Tarea de redes neuronales

Cedric Prieels

Diciembre 2016

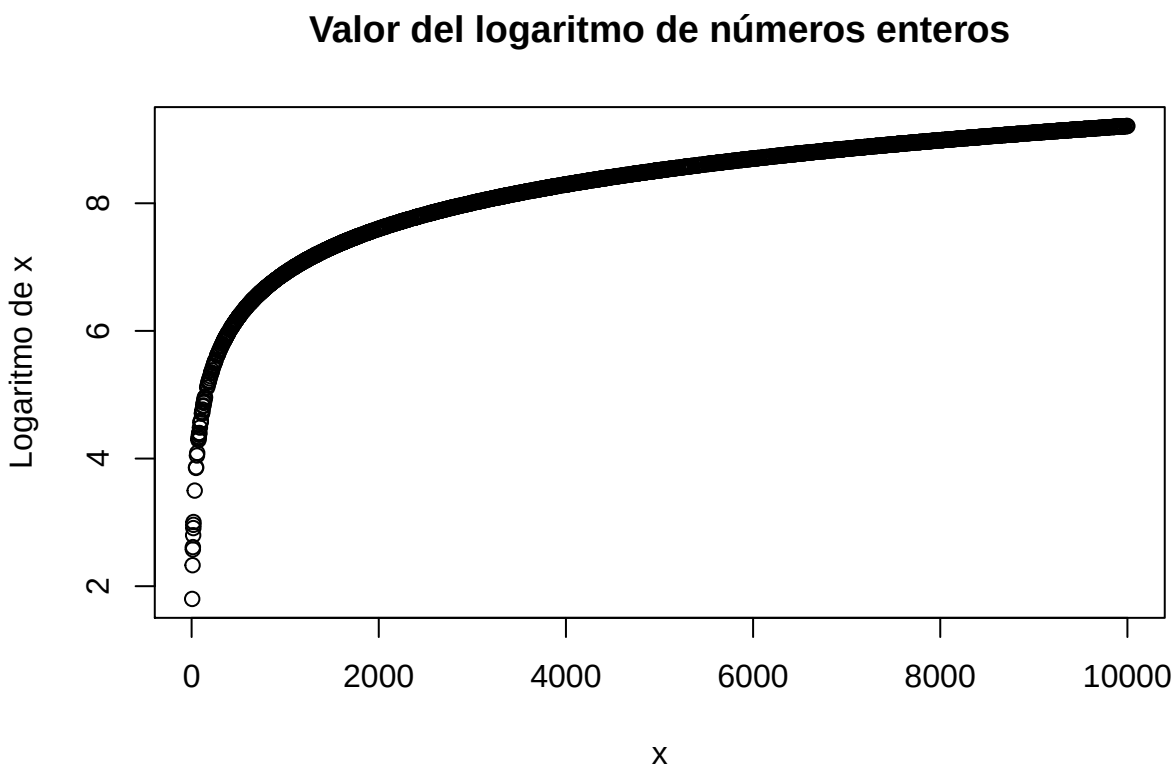
Este ejercicio consiste en entrenar una red neuronal que nos permita obtener el valor del logaritmo de un conjunto de números aleatorios, usando un método de validación cruzada visto en clase (método k-fold con k igual a 10).

Primero, borramos de la memoria las variables precedientes, definimos el directorio de trabajo y cargamos las librerias necesarias para crear y pintar redes neuronales.

```
rm(list=ls())
setwd('/Users/ced2718/Documents/Universite/Modelizacion/')
library(nnet)
source('nnet_plot_fun.r')
```

Primero, generamos 2000 números aleatorios que toman valores entre 0 y 10000, de los cuales calculamos el valor de logaritmo, usando la función *log* de R. Los pintamos y rellenamos un dataframe para poder usar la función *nnet* que nos crea una red neuronal. Estos datos van a formar el conjunto de datos que vamos a usar para entrenar y testear nuestra red, usando un método K-fold.

```
#Fijamos la semilla para obtener siempre los mismos números
set.seed(1)

par(mfrow=c(1,1))
N <- 2000
x <- runif(N, min = 0, max = 10000)
x <- sort(x)
y <- log(x)
plot(x, y, ylab="Logaritmo de x", main="Valor del logaritmo de números enteros")
```




```
dataframe <- data.frame(x,y)
```

