DEEP LEARNING EN RSTUDIO

REDES NEURONALES

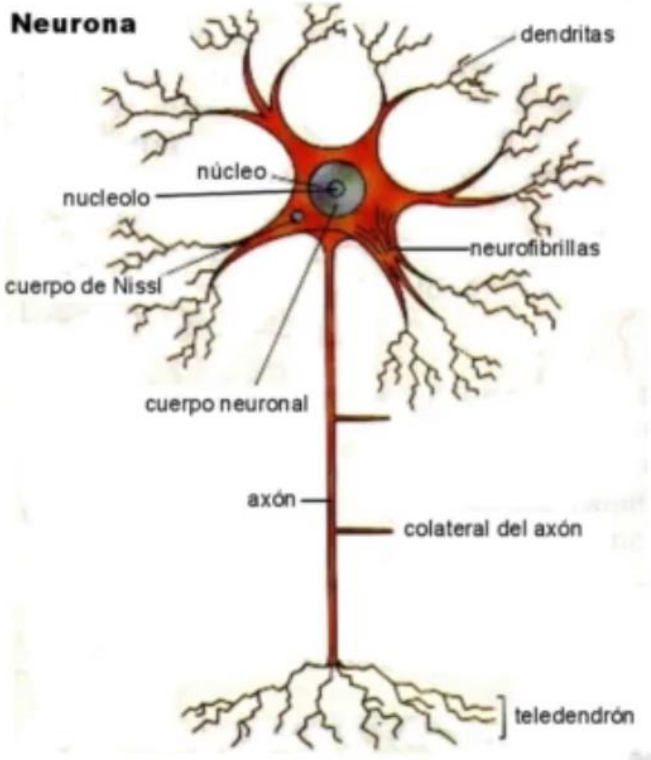
EDUARD LARA

1. INDICE

1. INTRODUCCION A LA REDES NEURONALES

Neurona

- Una neurona es una célula y es el componente principal del sistema nervioso, cuya función es recibir, procesar y transmitir información a través de señales químicas y eléctricas.
- Diagrama de una neurona biológica con 3 partes principales:
- Soma: Cuerpo celular o núcleo
- Dendritas: Prolongaciones cortas que reciben información y la transmiten al Soma
- Axón: Prolongación corta que conduce los impulsos hacia otra neurona



