5. ANN

Predicción de la malignidad/benignidad de cáncer de mama

José Félix Rojas Cabeza

12 de May de 2020

Contents

Algoritmo Red Neuronal Artificial (ANN) **1**

Diagnóstico de cáncer de mama a partir de distintas variables biológicas **2**
1. Obtención de los datos: . . . 2
2. Exploración y preparación de datos . . . 2
3. Normalización de los datos: . . . 4
4. Preparación de los datos - Creando datasets de entrenamiento y de prueba. . . . 6
5. Entrenamiento del modelo en los datos . . . 7
6. Evaluación del desempeño del modelo . . . 11
7. Paquete `caret` . . . 13

Referencias **16**

Algoritmo Red Neuronal Artificial (ANN)

Las redes neuronales artificiales están inspiradas en las redes neuronales como las que tiene el cerebro. Las neuronas se sustituyen por nodos que reciben y envian señales (información). Se crea una red con diferentes nodos y capas interconectados para procesar la información. Cada capa esta formada por un grupo de nodos que transmite la información a los otros nodos de las capas siguientes.

Una red neuronal artificial se caracteriza por:

- La topología: Esto corresponde al número de capas y de nodos. Además de la dirección en que se la información pasa de un nodo al siguiente, dentro de capas o entre capas.
- La función de activación: Función que recibe un conjunto de entradas e integra la señales para transmitir la información a otro nodo.
- El algoritmo de entrenamiento: Establece la importancia de cada conexión para transmitir o no la señal a los nodos correspondientes. El más usado es el algoritmo "backpropagation". El nombre indica que para corregir los errores de predicción va hacia atras de la red corrigiendo los pesos de los nodos.

Las fortalezas y debilidades de este algoritmo son:

Table 1: fortalezas y debilidades de algoritmo ANN

Fortalezas	Debilidades
Se puede adaptar a problemas de clasificación o predicción numérica	Extremadamente intenso computacionalmente. Le
Capaz de modelar patrones más complejos que casi cualquier algoritmo	Muy propenso a sobreajustar los datos de entrena
