Tarea 2+

Dalia Camacho

August 22, 2018

Para este ejemplo usaremos los datos de https://archive.ics.uci.edu/ml/machine-learning-databases/housing/. El objetivo es predecir el valor mediano de las viviendas en áreas del censo de Estados Unidos, utilizando variables relacionadas con criminalidad, ambiente, tipo de viviendas, etc.

```
housing <- read.csv("/home/dalia/Dropbox/ITAM_MCC/Semestre_1/Aprendizaje de maquina/Tareas/Tarea 2/housi

names(housing) <- c("CRIM", "ZN", "INDUS", "CHAS", "NOX",
                    "RM", "AGE", "DIS", "RAD", "TAX", "PTRATIO",
                    "B", "LSTAT", "MEDV")

housing <- housing[complete.cases(housing$MEDV),]
```

Separa la muestra en dos partes: unos 400 para entrenamiento y el resto para prueba.

```
entrena      <- sample(1:nrow(housing),400)
Base_entrena <- housing[entrena,]
Base_prueba  <- housing[-entrena,]
```

Describe las variables en la muestra de prueba (rango, media, mediana, por ejemplo).

```
# Promedio
Prom <- colMeans(Base_prueba)

# Mínimo
Min <- unlist(lapply(1:14, function(i){min(Base_prueba[,i])}))

# Máxiimo
Max <- unlist(lapply(1:14, function(i){max(Base_prueba[,i])}))

# Mediana
Mediana <- unlist(lapply(1:14, function(i){median(Base_prueba[,i])}))

# Tabla Resumen

matrix(c("Variable",c("CRIM", "ZN", "INDUS", "CHAS",
                      "NOX", "RM", "AGE", "DIS", "RAD",
                      "TAX", "PTRATIO", "B", "LSTAT", "MEDV"),
         "Promedio", Prom, "Mediana",  Mediana,
         "Mínimo", Min, "Máximo", Max), byrow = FALSE, ncol = 5)
```

```
##       [,1]       [,2]                [,3]       [,4]      [,5]
##  [1,] "Variable" "Promedio"          "Mediana"  "Mínimo"  "Máximo"
```

```
##  [2,] "CRIM"     "1.78763403846154"   "0.343255" "0.01096" "9.91655"
##  [3,] "ZN"       "11.2307692307692"   "0"        "0"       "85"
##  [4,] "INDUS"    "11.3663461538462"   "8.14"     "2.03"    "25.65"
##  [5,] "CHAS"     "0.0769230769230769" "0"        "0"       "1"
##  [6,] "NOX"      "0.559596153846154"  "0.531"    "0.389"   "0.871"
##  [7,] "RM"       "6.18269230769231"   "6.1085"   "4.926"   "8.725"
##  [8,] "AGE"      "68.9538461538462"   "80.1"     "9.9"     "100"
##  [9,] "DIS"      "4.00054038461538"   "3.80355"  "1.1691"  "8.5353"
## [10,] "RAD"      "7.94230769230769"   "5"        "1"       "24"
## [11,] "TAX"      "372.961538461538"   "307"      "188"     "666"
## [12,] "PTRATIO"  "18.4230769230769"   "18.95"    "13"      "21.2"
## [13,] "B"        "349.858653846154"   "391.1"    "3.5"     "396.9"
## [14,] "LSTAT"    "13.4773076923077"   "12.085"   "1.98"    "30.81"
## [15,] "MEDV"     "21.4903846153846"   "20.4"     "6.3"     "50"
```

Construye un modelo lineal para predecir MEDV en términos de las otras variables. Utiliza descenso en gradiente para estimar los coeficientes con los predictores estandarizados.

```
#Estandarizar
std_entrena <- data.frame(matrix(unlist(lapply(1:14, function(i){(Base_entrena[,i]-mean(Base_entrena[,i]

names(std_entrena) <- c("CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE", "DIS", "RAD", "TAX", "PTRATI
```

```
X       <- as.matrix(cbind(rep(1,400), std_entrena[,1:13]))
Y       <- as.matrix(std_entrena[,14])
RSS     <- function(beta){
  1/2*sum((Y-X%*%beta)^2)
}

gradRSS  <- function(beta){
  t(X)%*%(X%*%beta-Y)
}
```

```
OptimBeta <- function(fun, grad, X, Y, beta, eta, maxiter, tol, fprint=FALSE){
  iter    <- 0
  toler   <- Inf
  funval  <- Inf
  while (iter<maxiter & toler > tol ) {
    beta0     <- beta
    funvalNew <- eval(fun(beta))
    beta      <- beta - eta*eval(grad(beta))
    toler     <- abs(funvalNew-funval)
    funval    <- funvalNew
    if(fprint){print(paste("Value of function",funval))}
    iter      <- iter+1
  }
  if(iter==maxiter){
    flag <- "Did not converge"
  }else{
    flag <- paste("Converged in", iter, "steps")
  }
```

```
  return(list("Value" = funval, "Beta" = beta, "Flag"=flag))
}
```

```
# Get optimized beta
beta     <- rep(0,14)
eta      <- 6.e-5
maxiter  <- 2000
tol      <- 1e-6

sol <- OptimBeta(fun = RSS, grad = gradRSS, X=X, Y=Y, beta=beta, eta = eta, maxiter = maxiter, tol = tol
```