1. INTRODUCCION A LA REDES NEURONALES

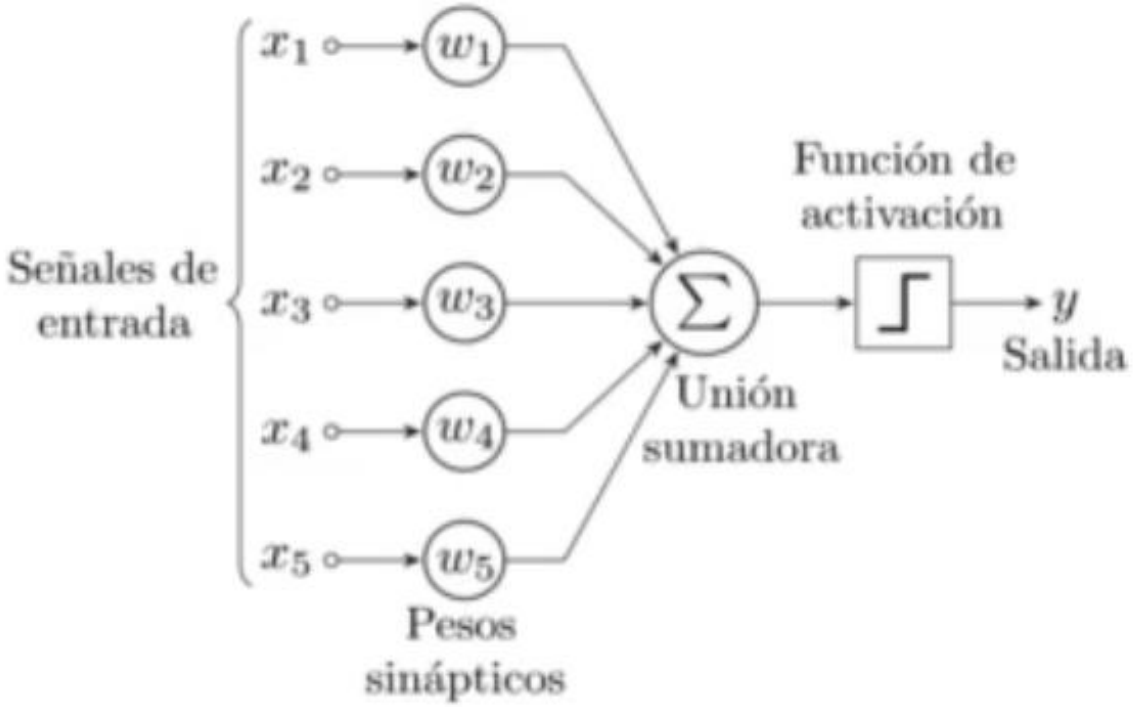
Perceptrón

- Un perceptrón es una neurona artificial
- Se une con otros perceptrones para crear una red neuronal artificial.
- Cada perceptrón tiene:
  - Canales de entrada (X1, X2, Xn) (Dentritas)
  - Una función de activación (Soma o nucleo)
  - Un canal de salida (y) (Axon)

1. INTRODUCCION A LA REDES NEURONALES

- Su representación matemática es

$$\sum_{i=0}^{n} w_i x_i + b$$



2. EJEMPLO RED NEURONAL

Paso 1. Vamos a ver ahora las redes neuronales, la parte que se corresponde con el deep learning o aprendizaje profundo. Instalamos el paquete MASS y lo cargamos en memoria

```
> install.packages('MASS')
Installing package into 'C:/Users/eduar/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/MASS_7.3-60.0.1.zip'
Content type 'application/zip' length 1174208 bytes (1.1 MB)
downloaded 1.1 MB

package 'MASS' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\eduar\AppData\Local\Temp\RtmpmKsdJZ\downloaded_packages
> library(MASS)
>
```

2. EJEMPLO RED NEURONAL

Paso 2. Recuperamos los datos de un dataset llamado Boston, que contienen las características de las casas de Boston de 1978. Contiene 506 filas/observaciones y 14 variables con las características de los inmuebles: % de crimen, numero de Habitaciones, la edad, etc

```
> datos = Boston
> str(datos)
'data.frame':   506 obs. of  14 variables:
 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
 $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
 $ rm     : num  6.58 6.42 7.18 7 7.15 ...
 $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
 $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
 $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
 $ black  : num  397 397 393 395 397 ...
 $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
 $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
>
```

La ultima columna es la que queremos predecir: valor medio de la vivienda en miles de dólares.

Data	
datos	506 obs. of 14 variables

2. EJEMPLO RED NEURONAL

Paso 3. Con head() vemos las primeras 6 líneas de datos

```
> head(datos)
     crim zn indus chas   nox    rm  age    dis rad tax
1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296
2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242
3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242
4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222
5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222
6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222
  ptratio  black lstat medv
1    15.3 396.90  4.98 24.0
2    17.8 396.90  9.14 21.6
3    17.8 392.83  4.03 34.7
4    18.7 394.63  2.94 33.4
5    18.7 396.90  5.33 36.2
6    18.7 394.12  5.21 28.7
>
```

2. EJEMPLO RED NEURONAL

Paso 4. Revisamos si hay algún valor no nulo dentro del dataset, poder ponerle algún valor.

```
> any(is.na(datos))
[1] FALSE
>
```

Con any(is.na()) verificamos si hay algún valor nulo dentro de Boston. No hay ningún valor nulo, por lo cual podemos seguir.

