

5. ANN

Predicción de la malignidad/benignidad de cáncer de mama

José Félix Rojas Cabeza

28 de April 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
3. Preparación de los datos - Creando datasets de entrenamiento y de prueba.	6
4.- Entrenamiento del modelo en los datos	7
5. Evaluación del desempeño del modelo	11
6. Paquete <code>caret</code>	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 envían señales (información). Se crea una red con diferentes nodos y capas interconectados para procesar la información. Cada capa está 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 las 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 atrás 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)
```

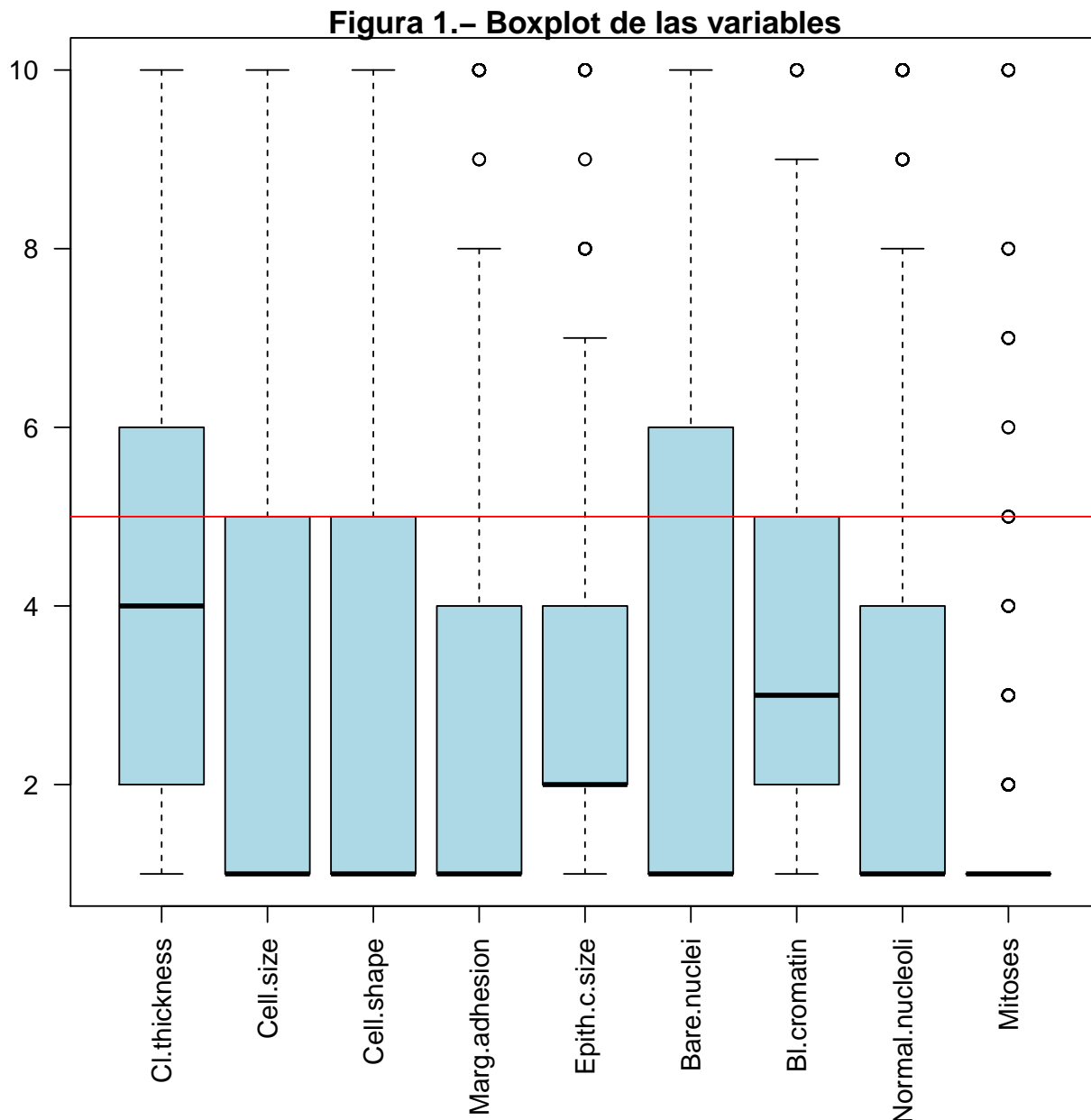
```
'data.frame': 683 obs. of 10 variables:
 $ Cl.thickness : int 5 5 3 6 4 8 1 2 2 4 ...
 $ Cell.size : int 1 4 1 8 1 10 1 1 1 2 ...
 $ Cell.shape : int 1 4 1 8 1 10 1 2 1 1 ...
 $ Marg.adhesion : int 1 5 1 1 3 8 1 1 1 1 ...
 $ Epith.c.size : int 2 7 2 3 2 7 2 2 2 2 ...
 $ Bare.nuclei : int 1 10 2 4 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			