Hace pocas suposiciones sobre las relaciones subyacentes de los datos	Resulta en un modelo de caja negra compleja que

Diagnóstico de cáncer de mama a partir de distintas variables biológicas

1. Obtención de los datos:

```
> data <- read.csv(params$data)
```

2. Exploración y preparación de datos

```
> dim(data)
```

```
[1] 683  10
```

```
> table(is.na(data))
```

```
FALSE
 6830
```

```
> str(data)
```

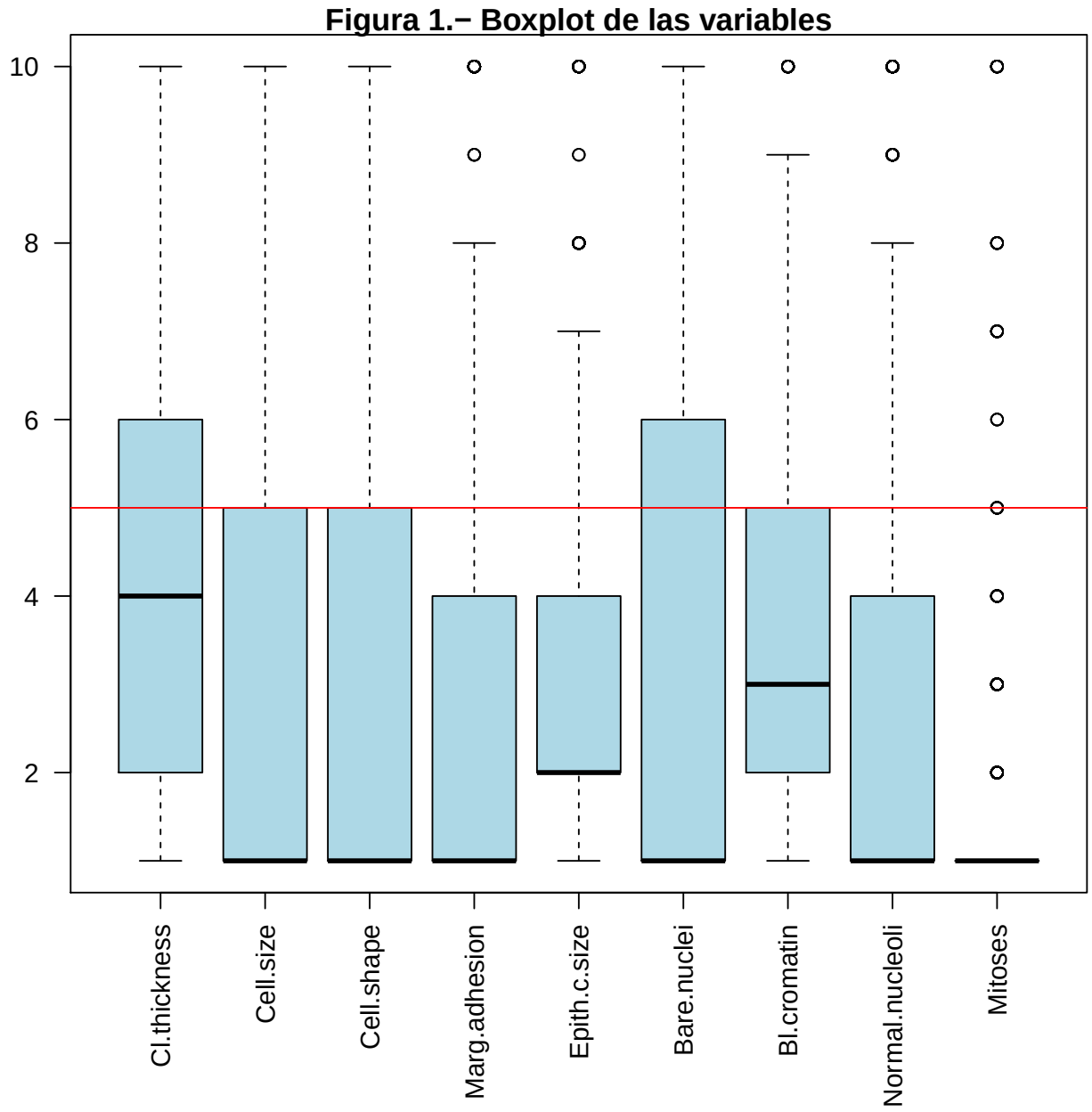
```
'data.frame':   683 obs. of  10 variables:
 $ Cl.thickness   : int  5 5 3 6 4 8 1 2 2 4 ...
 $ Cell.size      : int  1 4 1 8 1 10 1 1 1 2 ...
 $ Cell.shape     : int  1 4 1 8 1 10 1 2 1 1 ...
 $ Marg.adhesion  : int  1 5 1 1 3 8 1 1 1 1 ...
 $ Epith.c.size   : int  2 7 2 3 2 7 2 2 2 2 ...
 $ Bare.nuclei    : int  1 10 2 4 1 10 10 1 1 1 ...
 $ Bl.cromatin    : int  3 3 3 3 3 9 3 3 1 2 ...
 $ Normal.nucleoli: int  1 2 1 7 1 7 1 1 1 1 ...
 $ Mitoses        : int  1 1 1 1 1 1 1 1 5 1 ...
 $ Class          : Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 1 1 1 ...
```

```
> summary(data)
```

```
  Cl.thickness      Cell.size        Cell.shape     Marg.adhesion
 Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.00
 1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.: 1.00
 Median : 4.000   Median : 1.000   Median : 1.000   Median : 1.00
 Mean   : 4.442   Mean   : 3.151   Mean   : 3.215   Mean   : 2.83
 3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000   3rd Qu.: 4.00
 Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.00
  Epith.c.size     Bare.nuclei      Bl.cromatin     Normal.nucleoli
 Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.00
 1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.00
 Median : 2.000   Median : 1.000   Median : 3.000   Median : 1.00
 Mean   : 3.234   Mean   : 3.545   Mean   : 3.445   Mean   : 2.87
 3rd Qu.: 4.000   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 4.00
 Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.00
    Mitoses            Class
 Min.   : 1.000   benign   :444
 1st Qu.: 1.000   malignant:239
 Median : 1.000
 Mean   : 1.603
 3rd Qu.: 1.000
 Max.   :10.000
```