2. EJEMPLO RED NEURONAL

Paso 5. Lo siguiente que haremos será normalizar los datos del dataset para ver las relaciones entre diferentes columnas. Calcularemos un máximo y un mínimo para cada una de las columnas para hacer esa normalización.

```
> maximo = apply(datos, 2, max)
> maximo
    crim       zn    indus     chas      nox       rm      age
 88.9762 100.0000  27.7400   1.0000   0.8710   8.7800 100.0000
     dis      rad      tax  ptratio    black    lstat     medv
 12.1265  24.0000 711.0000  22.0000 396.9000  37.9700  50.0000
> |
```

El 2 y max indica que calculará los valores máximos para cada una de las columnas, dentro de todas las filas que tiene este dataset que tiene 506

2. EJEMPLO RED NEURONAL

Paso 6. Haremos lo mismo para los valores mínimos.

```
> minimo = apply(datos, 2, min)
> minimo
     crim        zn     indus      chas       nox        rm
  0.00632   0.00000   0.46000   0.00000   0.38500   3.56100
      age       dis       rad       tax   ptratio     black
  2.90000   1.12960   1.00000 187.00000  12.60000   0.32000
    lstat      medv
  1.73000   5.00000
>
```

Aplica la función mínimo a las columnas y devuelve en este caso el mínimo por cada una de las columnas.

Estos dos valores máximos y mínimos lo utilizaremos para normalizar

2. EJEMPLO RED NEURONAL

Paso 7. Calcularemos los datos normalizados mediante las funciones scale:

```
> datos.normalizados = scale(datos, center=minimo, scale=maximo-minimo)
>
```

Paso 8. Convertimos los datos normalizados en un dataframe. Los mostramos:

```
> datos.normalizados = as.data.frame(datos.normalizados)
>
```

```
> datos.normalizados
             crim    zn      indus chas       nox       rm       age
1   0.000000e+00 0.180 0.06781525    0 0.31481481 0.5775053 0.64160659
2   2.359225e-04 0.000 0.24230205    0 0.17283951 0.5479977 0.78269825
3   2.356977e-04 0.000 0.24230205    0 0.17283951 0.6943859 0.59938208
4   2.927957e-04 0.000 0.06304985    0 0.15020576 0.6585553 0.44181256
5   7.050701e-04 0.000 0.06304985    0 0.15020576 0.6871048 0.52832132
6   2.644715e-04 0.000 0.06304985    0 0.15020576 0.5497222 0.57466529
7   9.213230e-04 0.125 0.27162757    0 0.28600823 0.4696302 0.65602472
8   1.553672e-03 0.125 0.27162757    0 0.28600823 0.5002874 0.95983522
9   2.303251e-03 0.125 0.27162757    0 0.28600823 0.3966277 1.00000000
10 1.840173e-03 0.125 0.27162757    0 0.28600823 0.4680973 0.85478888
11 2.456674e-03 0.125 0.27162757    0 0.28600823 0.5395670 0.94129763
12 1.249299e-03 0.125 0.27162757    0 0.28600823 0.4690554 0.82389289
13 9.830293e-04 0.125 0.27162757    0 0.28600823 0.4460625 0.37178167
```

