="NB"] <- 1
khan.est$class4=0
khan.est$class4[khan$class=="RMS"] <- 1

sc2=sqrt((var(khan.est$class1)+var(khan.est$class2)+var(khan.est$class3)+var(khan.est$class4))/4)
khan.est$class1=khan.est$class1/sc2
khan.est$class2=khan.est$class2/sc2
khan.est$class3=khan.est$class3/sc2
khan.est$class4=khan.est$class4/sc2

# Las variables ordinales serían las "ord1", "ord2" y "ord3", las cuáles tienen 5 clases
# a excepción de "ord3" que tiene 4 clases y las someteremos al mismo proceso de estandarización
# que las variables cuantitativas

khan.est$ord1=scale(khan$ord1)
khan.est$ord2=scale(khan$ord2)
khan.est$ord3=scale(khan$ord3)
```

```r
# Por último, normalizamos todas las variables por el procedimiento min-max.

khan.norm = data.frame("target"=khan$target,normalize.unit(khan.est))

# Aplicamos el modelo kNN ponderado en la variable "target" con 30 como máximo de k
# y distancia euclídea:

fit.train <- train.kknn(target ~ ., data = khan.norm, kmax = 30, scale="TRUE", kernel = c("triangular",

# Obtenemos el mejor núcleo y valor de k

fit.train$best.parameters # Mejores parámetros para función núcleo y valor k
```

```
## $kernel
## [1] "inv"
##
## $k
## [1] 4
```

```r
fit.train$response        # Tipo de variable respuesta (continua, ordinal, norminal)
```

```
## [1] "continuous"
```

```r
fit.train$distance        # Parámetro de la distancia de Minkowski.
```

```
## [1] 2
```

```r
# Obtenemos que el mejor núcleo es "inv" y el mejor k es k=4, por lo que ahora sólo nos
# quedaría obtener la estimación del error. Como la variable objetivo es continua,
# calcularemos el error cuadrático medio:

ajusteOPT=fit.train$fitted.values[184] # Estos son los valores ajustados para el núcleo "inv"
# y k=4.

MSE_Ponderado=mean((khan.norm$target-ajusteOPT[[1]][1:64])^2)
```

Obtenemos un error de 130.9 aproximadamente, valor que está lejos del ideal.

---

2. - Aplicación de kNN aleatorio con validación cruzada. Considerar el número de vecinos k óptimo determinado en el paso anterior y la distancia euclídea.

2.1 Obtener el soporte de cada predictor. Aplicar un método de selección de predictores basado en dicha medida.

```r
# Ahora aplicaremos el kNN aleatorio:

p=ncol(khan.norm)-1
m=trunc(sqrt(ncol(khan.norm)))
rnc=r(p,m,eta=0.95,method="binomial")
m
```

2.2 Obtener una estimación del error con la selección de atributos realizada en el paso anterior.

```
## [1] 48
```

```r
khan.rknn = rknn.cv(data=khan.norm[,-1], y=khan.norm$target, k = 4, r=rnc, mtry=m , seed=987654321)

# Para obtener el soporte de cada predictor:

khan.rknnsupport=rknnRegSupport(data=khan.norm[,-1], y=khan.norm$target, k = 4, r=rnc, mtry=m , seed=98

# Primero entramos en la etapa geométrica para identificar el paso en el que obtenemos
# mejor exactitud, y partiendo del paso justo anterior, entraremos en la etapa lineal:

khan.rknnselG = rknnBeg(data=khan.norm[,-1], y=khan.norm$target, k = 4, r=rnc, mtry=m , seed=987654321,

mejorsel1=prebestset(khan.rknnselG, criterion="mean_support")

khan.rknnselLIN = rknnBel(data=khan.norm[,mejorsel1], y=khan.norm$target, k = 4, r=rnc, mtry=m , seed=98

mejorsel=bestset(khan.rknnselLIN, criterion="mean_support")

# Una vez tenemos los atributos más importantes, creamos nuestro nuevo modelo con estos atributos
# y calculamos el error cuadrático medio:

numsel=khan.rknnselLIN$p[18]

sel_mtry=round(0.5*numsel,0)

khan.bestsel = rknn.cv(data=khan.norm[,mejorsel], y=khan.norm$target, k = 4, r=rnc, mtry=sel_mtry , seed

MSE_Aleatorio=mean((khan.norm$target-as.numeric(as.character(khan.bestsel$pred)))^2)
```

3. Realizar un estudio comparativo entre ambos resultados.

Observando los valores obtenidos para el error cuadrático medio (MSE) de los distintos modelos tenemos que el modelo asociado al kNN ponderado nos devuelve mejor resultado que el modelo kNN aleatorio, tanto antes como después de hacer la selección de los atributos óptimos para hacer la regresión. En el caso del kNN ponderado obtenemos un error de 130.86, mientras que en aleatorio obtenemos 209.84 y 157.17, antes y después de elegir los atributos óptimos respectivamente.