Como ya lo hicimos en un ejercicio anterior, definimos ahora la función *mse* que no da una idea del error que cometemos con el método de validación cruzada que vamos a usar después. También definimos la función *mae* que calcula el error de manera un poco distinta, para verificar nuestros resultados.

```
#Definimos la función mse, que nos devuelve
#la media de la diferencia entre los valores del modelo y los valores medidos al cuadrado
mse <-function(obs,est){
  return(mean((obs-est)^2))
}

#Definimos otra manera de calcular el error
mae<-function(obs,est){
  return(mean(abs(obs-est)))
}
```

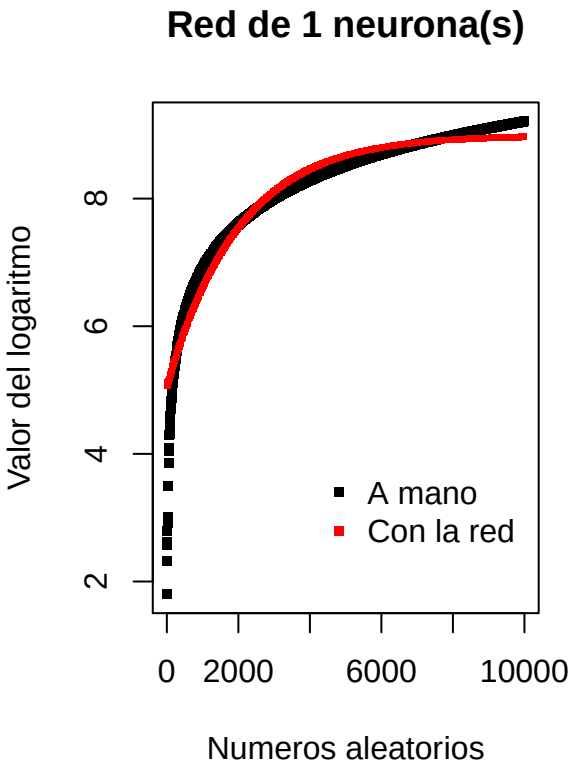
Empezamos un método de validación cruzada con K=10 (número optimal, según el artículo de Ron Kohabi), cambiando cada vez el número de neuronas usado en la red neuronal (usaremos siempre una función de activación a la salida de la red lineal en este caso). En cada paso, pintamos nuestros números que obtenemos con la red neuronal para compararlos con los números de verdad que obtenemos usando la función *log* de R y pintamos también la red que estamos usando, con sus neuronas y conexiones (el signo de los pesos relativos se representa con un color diferente en los plots, negro o gris, y el tamaño de las lineas que unen dos neuronas da una idea del valor de estos pesos).

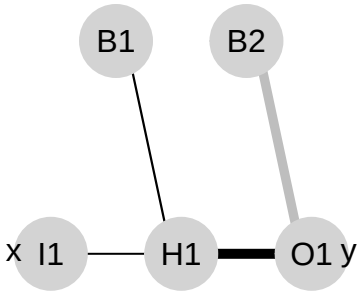
```
k <- 10
numberNeurons <- 8
size <- ceiling(N/k)
```