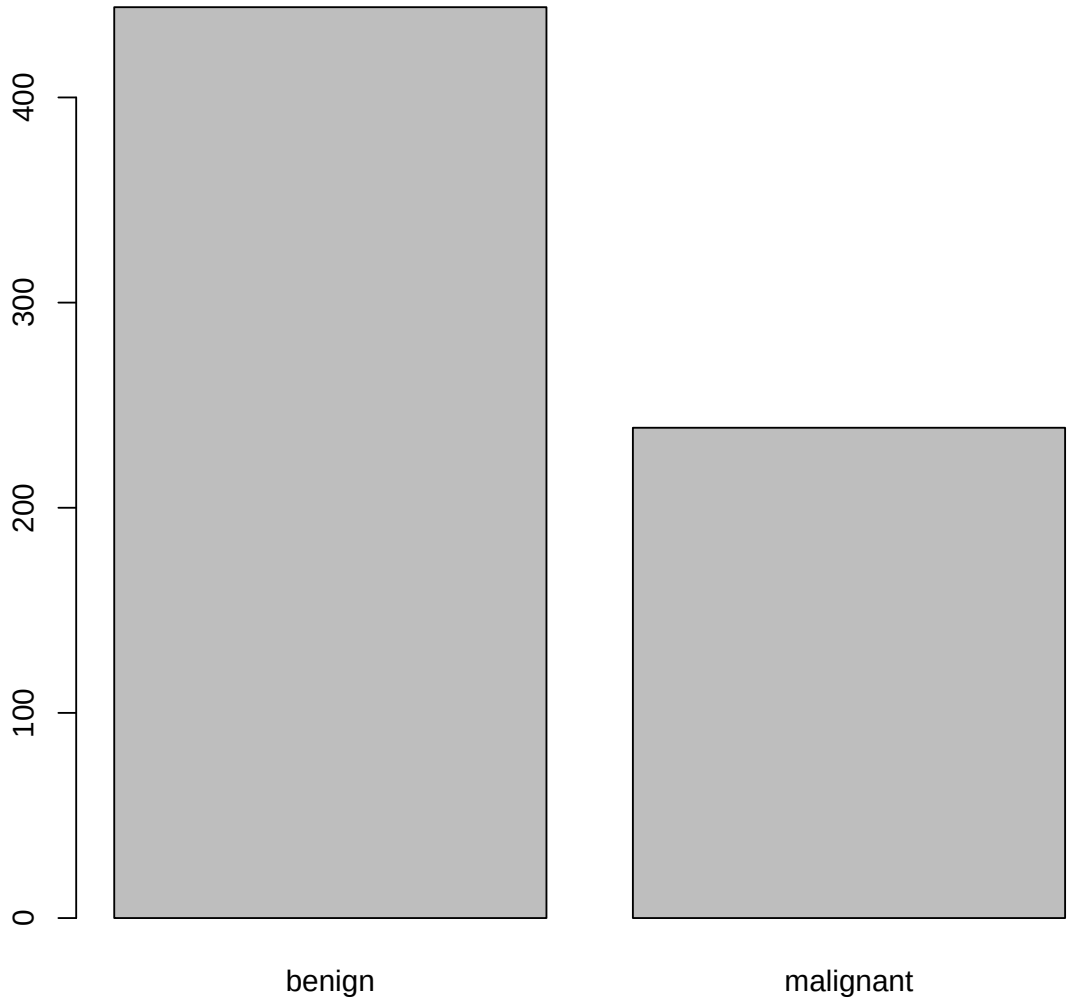
El archivo tiene 683 observaciones y 10 variables. Se observa que no hay datos faltantes, por lo que no hay que ajustar `data`. Los datos son numéricos, posiblmente enteros, excepto en el caso de la última columna (Clase), que tiene dos factores. A continuación se presenta la distribución de los datos en un boxplot (fig. 1 y 2):

```
> par(mar=c(7,2,1,1))
>
> boxplot(data[,1:9], las=2, col="lightblue", main="Figura 1.- Boxplot de las variables")
> abline(h=5, col="red")
```




```
> plot(data$Class, xlab = colnames(data$Class), main = "Figura 2.- gráfico de barras de Class")
```