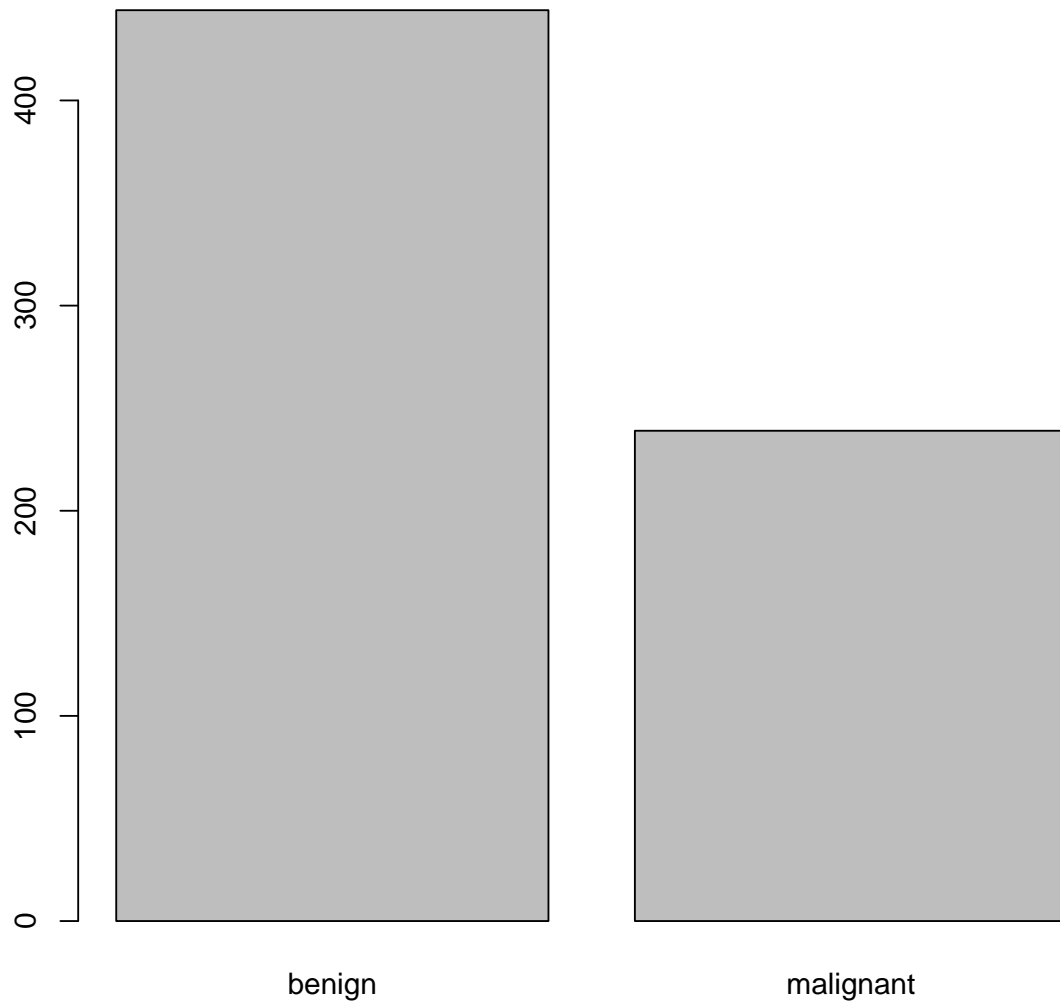
El archivo tiene 683 observaciones y 10 variables. Se observa que no hay datos faltantes, por lo que no hay que ajustar `data`. Los datos son numéricos, posiblemente enteros, excepto en el caso de la última columna (Clase), que tiene dos factores. A continuación se presenta la distribución de los datos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