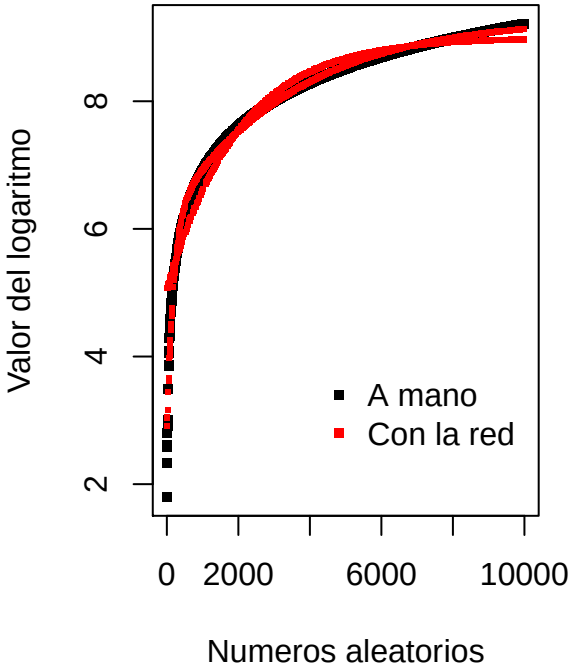
```r
idx.aleatorios <- sample(1:N, replace=F)
estimatedValues <- matrix('NA', N, numberNeurons)
mseVector <- rep('NA', numberNeurons)
maeVector <- rep('NA', numberNeurons)

par(mfrow=c(1,2))

#Consideramos diferentes números de neuronas
for(j in 1:numberNeurons) {

  #Método K-fold
  for(i in 0:(k-1)) {

    idx.test <- idx.aleatorios[(i*size+1):((i+1)*size)]
    idx.test <- idx.test[!is.na(idx.test)]

    idx.train <- idx.aleatorios[-idx.test]

    #Entrenamos nuestra red con los elementos que no apartenecen a la muestra de test
    network <- nnet(y~x, data=dataframe, size=j, linout=T, subset=idx.train, maxit=200, trace=F)

    #Repetimos la creación de la red si $value es demasiado grande
    while(network$value>500) {
      network <- nnet(y~x, data=dataframe, size=j, linout=T, subset=idx.train, maxit=200, trace=F)
    }
    estimatedValues[idx.test, j] <- predict(network, data.frame(x=x[idx.test]))

  }

  plot(x, y, col="black", xlab = "Numeros aleatorios", ylab="Valor del logaritmo",
       main=paste("Red de", j, "neurona(s)"), pch=".", cex=5)

  points(x, as.numeric(estimatedValues[,j]), col="red", pch=".", cex=3)
  legend(4000, 4, legend=c("A mano","Con la red"), col=c("black","red"), pch=c(".",".",pt.cex=c(5,5),

  #Pintamos la red gracias al código nnet_plot_fun.r
  plot(network)

  #Guardamos los valores de mse y de mae en un vector
  mseVector[j] <- mse(y, as.numeric(estimatedValues[,j]))
  maeVector[j] <- mae(y, as.numeric(estimatedValues[,j]))

}
```

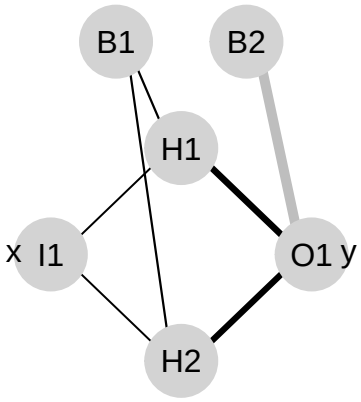
```
## Loading required package: scales
```

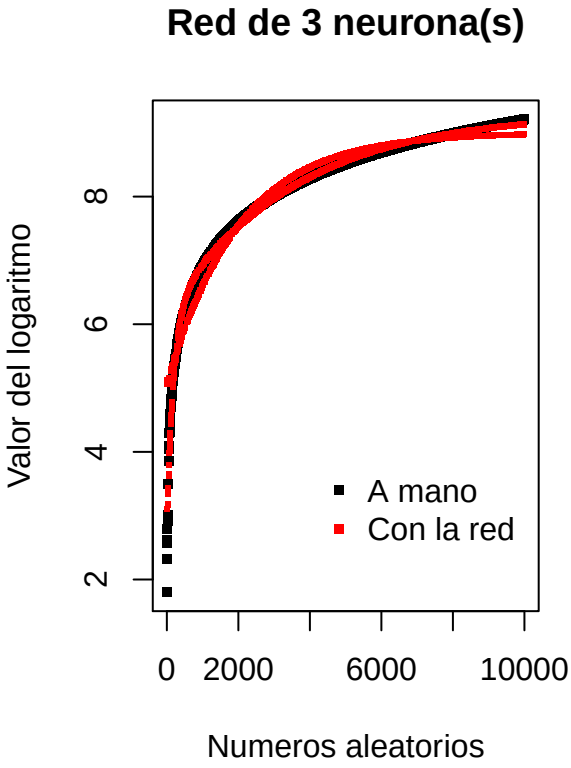
```
## Warning: package 'scales' was built under R version 3.3.2
```