Figura 2.– gráfico de barras de Class



3. Normalización de los datos:

Los datos numéricos deben normalizarse para que tomen valores entre 0 y 1. Se define la función `normalize()` de acuerdo con la literatura (Lantz 2013)

```
> normalize <- function(x) {
+ return ((x - min(x)) / (max(x) - min(x)))
+ }
>
> data_norm <- as.data.frame(lapply(data[,-10], normalize))
>
> # confirmación de rangos normalizados
> summary(data_norm)
```

```
 Cl.thickness      Cell.size        Cell.shape     Marg.adhesion
Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
1st Qu.:0.1111   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
Median :0.3333   Median :0.0000   Median :0.0000   Median :0.0000
Mean   :0.3825   Mean   :0.2390   Mean   :0.2461   Mean   :0.2034
3rd Qu.:0.5556   3rd Qu.:0.4444   3rd Qu.:0.4444   3rd Qu.:0.3333
Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
 Epith.c.size     Bare.nuclei      Bl.cromatin     Normal.nucleoli
Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
1st Qu.:0.1111   1st Qu.:0.0000   1st Qu.:0.1111   1st Qu.:0.0000
Median :0.1111   Median :0.0000   Median :0.2222   Median :0.0000
Mean   :0.2483   Mean   :0.2827   Mean   :0.2717   Mean   :0.2077
3rd Qu.:0.3333   3rd Qu.:0.5556   3rd Qu.:0.4444   3rd Qu.:0.3333
Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
   Mitoses
Min.   :0.00000
1st Qu.:0.00000
Median :0.00000
Mean   :0.06702
3rd Qu.:0.00000
Max.   :1.00000
```

Creación de variables:

Se crean tantas variables binarias como tenga la variable `Class`

```
> # Variables binarias que sustituirán a la variable factor:
> data_norm$M <- data$Class=="malignant"
> data_norm$B <- data$Class=="benign"
```

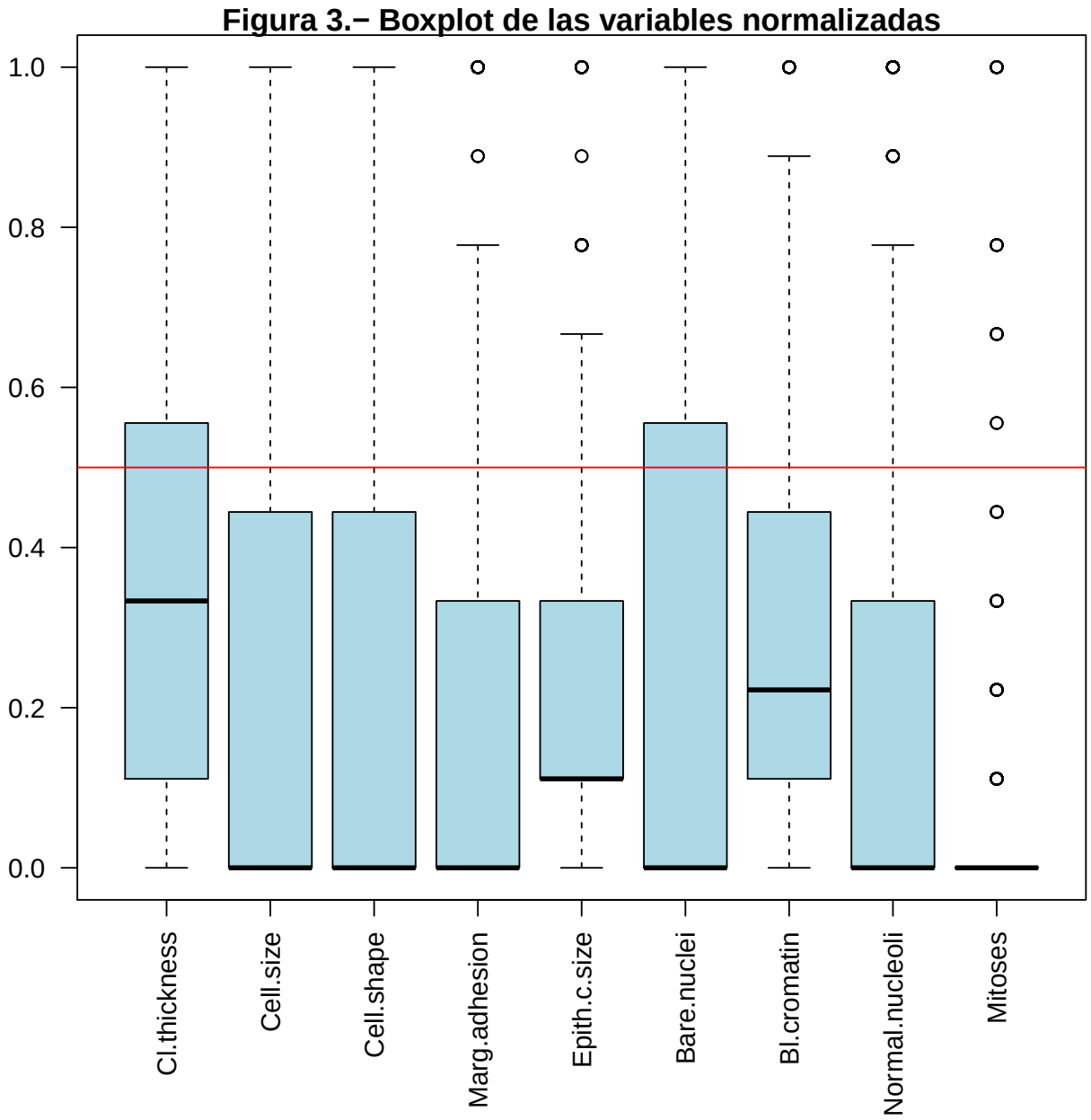
Resumen de `data_norm`:

```
> summary(data_norm)
```

```
 Cl.thickness      Cell.size        Cell.shape     Marg.adhesion
Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
1st Qu.:0.1111   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
Median :0.3333   Median :0.0000   Median :0.0000   Median :0.0000
Mean   :0.3825   Mean   :0.2390   Mean   :0.2461   Mean   :0.2034
3rd Qu.:0.5556   3rd Qu.:0.4444   3rd Qu.:0.4444   3rd Qu.:0.3333
Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
 Epith.c.size     Bare.nuclei      Bl.cromatin     Normal.nucleoli
Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
1st Qu.:0.1111   1st Qu.:0.0000   1st Qu.:0.1111   1st Qu.:0.0000
Median :0.1111   Median :0.0000   Median :0.2222   Median :0.0000
Mean   :0.2483   Mean   :0.2827   Mean   :0.2717   Mean   :0.2077
3rd Qu.:0.3333   3rd Qu.:0.5556   3rd Qu.:0.4444   3rd Qu.:0.3333
Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
   Mitoses            M               B
Min.   :0.00000   Mode :logical   Mode :logical
1st Qu.:0.00000   FALSE:444       FALSE:239
Median :0.00000   TRUE :239       TRUE :444
Mean   :0.06702
3rd Qu.:0.00000
Max.   :1.00000
```