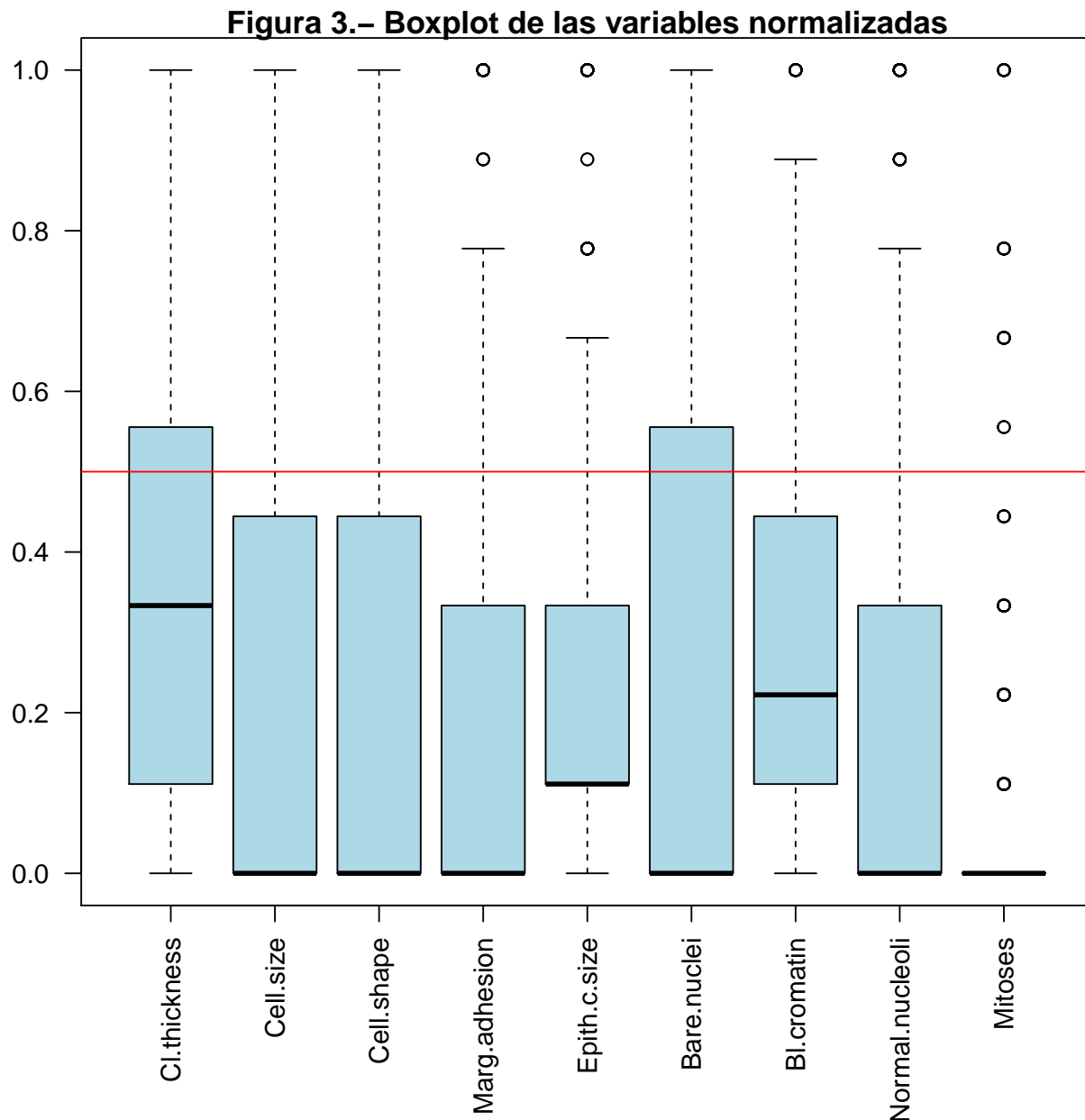
Cl.thickness	Cell.size	Cell.shape	Marg.adhesion
Min. :0.0000	Min. :0.0000	Min. :0.0000	Min. :0.0000
1st Qu.:0.1111	1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.:0.0000
Median :0.3333	Median :0.0000	Median :0.0000	Median :0.0000
Mean :0.3825	Mean :0.2390	Mean :0.2461	Mean :0.2034
3rd Qu.:0.5556	3rd Qu.:0.4444	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 normalizadas")
> abline(h=0.5, col="red")
```



3. 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 (142, 51, 220, 664, 605, 587, 352, 216, 86, 75, 38, 615, 465, 40, 286, 257, 506, 12, 393, 14, 653, 148, 618, 528, 592, 62, 480, 354, 500, 635, 572, 537, 36, 166, 649, 145, 648, 570, 106, 451, 13, 579, 433, 586, 56, 363, 504, 91, 628, 546, 621, 567, 234, 255, 535, 154, 439, 399, 46, 471, 90, 124, 472, 517, 638, 580, 377, 323, 74, 7, 554, 346, 311, 544, 449, 681, 432, 632, 647, 3, 18, 453, 458, 249, 404, 396, 324, 419, 390, 376, 543, 355, 232, 176, 652, 412, 76, 512, 92, 356, 520, 146, 599, 341, 21, 47, 267, 682, 71, 63, 244, 110, 253, 639, 468, 37, 577, 10, 455, 77, 338, 129, 485, 563, 525, 175, 558, 240, 658, 218, 378, 602, 680, 287, 60, 555, 66, 245, 369, 675, 601, 673, 205, 88, 247, 215, 394, 82, 94, 123, 81, 424, 616, 591, 441, 351, 159, 296, 137, 271, 530, 322, 276, 415, 197, 654, 229, 677, 662, 156, 381, 522, 339, 328, 466, 421, 450, 171, 238, 277, 