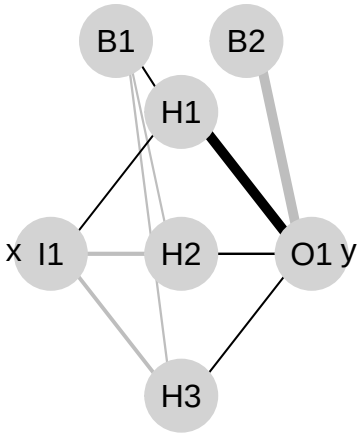

Red de 1 neurona(s)
Valor del logaritmo
2
4
6
8
0
2000
6000
10000
Numeros aleatorios
A mano
Con la red

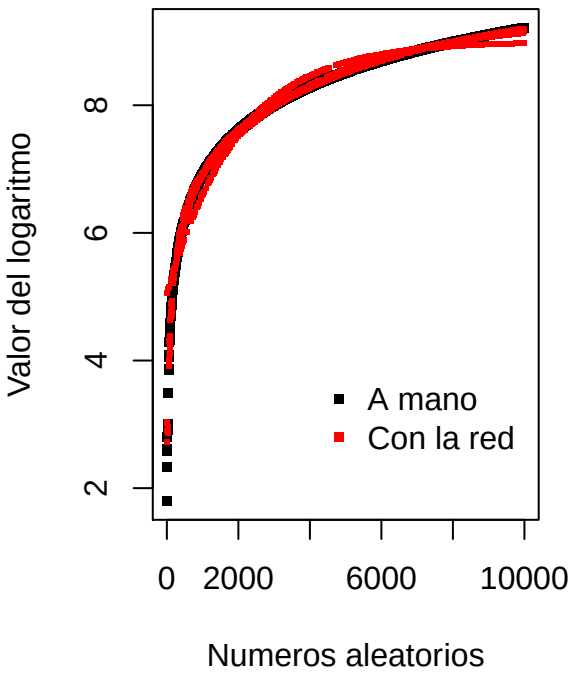

B1
B2
x
I1
H1
O1
y

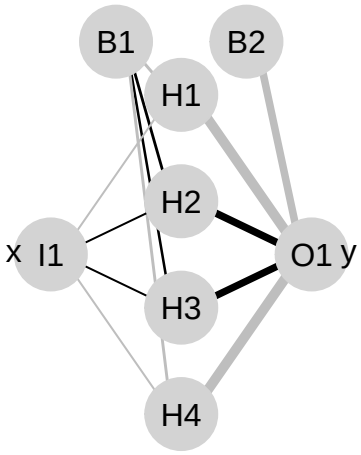

Red de 2 neurona(s)
Valor del logaritmo
2
4
6
8
0
2000
6000
10000
Numeros aleatorios
A mano
Con la red