Nótese que las variables numéricas no han cambiado su distribución, y que la normalización solamente ha escalado los valores (incluyendo la línea que marca el punto medio de los valores posibles, que se coloca en rojo para facilidad de interpretación de la información gráfica):

```
> par(mar=c(7,2.5,1,1))
>
> boxplot(data_norm[,1:9], las=2, col="lightblue", main="Figura 3.- Boxplot de las variables normalizada
> abline(h=0.5, col="red")
```



4. Preparación de los datos - Creando datasets de entrenamiento y de prueba.

Se crean los data sets de entrenamiento (train) y prueba (test):

```
> n <- nrow(data_norm)
> n_train <- params$ptrain
> set.seed(params$seed_train)
>
> train <- sample(n, floor(n*n_train))
>
> data_norm_train <- data_norm[train,]
> data_norm_test <- data_norm[-train,] # quita las columnas seleccionadas en train
```

Se toman 66.67% de las observaciones (455) para entrenar al modelo, y se guardan las otras (228) para evaluar el desempeño. Nótese que se tienen 11 variables porque se dividió `Class` en `B` y `M`

```
> dim(data_norm_train)
```

```
[1] 455  11
```

```
> dim(data_norm_test)
```

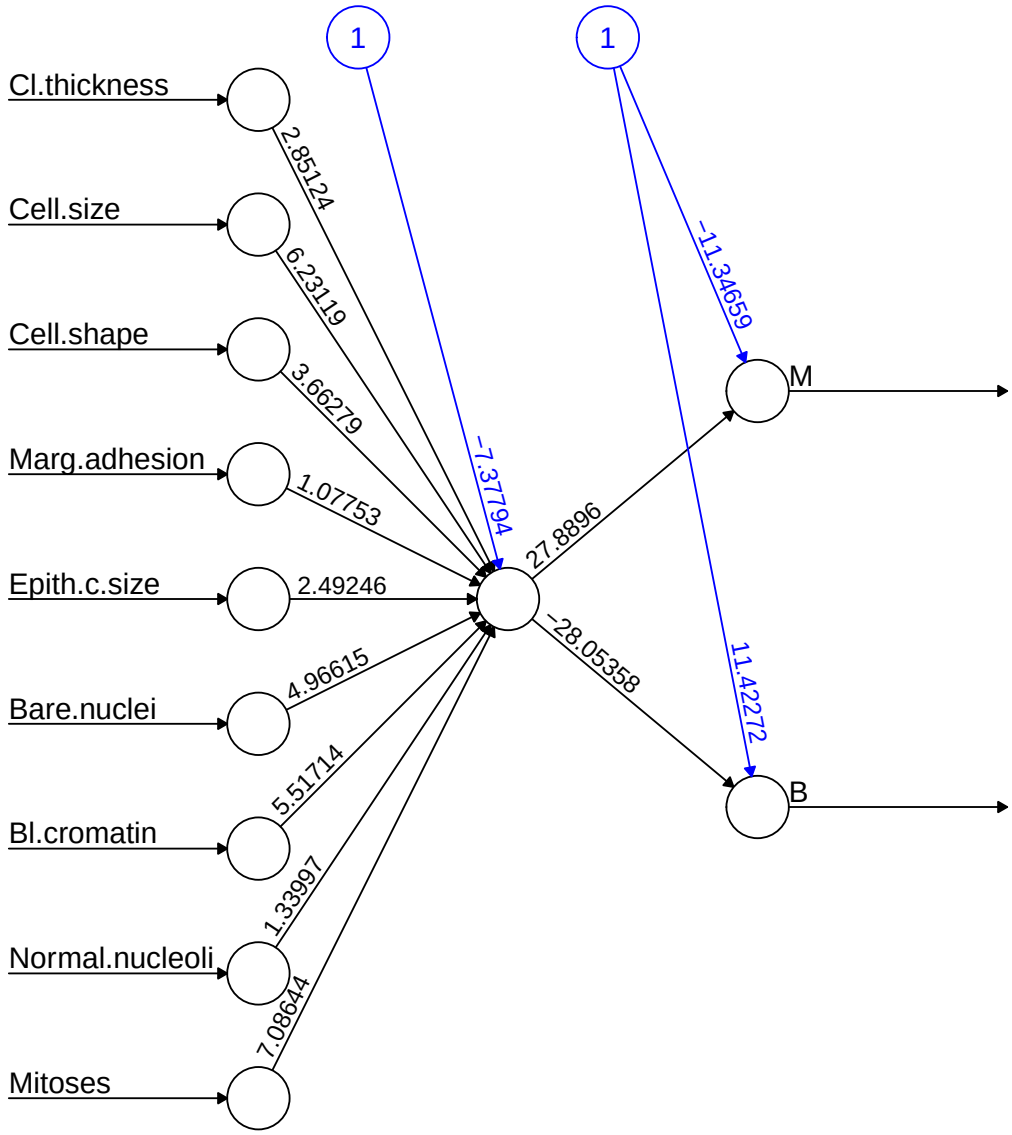
```
[1] 228  11
```

5. Entrenamiento del modelo en los datos

Para construir la red neuronal se utiliza la función `neuralnet()`, del paquete homónimo.

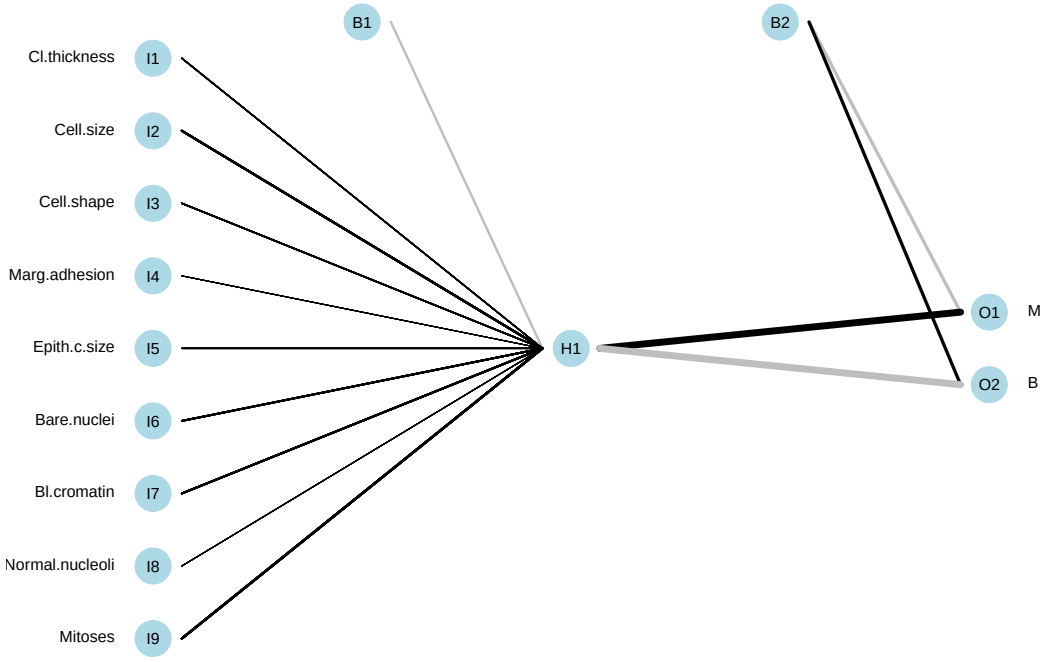
5.1.- Una neurona escondida

```
> if(!(require(neuralnet))) install.packages("neuralnet")
> if(!(require(NeuralNetTools))) install.packages("NeuralNetTools")
> set.seed(params$seed_neuralnet)
>
> library(neuralnet)
> library(NeuralNetTools)
>
> frmla = M+B ~ Cl.thickness+Cell.size+Cell.shape+Marg.adhesion+Epith.c.size+
+                 Bare.nuclei+Bl.cromatin+Normal.nucleoli+Mitoses
>
> # una sóla neurona escondida
> data_model_1 <- neuralnet(frmla, data=data_norm_train, hidden=1, linear.output = F)
>
>
> plot(data_model_1, rep = "best")
```



