Regresión Lineal R - Laboratorio 2

Juanjo Sierra

28 de noviembre de 2018

Regresión Lineal con R - Laboratorio 2

En primer lugar hay que leer el dataset que se va a utilizar, ubicado en la carpeta 'Datos': `california.dat`. El dataset está en formato KEEL.

```
california = read.csv("../Datos/california.dat", header = FALSE, comment.char = "@")
head(california)
```

```
##        V1    V2 V3   V4   V5   V6   V7     V8     V9
## 1 -117.03 32.78 17 5481 1618 2957 1537 2.5707 171300
## 2 -118.23 33.80 26  239  135  165  112 1.3333 187500
## 3 -122.46 37.71 39 2076  482 1738  445 3.1958 232100
## 4 -122.06 37.94 19 4005  972 1896  893 2.5268 235700
## 5 -122.87 38.68 32 4073  718 2053  629 3.7352 228000
## 6 -122.47 37.66 18 4172  806 3226  790 5.7535 297900
```

Se asignan los nombres de las variables adecuadamente.

```
names(california) = c("Longitude", "Latitude", "HousingMedianAge",
"TotalRooms", "TotalBedrooms", "Population", "Households",
"MedianIncome", "MedianHouseValue")

head(california)
```

```
##   Longitude Latitude HousingMedianAge TotalRooms TotalBedrooms Population
## 1   -117.03    32.78               17       5481          1618       2957
## 2   -118.23    33.80               26        239           135        165
## 3   -122.46    37.71               39       2076           482       1738
## 4   -122.06    37.94               19       4005           972       1896
## 5   -122.87    38.68               32       4073           718       2053
## 6   -122.47    37.66               18       4172           806       3226
##   Households MedianIncome MedianHouseValue
## 1       1537       2.5707           171300
## 2        112       1.3333           187500
## 3        445       3.1958           232100
## 4        893       2.5268           235700
## 5        629       3.7352           228000
## 6        790       5.7535           297900
```

Aplicación de K-NN

Para poder trabajar con K-NN vamos a importar el paquete `kknn` de R.

```
require("kknn")
```

```
## Loading required package: kknn
```

Ahora podemos realizar un modelo para el conjunto de datos de California utilizando el algoritmo K-NN. Por ahora utilizamos el mismo conjunto para train y para test.

```
fitknn1 = kknn(MedianHouseValue ~ ., california, california)
names(fitknn1)
```

```
##  [1] "fitted.values" "CL"            "W"             "D"
##  [5] "C"             "prob"          "response"      "distance"
##  [9] "call"          "terms"
```

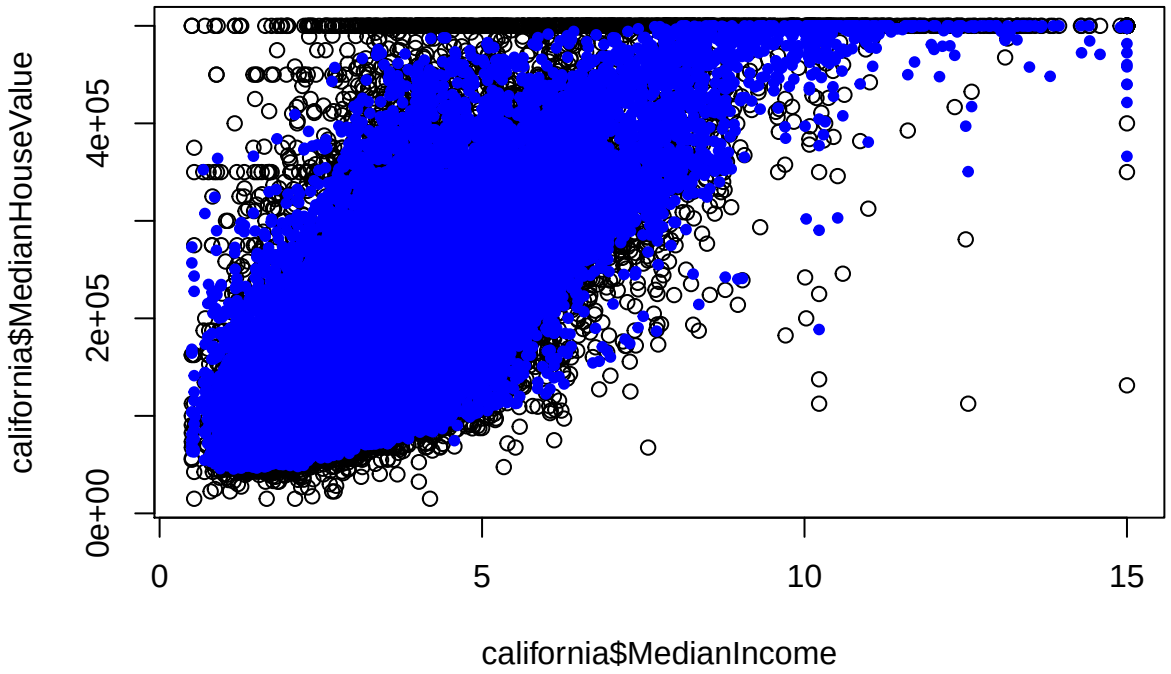
Los valores aproximados por el modelo para los ejemplos de test se pueden comprobar accediendo al atributo `fitted.values`.

```
head(fitknn1$fitted.values)
```

```
## [1] 166076.7 182252.1 213647.9 210043.7 221157.5 255370.6
```