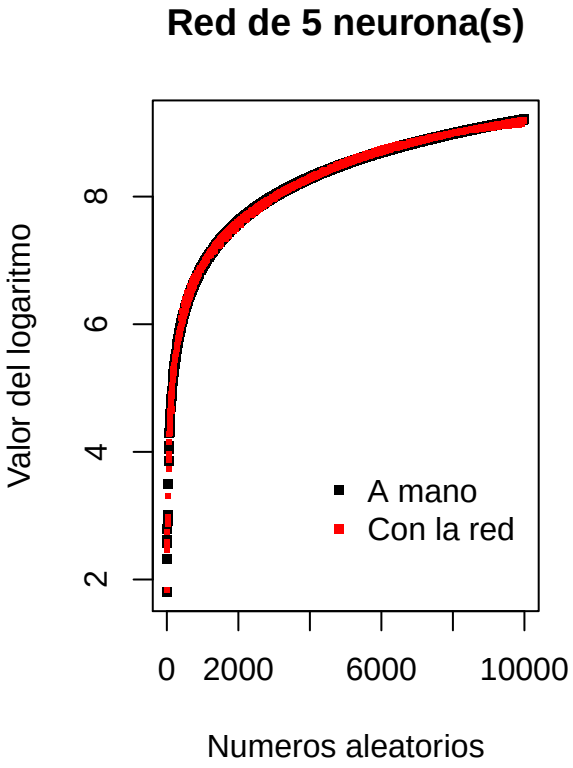

B1
B2
H1
x
I1
O1
y
H2

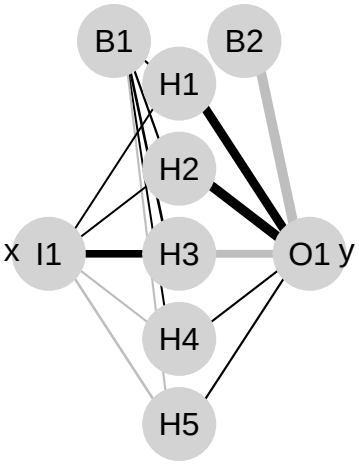

Red de 3 neurona(s)
Valor del logaritmo
2
4
6
8
0
2000
6000
10000
Numeros aleatorios
A mano
Con la red

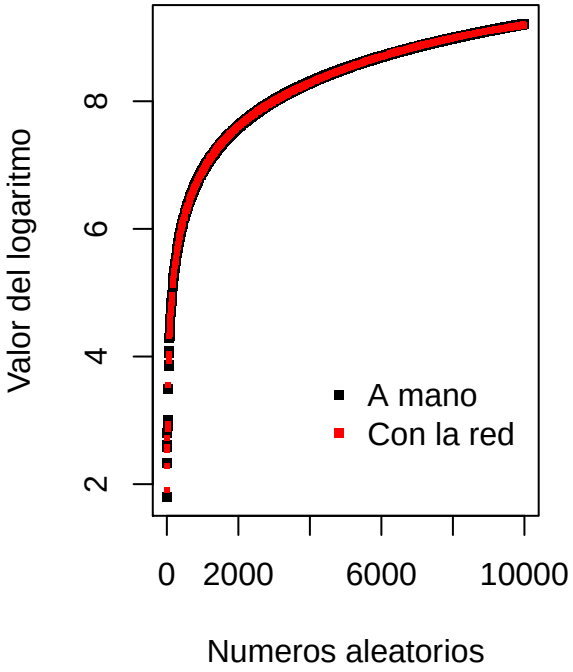

B1
B2
H1
x
I1
H2
O1
y
H3

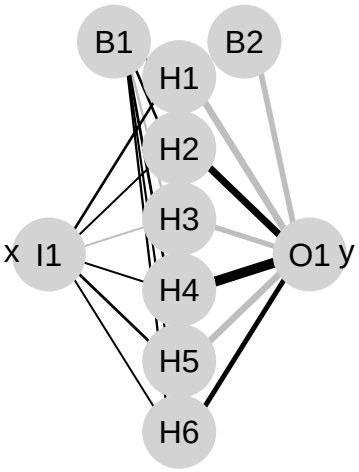

Red de 4 neurona(s)
Valor del logaritmo
2
4
6
8
0
2000
6000
10000
Numeros aleatorios
A mano
Con la red