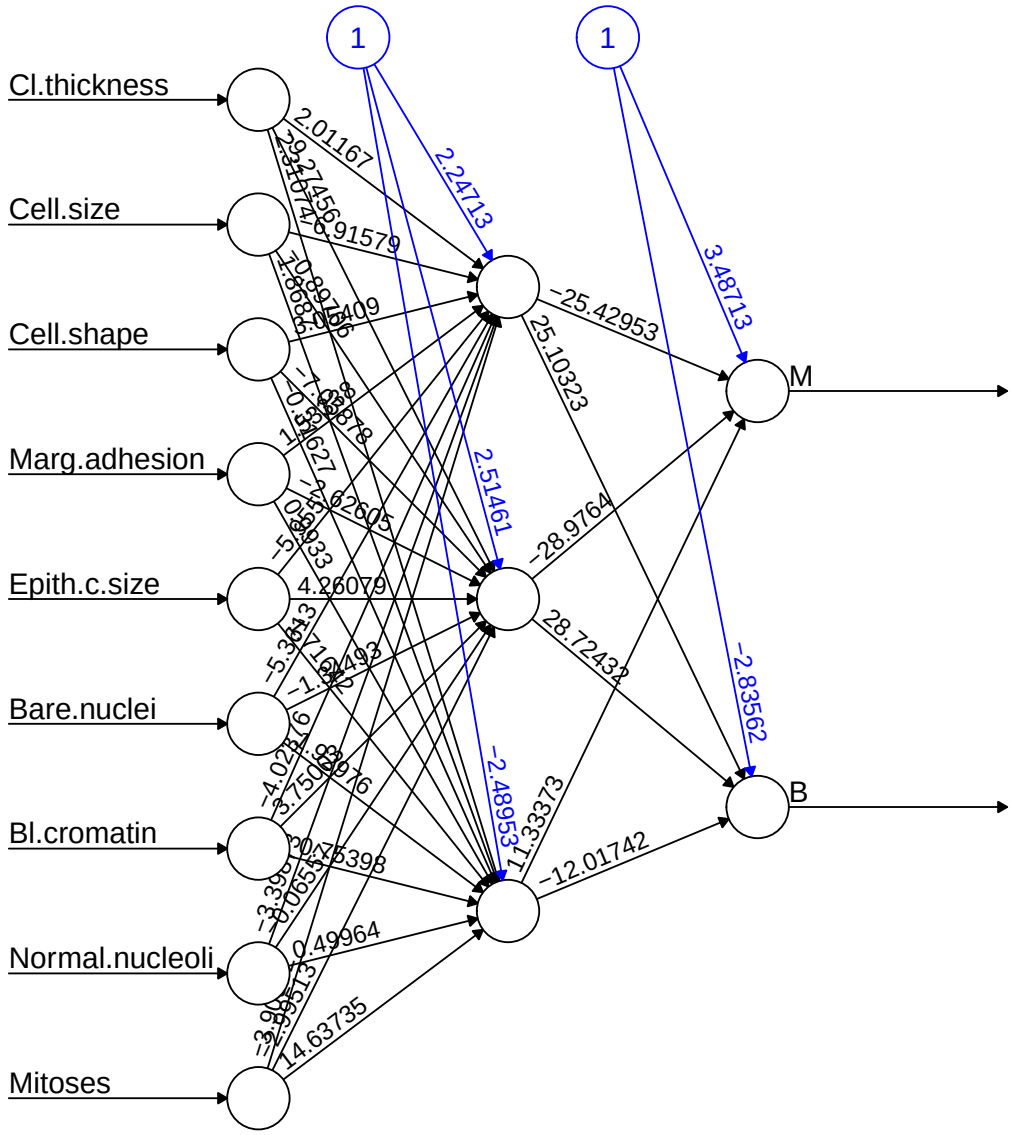
Error: 6.019217 Steps: 496

```
> # Hay que cambiar el tamaño de foto para que no corte los nombres!
> plotnet(data_model_1)
```



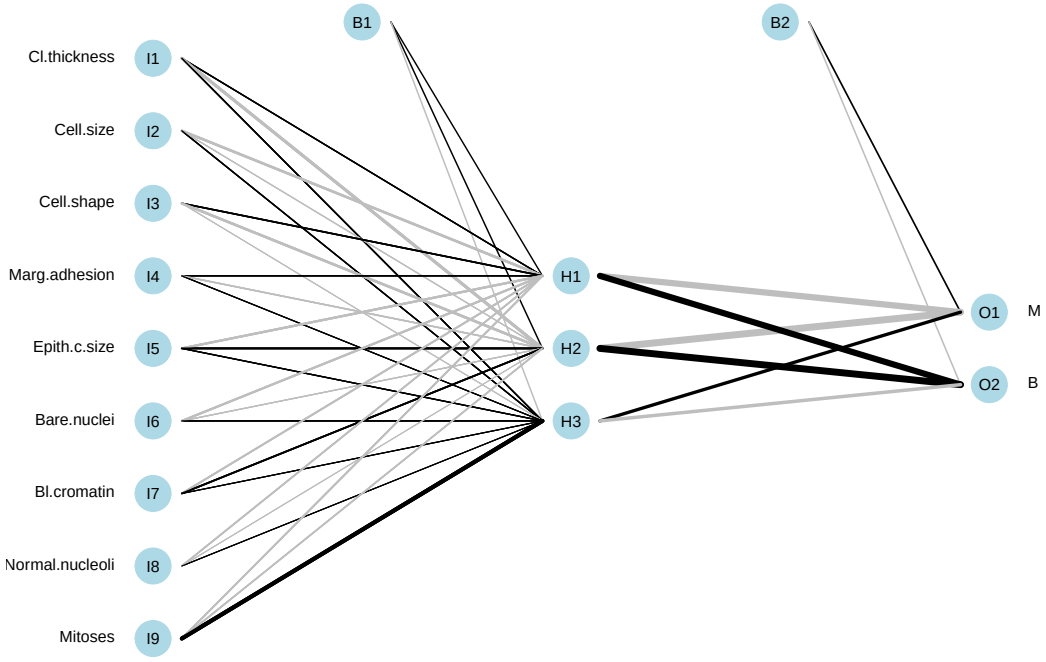
5.2.- Tres neuronas escondidas

```
> # tres neuronas escondidas
> data_model_3 <- neuralnet(frmla, data=data_norm_train, hidden=3, linear.output = F)
>
> plot(data_model_3, rep = "best")
```