2. EJEMPLO RED NEURONAL

Paso 9. Con un head() vemos mejor los datos normalizados, para ver las similitudes entre una característica u otra

```
> head(datos.normalizados)
          crim   zn      indus chas       nox       rm       age
1 0.0000000000 0.18 0.06781525    0 0.3148148 0.5775053 0.6416066
2 0.0002359225 0.00 0.24230205    0 0.1728395 0.5479977 0.7826982
3 0.0002356977 0.00 0.24230205    0 0.1728395 0.6943859 0.5993821
4 0.0002927957 0.00 0.06304985    0 0.1502058 0.6585553 0.4418126
5 0.0007050701 0.00 0.06304985    0 0.1502058 0.6871048 0.5283213
6 0.0002644715 0.00 0.06304985    0 0.1502058 0.5497222 0.5746653
        dis        rad        tax   ptratio     black      lstat
1 0.2692031 0.00000000 0.20801527 0.2872340 1.0000000 0.08967991
2 0.3489620 0.04347826 0.10496183 0.5531915 1.0000000 0.20447020
3 0.3489620 0.04347826 0.10496183 0.5531915 0.9897373 0.06346578
4 0.4485446 0.08695652 0.06679389 0.6489362 0.9942761 0.03338852
5 0.4485446 0.08695652 0.06679389 0.6489362 1.0000000 0.09933775
6 0.4485446 0.08695652 0.06679389 0.6489362 0.9929901 0.09602649
       medv
1 0.4222222
2 0.3688889
3 0.6600000
4 0.6311111
5 0.6933333
6 0.5266667
>
```

2. EJEMPLO RED NEURONAL

Paso 10. Dividiremos estos datos normalizados en datos de entrenamiento y datos de pruebas. Para ello utilizaremos la librería CaTools

```
> library(caTools)
> 
```

Paso 11. Creamos una variable división mediante samples.split. Le pasaremos la columna objetivo (el valor medio de la vivienda) de los datos normalizados y luego con ratio Split de 0.7, estamos indicando que utilizaremos el 70% para entrenamiento y el 30% para pruebas

```
> division = sample.split(datos.normalizados$medv, SplitRatio = 0.7)
> entrenamientos = subset(datos.normalizados, division==TRUE)
> pruebas = subset(datos.normalizados, division==FALSE)
> 
```

2. EJEMPLO RED NEURONAL

Paso 12. Si visualizamos algunos elementos de entrenamiento y de pruebas con head(), vemos que corresponden a filas distintas, con lo cual ya tenemos divididos los datos normalizados en entrenamiento y pruebas

```
> head(entrenamientos)
          crim    zn      indus chas       nox       rm       age       dis
2 0.0002359225 0.000 0.24230205    0 0.1728395 0.5479977 0.7826982 0.3489620
3 0.0002356977 0.000 0.24230205    0 0.1728395 0.6943859 0.5993821 0.3489620
5 0.0007050701 0.000 0.06304985    0 0.1502058 0.6871048 0.5283213 0.4485446
6 0.0002644715 0.000 0.06304985    0 0.1502058 0.5497222 0.5746653 0.4485446
7 0.0009213230 0.125 0.27162757    0 0.2860082 0.4696302 0.6560247 0.4029226
9 0.0023032514 0.125 0.27162757    0 0.2860082 0.3966277 1.0000000 0.4503542
         rad       tax   ptratio
2 0.04347826 0.10496183 0.5531915 1.00
3 0.04347826 0.10496183 0.5531915 0.98
5 0.08695652 0.06679389 0.6489362 1.00
6 0.08695652 0.06679389 0.6489362 0.99
7 0.17391304 0.23664122 0.2765957 0.99
9 0.17391304 0.23664122 0.2765957 0.97
>
```

```
> head(pruebas)
           crim    zn      indus chas       nox       rm       age
1  0.0000000000 0.180 0.06781525    0 0.3148148 0.5775053 0.6416066
4  0.0002927957 0.000 0.06304985    0 0.1502058 0.6585553 0.4418126
8  0.0015536719 0.125 0.27162757    0 0.2860082 0.5002874 0.9598352
10 0.0018401733 0.125 0.27162757    0 0.2860082 0.4680973 0.8547889
11 0.0024566741 0.125 0.27162757    0 0.2860082 0.5395670 0.9412976
13 0.0009830293 0.125 0.27162757    0 0.2860082 0.4460625 0.3717817
         dis        rad        tax   ptratio     black      lstat      medv
1  0.2692031 0.00000000 0.20801527 0.2872340 1.0000000 0.08967991 0.4222222
4  0.4485446 0.08695652 0.06679389 0.6489362 0.9942761 0.03338852 0.6311111
8  0.4383872 0.17391304 0.23664122 0.2765957 1.0000000 0.48068433 0.4911111
10 0.4967309 0.17391304 0.23664122 0.2765957 0.9743053 0.42411700 0.3088889
11 0.4744155 0.17391304 0.23664122 0.2765957 0.9889556 0.51655629 0.2222222
13 0.3929562 0.17391304 0.23664122 0.2765957 0.9838620 0.38576159 0.3711111
>
```