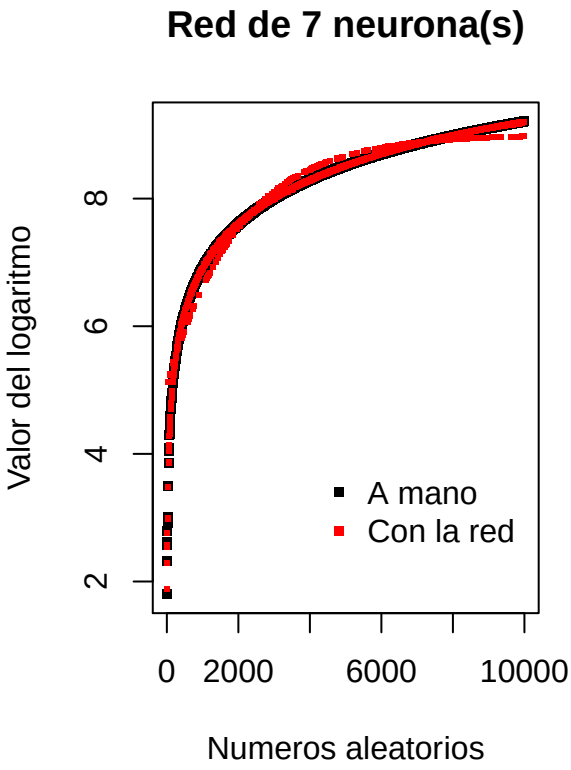

B1
B2
H1
H2
x
I1
O1
y
H3
H4

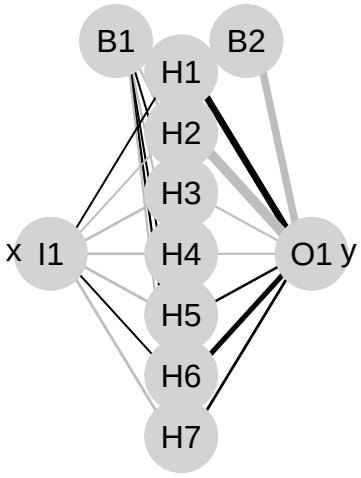

Red de 5 neurona(s)
Valor del logaritmo
2
4
6
8
0
2000
6000
10000
Numeros aleatorios
A mano
Con la red

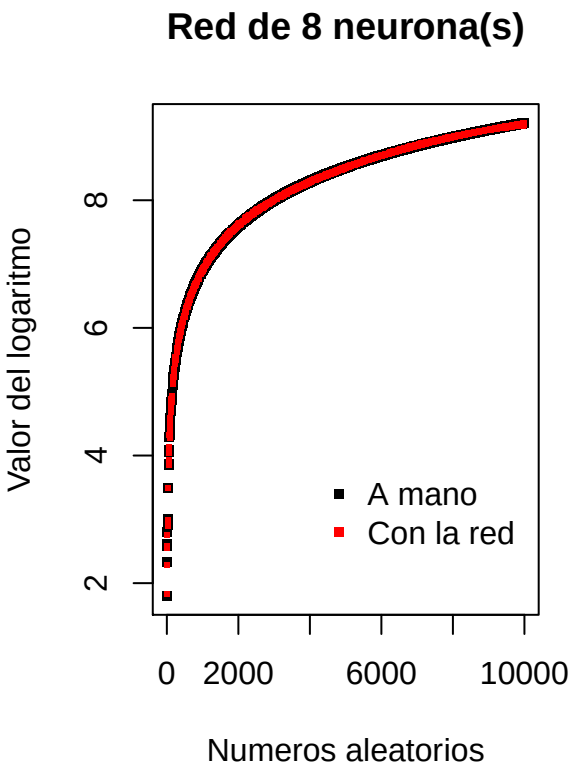

B1
B2
H1
H2
x
I1
H3
O1
y
H4
H5

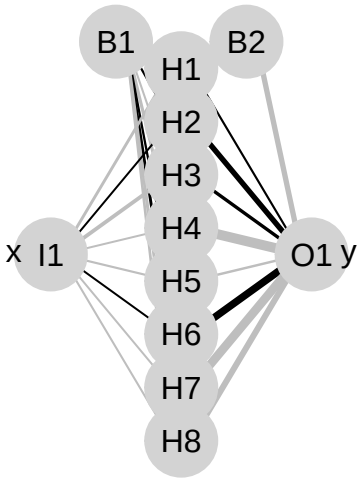

Red de 6 neurona(s)
Valor del logaritmo
2
4
6
8
0
2000
6000
10000
Numeros aleatorios
A mano
Con la red