164, 132, 515, 523, 595, 490, 235, 111, 27, 65, 127, 334, 391, 585, 503, 17, 289, 177, 438, 660, 149, 195, 491, 310, 15, 206, 405, 481, 73, 609, 314, 126, 431, 104, 402, 414, 345, 620, 331, 661, 610, 187, 58, 326, 227, 379, 83, 280, 630, 24, 437, 418, 556, 263, 655, 564, 217, 336, 295, 342, 309, 162, 417, 202, 265, 509, 416, 133, 406, 200, 52, 72, 68, 427, 646, 423, 186, 374, 258, 568, 266, 368, 348, 547, 511, 370, 275, 161, 590, 561, 440, 16, 454, 430, 589, 113, 683, 297, 349, 44, 508, 459, 115, 463, 627, 107, 315, 617, 475, 538, 469, 397, 576, 57, 428, 6, 169, 288, 210, 573, 184, 305, 225, 279, 552, 645, 434, 151, 319, 548, 1, 167, 408, 239, 201, 636, 494, 212, 193, 593, 178, 666, 34, 582, 283, 261, 387, 231, 545, 435, 607, 598, 316, 254, 188, 230, 53, 268, 41, 291, 259, 583, 313, 446, 656, 541, 61, 667, 290, 204, 281, 272, 237, 59, 109, 135, 224, 457, 375, 293, 67, 571, 189, 118, 85, 569, 663, 89, 479, 172, 32, 155, 650, 180, 157, 335, 373, 507, 383, 400, 371, 112, 214, 641, 173, 382, 409, 307, 248, 357, 566, 250, 624, 35, 301, 25, 343, 340, 668, 329, 643, 436, 8, 398, 395, 401, 350, 483, 285, 612, 26, 273, 608, 54, 23, 380, 623, 264, 496, 492, 260, 505, 108, 103, 97, 330, 226, 152, 501, 190, 198, 486, 102, 28, 413, 532, 223, 389, 516, 306, 140, 33, 425, 574, 163, 298, 372, 30, 294, 499, 361, 170, 117, 122, 386) 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