3. EJEMPLO RED NEURONAL I

Paso 1. Instalamos el paquete que necesitamos de redes neuronales **neuralnet**

```
> install.packages('neuralnet')
Installing package into 'C:/Users/eduar/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
also installing the dependency 'Deriv'

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/Deriv_4.1.3.zip'
Content type 'application/zip' length 148848 bytes (145 KB)
downloaded 145 KB

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/neuralnet_1.44.2.zip'
Content type 'application/zip' length 123877 bytes (120 KB)
downloaded 120 KB

package 'Deriv' successfully unpacked and MD5 sums checked
package 'neuralnet' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\eduar\AppData\Local\Temp\RtmpQLDm9Y\downloaded_packages
> library(neuralnet)
>
```

3. EJEMPLO RED NEURONAL II

Paso 2. Creamos la fórmula para nuestro modelo

```
> formula = medv ~ crim + zn + indus + chas + nox + rm + age + dis + rad + tax + ptratio + black + lstat
>
```

La fórmula va a ser la columna objetivo, el precio medio de las viviendas, y luego la suma de las demás columnas de características que las usaremos para calcular el valor medio.

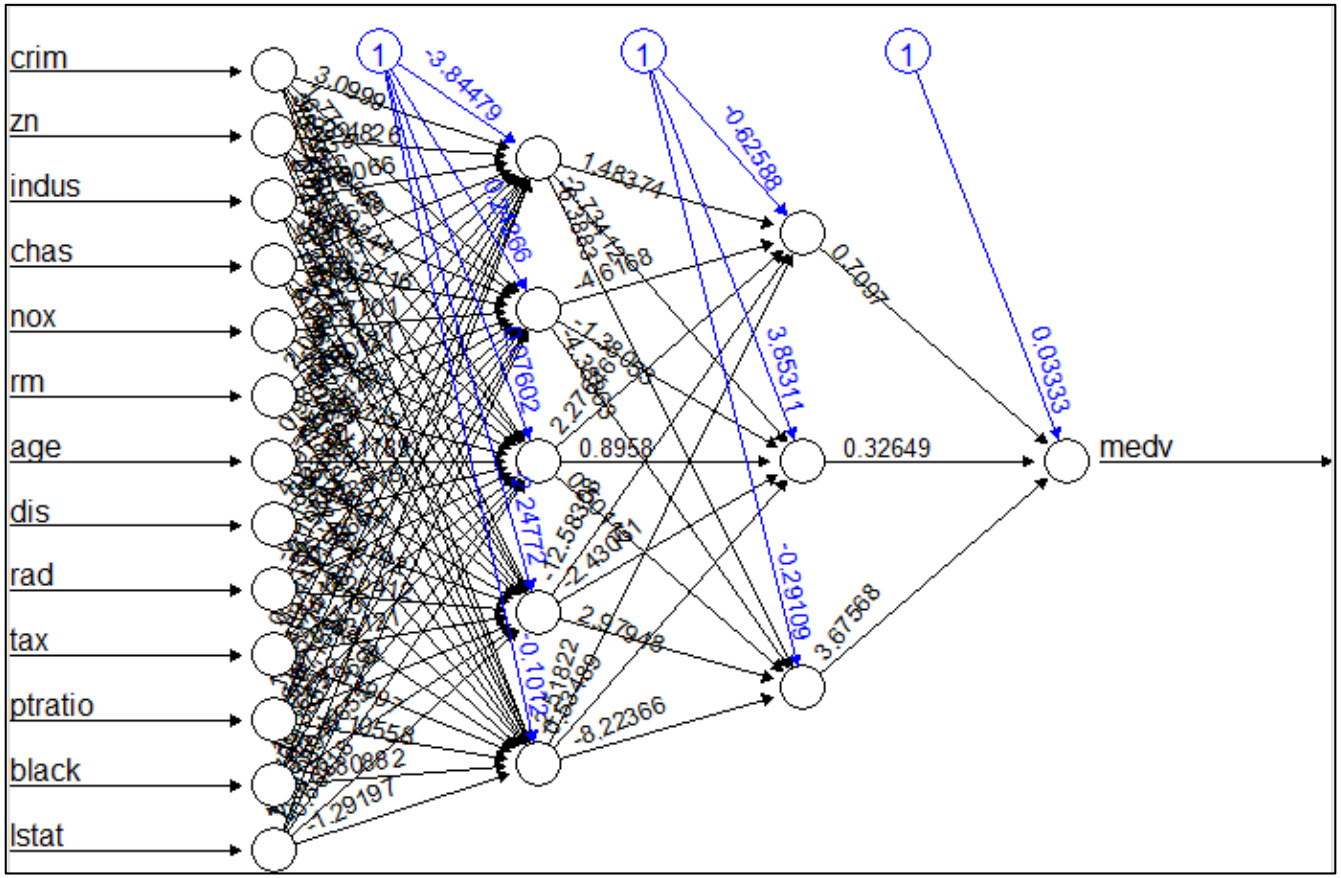
Paso 3. Ahora crearemos el modelo de red neuronal mediante la funcion neuralnet, y le pasamos como parámetros la formula, los datos de entrenamiento, las capas ocultas donde vamos a poner dos capas una de 5 neuronas y otra de 3. Y por último, el parámetro de línear output igual a TRUE

```
> red.neuronal = neuralnet(formula, data=entrenamientos, hidden=c(5,3), linear.output=TRUE)
>
```