B1
B2
H1
H2
H3
x
I1
O1
y
H4
H5
H6



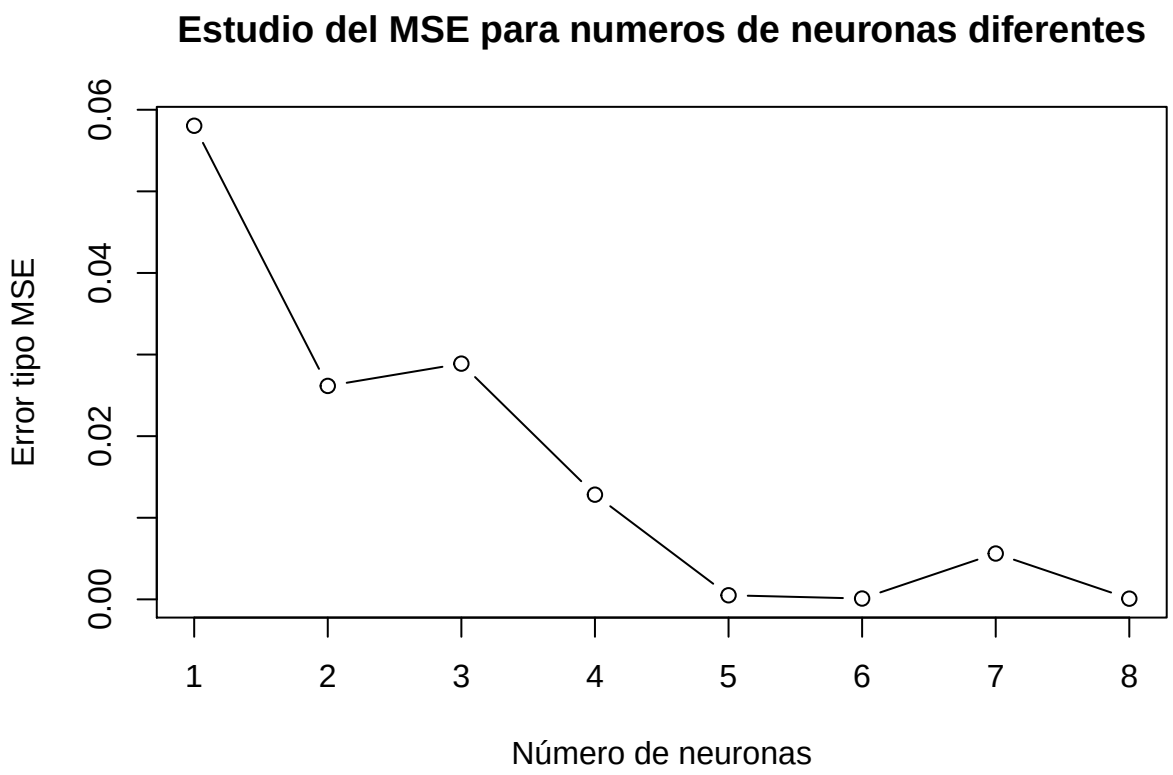






En los bucles anteriores, guardamos en un vector mseVector y maeVector los valores de los errores tipo *mse* y de *mae* obtenidos con diferentes números de neurones. Podemos ahora pintar estos valores para compararlos. Lo que buscamos es una manera de determinar la mejor red neuronal, es decir, la red que tiene el valor del error *mse* o de *mae* más pequeño.

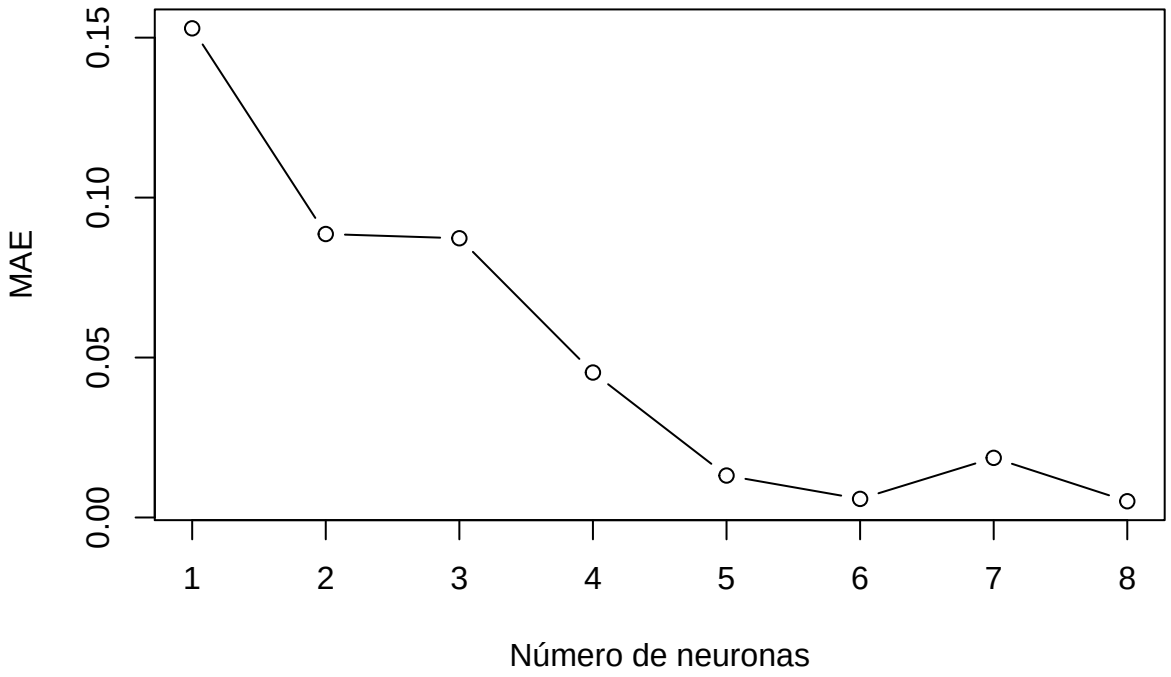
```
par(mfrow=c(1,1))
plot(1:numberNeurons, mseVector, type="b",
     main="Estudio del MSE para numeros de neuronas diferentes",
     xlab="Número de neuronas", ylab="Error tipo MSE")
```