```

4.- Entrenamiento del modelo en los datos

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

4.1.- Una neurona escondida

```

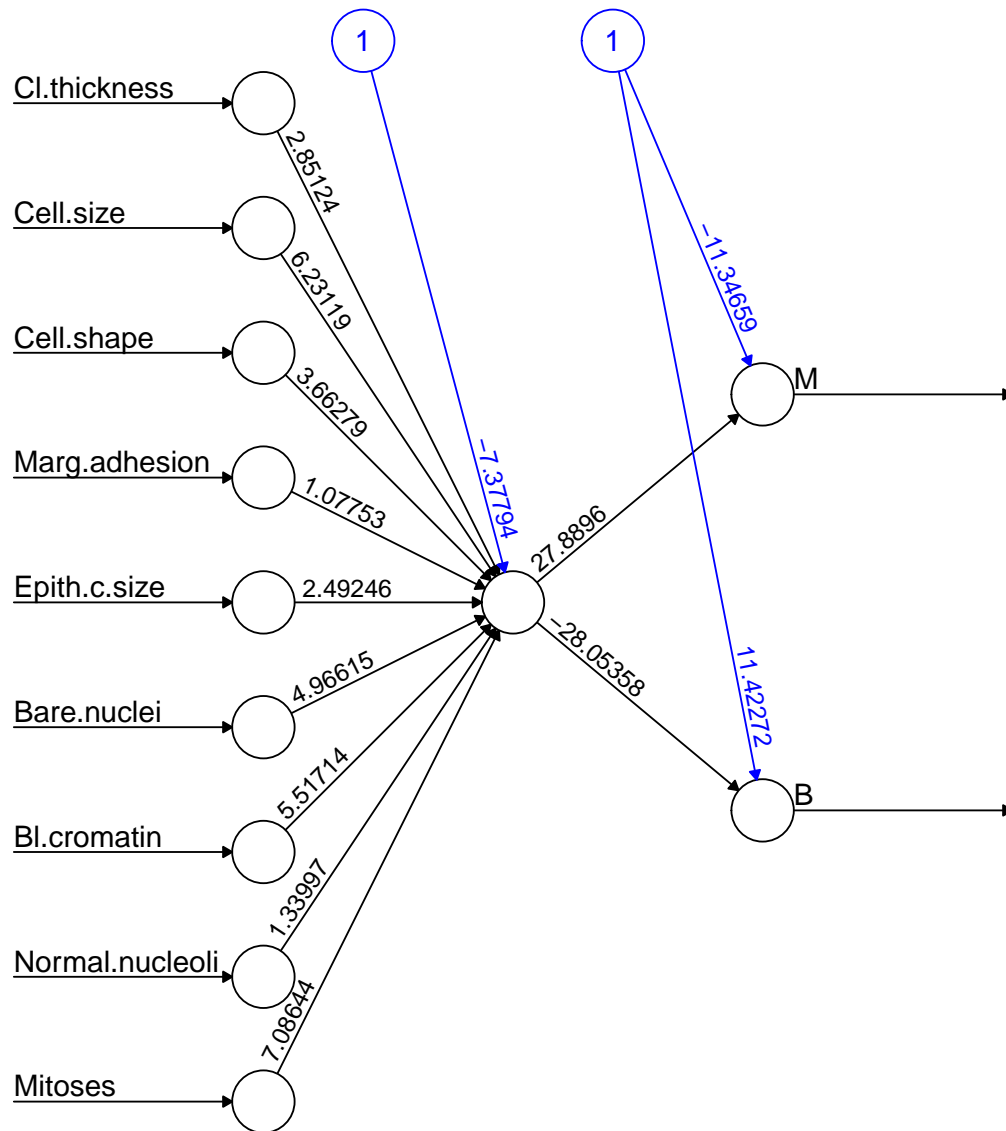
> if(!(require(neuralnet))) install.packages("neuralnet")
> if(!(require(NeuralNetTools))) install.packages("NeuralNetTools")
> set.seed(params$seed_neuralnet)
>
> library(neuralnet)
> library(NeuralNetTools)
>
> frmla = M+B ~ Cl.thickness+Cell.size+Cell.shape+Marg.adhesion+Epith.c.size+
+          Bare.nuclei+B1.cromatin+Normal.nucleoli+Mitoses
>

```

```

> # una sola neurona escondida
> data_model_1 <- neuralnet(frmla, data=data_norm_train, hidden=1, linear.output = F)
>
>
> plot(data_model_1, rep = "best")