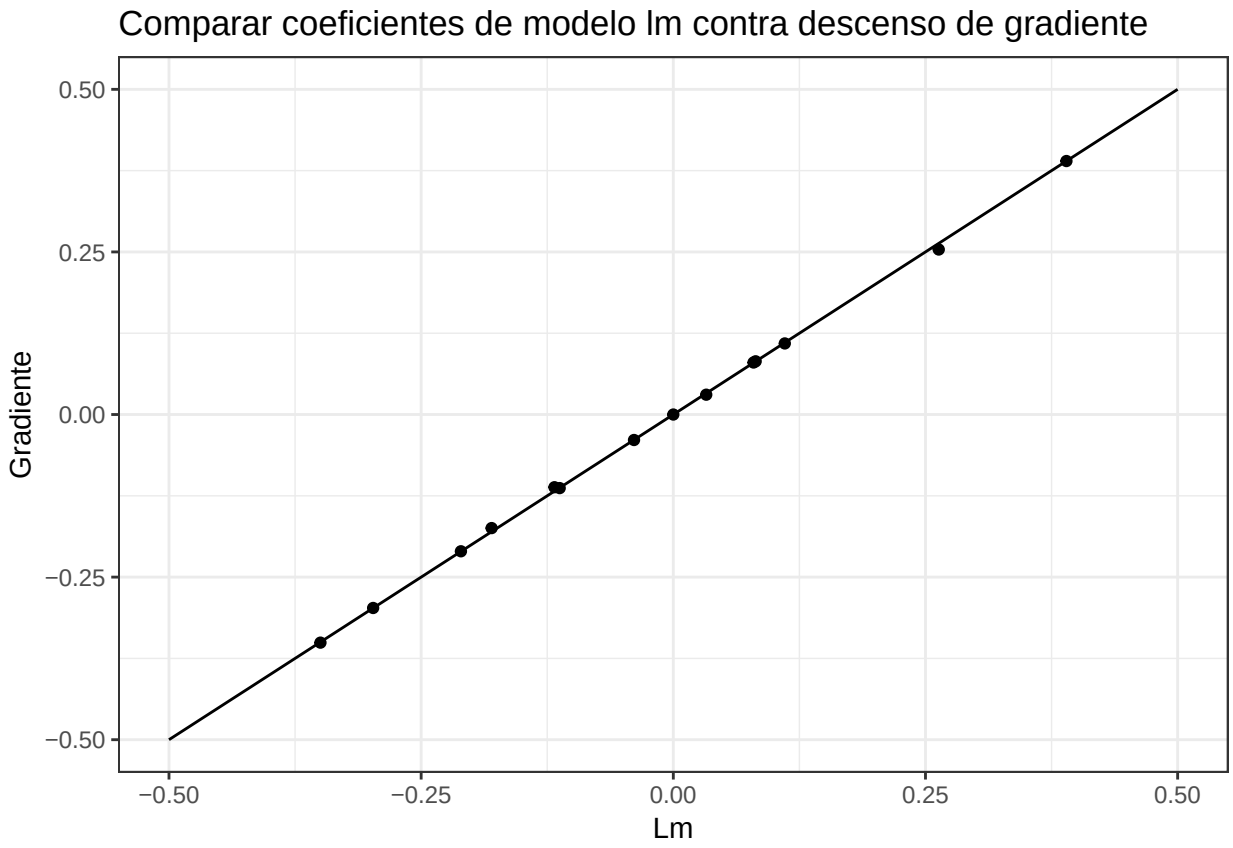
Verifica tus resultados con la función lm.

```
lmHousing <- lm(MEDV~.,data = std_entrena)

Diferencia <- formatC(t(lmHousing$coefficients-sol[[2]]), digits = 4,format = "f")
print(Diferencia)
```

```
##      rep(1, 400) CRIM      ZN       INDUS    CHAS      NOX      RM
## [1,] "-0.0000"   "-0.0061" "0.0012" "0.0022" "-0.0004" "0.0005" "-0.0000"
##      AGE      DIS       RAD      TAX       PTRATIO   B         LSTAT
## [1,] "0.0002" "-0.0002" "0.0095" "-0.0055" "-0.0003" "-0.0002" "0.0009"
```

```
ggplot()+
  geom_point(aes(lmHousing$coefficients,sol[[2]]))+
  geom_line(aes(seq(-.5,.5, length.out = 100),seq(-.5,.5, length.out = 100)))+
  theme_bw()+
  xlab("Lm")+ylab("Gradiente")+
  ggtitle("Comparar coeficientes de modelo lm contra descenso de gradiente")
```



Evalúa el error de entrenamiento y evalúa después la estimación del error de predicción Utiliza la raíz del la media de los errores al cuadrado.

```
# Desestandarizar coeficientes de beta

Beta <- c(sol[[2]][1]+sum(unlist(lapply(1:13, function(i){mean(Base_entrena[,i])/sd(Base_entrena[,i])})
          unlist(lapply(1:13, function(i){sol[[2]][i]/sd(Base_entrena[,i])})))

Error_entrena <- sqrt(mean((as.matrix(cbind(rep(1,400), Base_entrena[,1:13]))%*%Beta-
  Base_entrena$MEDV)^2))

Error_prueba <-  sqrt(mean((as.matrix(cbind(rep(1,nrow(Base_prueba)),
                                            Base_prueba[,1:13]))%*%Beta-
  Base_prueba$MEDV)^2))

Error_entrena
```

```
## [1] 17.5612
```

```
Error_prueba
```

```
## [1] 19.90298
```