Con este último plot, se ve claramente que la red que tiene una sola neurona es la que devuelve el peor error de todas. Cuando aumentamos el número de neuronas de la red, vemos que hay una tendencia a tener un error menor (como lo esperamos). Con la semilla que estamos usando, se observa también que para 5, 6 y 8 neuronas y un número máximo de iteraciones de 200, el error obtenido es casi igual a 0 (se ve también claramente en el plot de comparación entre datos calculados a mano y obtenidos con la red que obtenemos algo muy interesante a este nivel, ya no hay casi puntos rojos que se alejan mucho de la curva negra). Observamos también que el punto a 7 neuronas tiene un error un poco más grande que el error obtenido con la red de 6 neurones, pero es solamente una fluctuación estadística. Se puede comprobar cambiando la semillla al principio (si la ponemos por ejemplo a 3, observamos que la mejor red se obtiene con 7 neuronas).

Podemos ahora comparar estos resultados con los devueltos por la función *mae*.

```
plot(1:numberNeurons, maeVector, type="b",
     main="Estudio del MAE para numeros de neuronas diferentes",
     xlab="Número de neuronas", ylab="MAE")
```



Vemos que el plot obtenido usando la función de error *mae* es muy parecido al anterior (por lo menos, tiene la misma forma aunque por supuesto, los valores obtenidos no son los mismos porque se calculan de manera distinta).

En conclusión, este ejercicio nos permitió crear por primera vez una red neuronal, que tenía un objetivo muy sencillo : poder calcular el valor del logaritmo de números entre 0 y 10000. Para esto, intentamos crear diferentes redes neuronales, compuestos de un número distinto de neuronas cada vez (entre 1 y 8) y usando un método de validación cruzada de tipo k-fold, con k = 10. Para cada red, pintamos los valores esperados del log y el valor devuelto por la red para compararlos, y al final de todo, pintamos un plot que representa el valor del error cometido en función del número de neuronas. Lo que observamos en este plot es que el error baja bastante cuando el número de neuronas aumenta, como lo esperamos. Con la semilla elegida y los datos que tenemos, podemos así concluir que la mejor red ha sido la red obtenida con 5 neuronas en la capa oculta.

Bibliografia

KOHABI, R. 1995. *A study of cross-validation and boostrap for accuracy estimation and model selection.* International Joint Conference on Artifical Intelligence.

ETHZ. Sin fecha. *Concatenate Strings* [sitio web]. [Consulta: 14 diciembre 2016]. Disponible en: https://stat.ethz.ch/R-manual/R-devel/library/base/html/paste.html

ETHZ. 2015. *Pch size in a legend* [sitio web]. [Consulta: 19 diciembre 2016]. Disponible en: https://stat.ethz.ch/pipermail/r-help/2015-February/425395.html