```

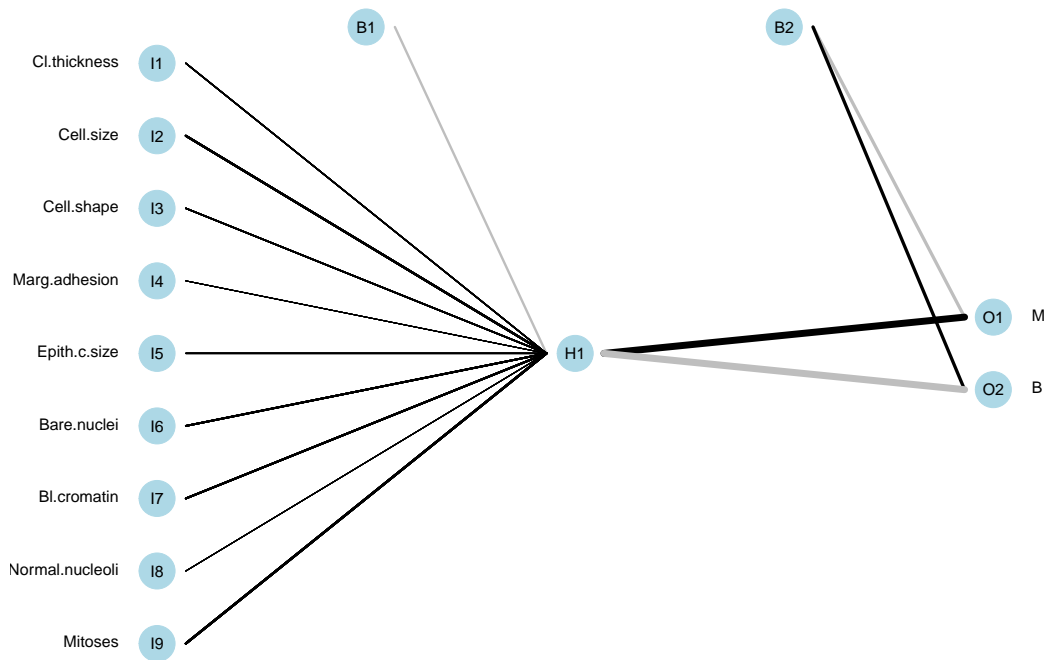


Error: 6.019217 Steps: 496

```

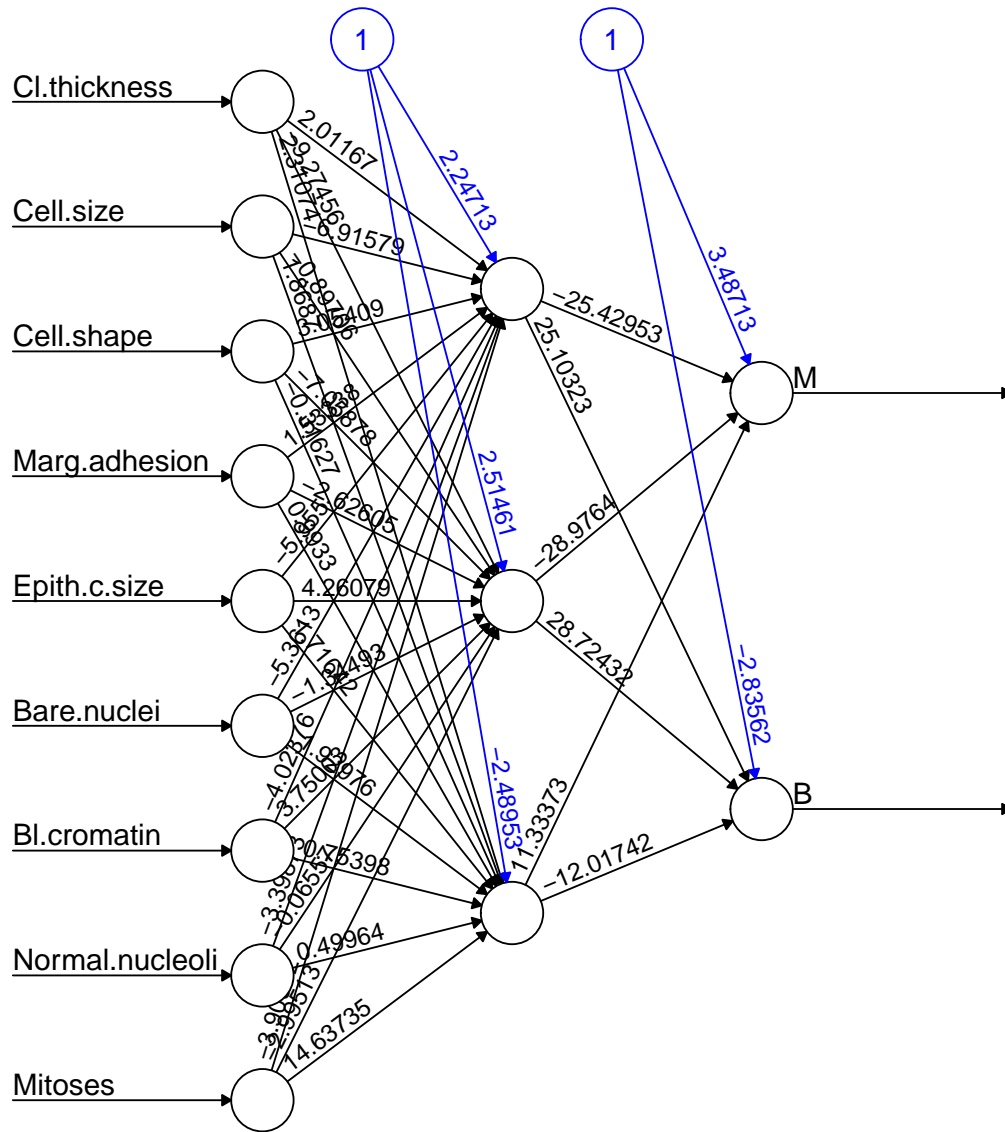
> # Hay que cambiar el tamaño de foto para que no corte los nombres!
> plotnet(data_model_1)