Error: 3.039904 Steps: 351

```
> plotnet(data_model_3)
```



6. Evaluación del desempeño del modelo

Luego de obtenidos los modelos, evaluamos su rendimiento con `data_train_test` utilizando la función `predict.nn()`

6.1 Matriz de confusión del modelo con 1 nodo

```
> library(neuralnet)
> # debido a que se descontinuó compute() traté de utilizar predict.nn,
> # pero no pude hacerlo de esa manera
> model_result_1 <- compute(data_model_1, data_norm_test[, 1:9])
```

```
> library(neuralnet)
> model_result_1n.r <- model_result_1$net.result
>
> max_idx <- function(X) {
+   return(which(X == max(X) ) )
+ }
>
> # index
> idx_1 <- apply(model_result_1n.r, 1, max_idx)
> pred_1 <- c("malignant", "benign")[idx_1]
> res_1 <- table(pred_1, data$Class[-train])
```

```
> if(!(require(caret))) install.packages("caret")
> set.seed(params$seed_neuralnet)
> require(caret)
> c_mat1 <- confusionMatrix(res_1, positive="malignant")
> c_mat1
```

```
Confusion Matrix and Statistics


pred_1      benign malignant
  benign       146         1
  malignant      8        73

               Accuracy : 0.9605
                 95% CI : (0.9264, 0.9818)
    No Information Rate : 0.6754
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9121

 Mcnemar's Test P-Value : 0.0455

            Sensitivity : 0.9865
            Specificity : 0.9481
         Pos Pred Value : 0.9012
         Neg Pred Value : 0.9932
             Prevalence : 0.3246
         Detection Rate : 0.3202
   Detection Prevalence : 0.3553
      Balanced Accuracy : 0.9673

       'Positive' Class : malignant
```

El modelo con 1 nodo oculto obtiene una precisión de 1 96.0526316%, y una sensitividad y especificidad de 98.6486486% y 94.8051948% respectivamente.

6.2 Matriz de confusión del modelo con 3 nodos

```
> require(neuralnet)
> model_result_3 <- compute(data_model_3, data_norm_test[,1:9])
```

```
> require(neuralnet)
> model_result_3n.r <- model_result_3$net.result
>
> idx_3 <- apply(model_result_3n.r, 1, max_idx)
> pred_3 <- c("malignant", "benign")[idx_3]
> res_3 <- table(pred_3, data$Class[-train])
```

```
> require(caret)
> c_mat3 <- confusionMatrix(res_3, positive="malignant")
> c_mat3
```

```
Confusion Matrix and Statistics
```

```
pred_3      benign malignant
  benign       146         1
  malignant      8        73

               Accuracy : 0.9605
                 95% CI : (0.9264, 0.9818)
    No Information Rate : 0.6754
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9121

 Mcnemar's Test P-Value : 0.0455

            Sensitivity : 0.9865
            Specificity : 0.9481
         Pos Pred Value : 0.9012
         Neg Pred Value : 0.9932
             Prevalence : 0.3246
         Detection Rate : 0.3202
   Detection Prevalence : 0.3553
      Balanced Accuracy : 0.9673

       'Positive' Class : malignant
```

```
> c_mat1$byClass[1]
```

```
Sensitivity
  0.9864865
```

6.3 Comentarios finales

El modelo obtenido con un solo nodo obtiene mayor precisión. Si el desempeño de dos modelos es similar, es recomendable utilizar el modelo más sencillo (Lantz 2013). Al parecer, también hay una relación entre la complejidad del modelo y el sobreajuste.

7. Paquete `caret`

La función `nnet()` admite datos de tipo factor, por lo que no debemos transformar `Class` en variables binarias. También se puede utilizar `createDataPartition()` para crear los datos de entrenamiento y de prueba.

El primer comando crea los índices para tomar las muestras. El 2do toma los datos normalizados y los une por columna en un data set, y el tercero crea los data sets de entrenamiento y prueba.

```
> if(!(require(caret))) install.packages("caret")
> set.seed(params$seed_train)
> idx_train <- createDataPartition(y=data$Class, p=params$ptrain, list=F)
> dim(idx_train)
```

```
[1] 457   1
```

```
> c_nrm <- cbind(data_norm[,1:9], Class=data[,10])
> dim(c_nrm)
```

```
[1] 683  10
```

```
> c_train <- c_nrm[idx_train,]
> c_test <- c_nrm[-idx_train,]
> dim(c_train)
```