3. EJEMPLO RED NEURONAL

Paso 4. Para ver la red neuronal en un gráfico utilizamos la función plot y le pasamos por parámetro red.neuronal

```
> plot(red.neuronal)
>
```



3. EJEMPLO RED NEURONAL II

Explicación

- Si hacemos un zoom lo veremos mejor.
- Vemos nuestra red neuronal, donde en la capa de entrada tenemos los distintos atributos del dataset, luego hay una capa intermedia de 5 neuronas ocultas, una segunda capa de 3 neuronas, y el resultado final sería conseguir una estimación del valor medio de la vivienda.
- También aparecen los elementos vías que se utilizan en la fórmula para calcular el valor final y cada uno de los enlaces tendrá un peso.
- Esto es lo conseguido mediante la formula de red neuronal

3. EJEMPLO RED NEURONAL II

Paso 5. Ahora vamos a calcular las predicciones. Creamos una variable llamada predicciones, donde mediante la función compute, le pasamos la red neuronal y las 13 columnas de los datos de pruebas, menos la columna objetivo

```
> predicciones = compute(red.neuronal, pruebas[1:13])
>
```

3. EJEMPLO RED NEURONAL II

Paso 6. Si hacemos un str() de la variable predicciones, vemos que una lista de elementos donde aparecen todas las predicciones que hace en función de los datos del dataset y esta red neuronal

```
> str(predicciones)
List of 2
 $ neurons   :List of 3
  ..$ : num [1:139, 1:14] 1 1 1 1 1 1 1 1 1 1 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:139] "1" "4" "8" "10" ...
  .. .. ..$ : chr [1:14] "" "crim" "zn" "indus" ...
  ..$ : num [1:139, 1:6] 1 1 1 1 1 1 1 1 1 1 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:139] "1" "4" "8" "10" ...
  .. .. ..$ : NULL
  ..$ : num [1:139, 1:4] 1 1 1 1 1 1 1 1 1 1 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:139] "1" "4" "8" "10" ...
  .. .. ..$ : NULL
 $ net.result: num [1:139, 1] 0.524 0.667 0.326 0.327 0.323 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:139] "1" "4" "8" "10" ...
  .. ..$ : NULL
>
```