Podemos comprobar gráficamente la similitud de estos valores con los conocidos (recordemos que por ahora train == test) mostrando ambos en la misma gráfica.

```
plot(california$MedianHouseValue~california$MedianIncome)
points(california$MedianIncome,fitknn1$fitted.values,col="blue",pch=20)
```



Se puede calcular la *Raíz del Error Cuadrático Medio* con (RMSE) manualmente con el siguiente fragmento de código:

```
yprime = fitknn1$fitted.values
sqrt(sum((california$MedianHouseValue-yprime)^2)/length(yprime))
```

```
## [1] 39131.14
```

Validación cruzada

Vamos a realizar ahora los experimentos con la técnica de validación cruzada. Para ello vamos a crear una función para leer las distintas particiones del conjunto de datos original ubicadas en la carpeta `Datos`, y poder crear modelos lineales con dichos datos. Probaremos con un modelo lineal que contenga todas las variables.

```r
nombre = "../Datos/california"
run_lm_fold <- function(i, x, tt = "test") {
  file <- paste(x, "-5-", i, "tra.dat", sep="")
  x_tra <- read.csv(file, comment.char="@")
  file <- paste(x, "-5-", i, "tst.dat", sep="")
  x_tst <- read.csv(file, comment.char="@")
  In <- length(names(x_tra)) - 1
  names(x_tra)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tra)[In+1] <- "Y"
  names(x_tst)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tst)[In+1] <- "Y"

  if (tt == "train") {
    test <- x_tra
  }
  else {
    test <- x_tst
  }

  fitMulti=lm(Y~.,x_tra)
  yprime=predict(fitMulti,test)
  sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE
}

lmMSEtrain<-mean(sapply(1:5,run_lm_fold,nombre,"train"))
lmMSEtest<-mean(sapply(1:5,run_lm_fold,nombre,"test"))

lmMSEtrain
```

```
## [1] 4826189710
```

```r
lmMSEtest
```

```
## [1] 4844365688
```

En base a los resultados obtenidos, observamos que en test se obtiene de media un error mayor que en train, que es el comportamiento habitual puesto que no se ha entrenado con dichos datos.

Comparemos con los resultados que se obtienen con el algoritmo K-NN.

```r
run_knn_fold <- function(i, x, tt = "test") {
  file <- paste(x, "-5-", i, "tra.dat", sep="")
  x_tra <- read.csv(file, comment.char="@")
  file <- paste(x, "-5-", i, "tst.dat", sep="")
  x_tst <- read.csv(file, comment.char="@")
  In <- length(names(x_tra)) - 1
  names(x_tra)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tra)[In+1] <- "Y"
  names(x_tst)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tst)[In+1] <- "Y"

  if (tt == "train") {
    test <- x_tra
  }
  else {
    test <- x_tst
```

```
  }

  fitMulti=kknn(Y~.,x_tra,test)
  yprime=fitMulti$fitted.values
  sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE
}
knnMSEtrain<-mean(sapply(1:5,run_knn_fold,nombre,"train"))
knnMSEtest<-mean(sapply(1:5,run_knn_fold,nombre,"test"))

knnMSEtrain
```

```
## [1] 1560868807
```

```
knnMSEtest
```

```
## [1] 3845914481
```

Como es entendible por la distribución de los puntos que hemos visualizado antes, K-NN obtendrá un error menor en el conjunto de entrenamiento, pero ese error difiere mucho del que hay a posteriori en el conjunto de prueba. Sin embargo, sigue siendo un error menor que el que encuentra el modelo lineal.