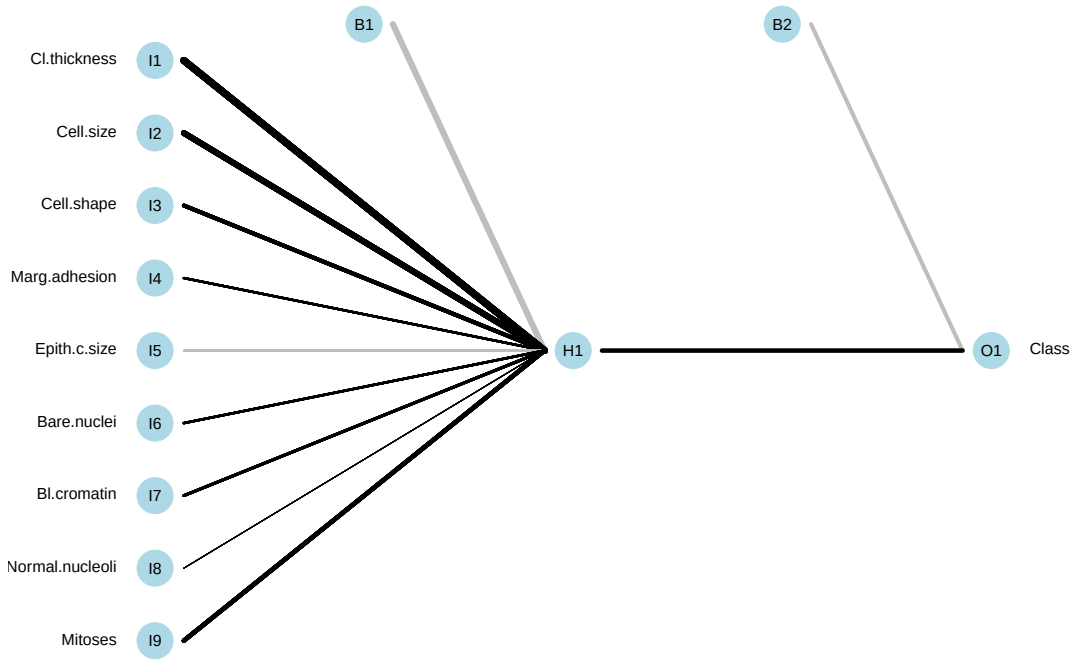
```
[1] 457  10
```

```
> dim(c_test)
```

```
[1] 226  10
```

```
> library(caret)
> c_model <- caret::train(Class ~ ., c_train, method="nnet",
+                  trControl=trainControl(method="none"),
+                  tuneGrid=NULL, tuneLength=1, trace=F)
```

```
> plotnet(c_model)
```




```
> summary(c_model)
```

```
a 9-1-1 network with 12 weights
options were - entropy fitting
 b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1 i6->h1 i7->h1 i8->h1 i9->h1
-45.90  52.08  42.38  27.43  14.70  -9.69  15.57  19.22   4.19  31.06
  b->o  h1->o
-29.31  32.60
```

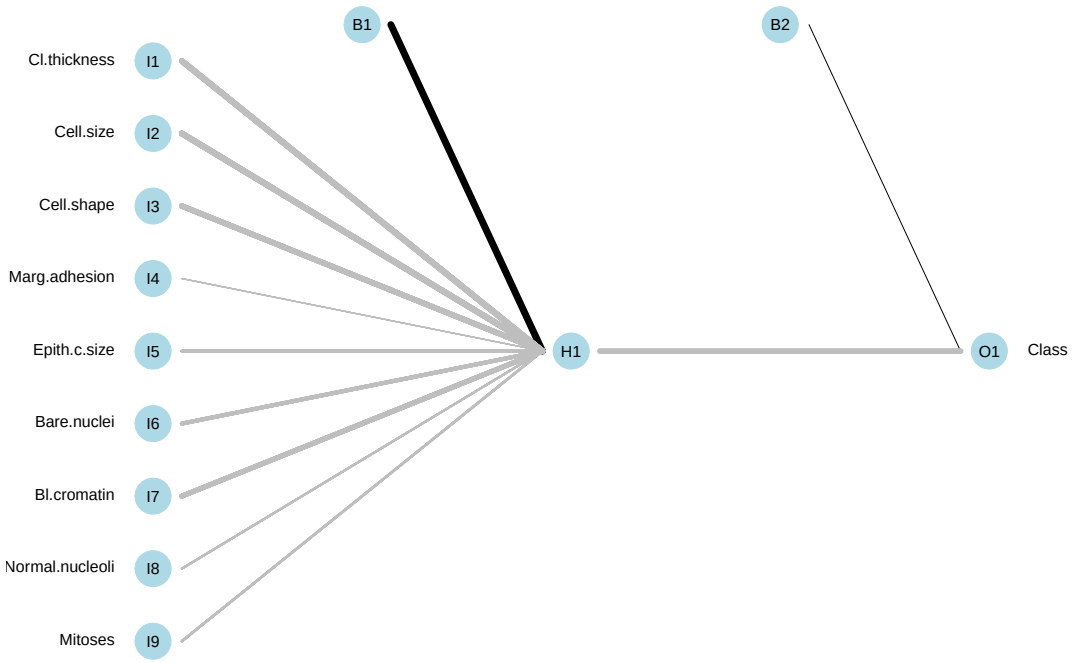
```
> library(caret)
> # Si se coloca type="prob" en las opciones,
> # retorna la probabilidad para cada clase
>
> c_pred <- predict(c_model, c_test[-10])
> table(c_pred, c_test$Class)
```

```
c_pred      benign malignant
  benign       141         2
  malignant      6        77
```

7.1.- 5 fold crossvalidation

```
> cv_model <- caret::train(Class ~ ., c_train, method="nnet",
+                  trControl=trainControl(method="cv"),
+                  tuneGrid=NULL, tuneLength=1, trace=F)
```

```
> plotnet(cv_model)
```




```
> summary(cv_model)
```

```
a 9-1-1 network with 12 weights
```

```
options were - entropy fitting
 b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1 i6->h1 i7->h1 i8->h1 i9->h1
 30.21 -24.67 -24.89 -22.83  -3.82 -10.79 -16.20 -20.44  -7.76  -9.17
  b->o  h1->o
  3.13 -25.11
```

```
> # Si se coloca type="prob" en las opciones,
> # retorna la probabilidad para cada clase
>
> cv_pred <- predict(cv_model, c_test[-10])
> table(cv_pred, c_test$Class)
```

```
cv_pred     benign malignant
  benign       140         2
  malignant      7        77
```

Referencias

Lantz, Brett. 2013. *Machine Learning with R*. Packt Publishing Ltd.