3. EJEMPLO RED NEURONAL II

Paso 7. Lo veremos de otra forma. Calcularemos las predicciones correctas, pero desnormalizando los datos que anteriormente normalizamos. Eliminaremos esa normalización, aplicándoles la función inversa tanto a las predicciones como a los datos pruebas.

```
> predicciones.correctas = predicciones$net.result * (max(datos$medv) - min(datos$medv)) + min(datos$medv)
>
```

La variable predicciones.correctas, son los datos de predicciones desnormalizados.

Las predicciones anteriores las hemos multiplicado por la resta entre máximo y mínimo, que lo que hace es darle la vuelta a lo que hicimos antes

3. EJEMPLO RED NEURONAL II

Paso 8. Hacemos lo mismo para las pruebas

```
> pruebas.convertidas = (pruebas$medv) * (max(datos$medv) - min(datos$medv)) + min(datos$medv)
> |
```

Paso 9. Una vez que tenemos tanto las predicciones como las pruebas desnormalizados, calculamos el error que hemos producido al hacer la estimación.

El error sería la suma de las pruebas convertidas menos las predicciones correctas, elevado al cuadrado, y dividido entre el número de filas. El error es un 21% de esta red neuronal al intentar hacer la estimación de los precios medios.

```
> error = sum((pruebas.convertidas - predicciones.correctas)^2) /nrow(pruebas)
> error
[1] 21.78142
> |
```

3. EJEMPLO RED NEURONAL II

Paso 10. Vamos a crear un dataframe con la relación entre el valor real y el valor de predicción, para compararlos y ver cuánto nos hemos ido equivocando.

Creamos un dataframe y le pasamos las pruebas convertidas y las predicciones correctas

```
> errores = data.frame(pruebas.convertidas, predicciones.correctas)
> 
```

3. EJEMPLO RED NEURONAL II

Paso 11. Si visualizamos estos errores, podemos ver los valores de la fila 4, el valor real del conjunto de pruebas 33.4 de nuestra predicción con la red neuronal día 35.01.

```
> errores
    pruebas.convertidas predicciones.correctas
1                  24.0                28.558495
4                  33.4                35.010222
8                  27.1                19.662344
10                 18.9                19.731539
11                 15.0                19.542792
13                 21.7                21.099439
14                 20.4                19.483072
17                 23.1                21.000675
18                 17.5                17.228166
21                 13.6                13.783447
23                 15.2                16.167004
```

Tenemos un porcentaje de error en torno al 20%, pero vemos que realiza las predicciones en función de todas las demás características o columnas del dataset.

3. EJEMPLO RED NEURONAL II

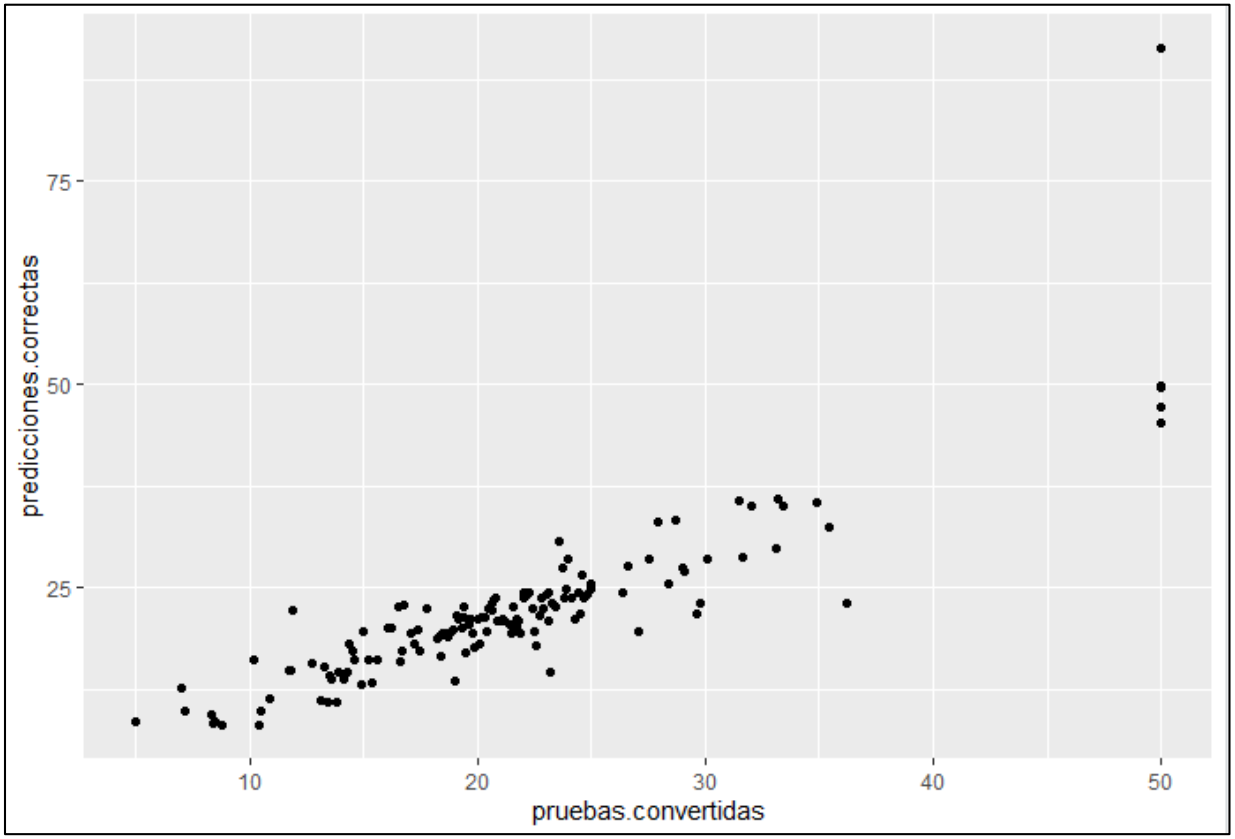
Paso 12. Podemos visualizar esto en un gráfico que se vea mejor. Cargamos la librería ggplot2, y creamos un gráfico al cual le pasamos el dataframe de los errores que acabamos de crear.

```
> library(ggplot2)
> grafico = ggplot(errores, aes(x=pruebas.convertidas, y=predicciones.correctas))
> grafico = grafico + geom_point()
> print(grafico)
>
```

En el eje x configuramos las pruebas convertidas (datos reales de prueba) y en el eje y las predicciones correctas. Indicamos que el grafico sea de tipo geom_point()

3. EJEMPLO RED NEURONAL II

Paso 13. Visualizamos el grafico y ampliamos con el zoom. Vemos la relación entre el valor real del conjunto de pruebas y las predicciones realizadas.



Cuanto más se acerca al centro, por ejemplo, el 20-20 sería el valor exacto, es decir, exactamente el valor. El valor de predicción coincide con el valor real.

3. EJEMPLO RED NEURONAL II

- Las aproximaciones son bastante más ajustadas, pero hay algunos casos que se va un poco, pero la mayor parte está bastante ajustadas entre el valor real y el valor de predicción.
- Si quisiéramos ajustar más, pues habría que aumentar el número de capas ocultas y el número de neuronas en cada una de las capas.