```

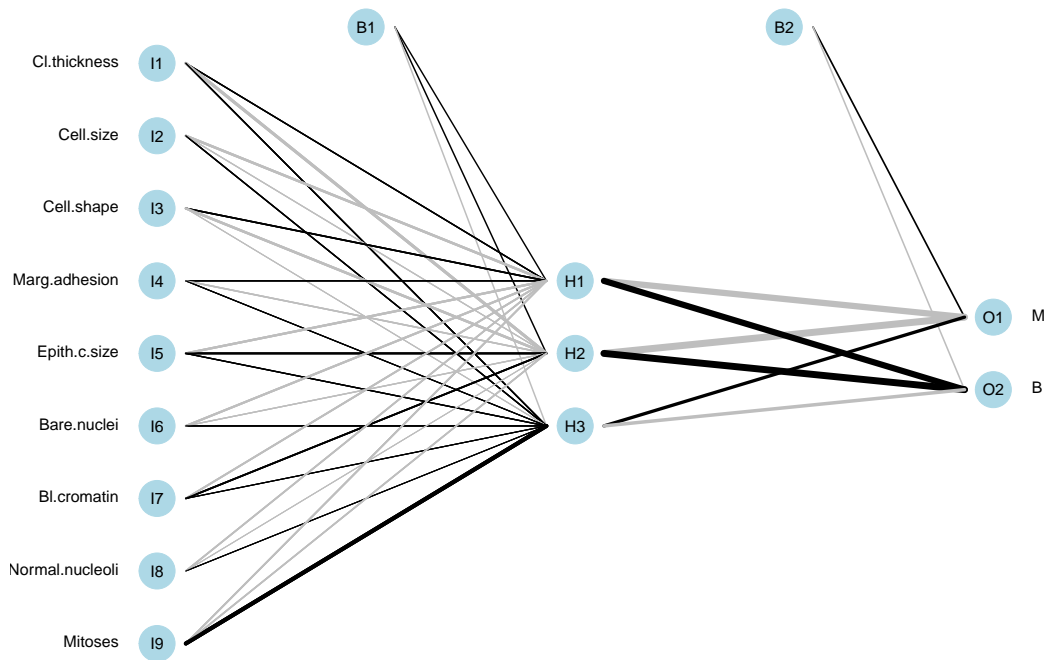
4.2.- Tres neuronas escondidas

```
> # 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)
```



5. Evaluación del desempeño del modelo

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

5.1 Matriz de confusión del modelo con 1 nodo

```
> library(neuralnet)
> # debido a que se discontinuó 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 sensibilidad y especificidad de 98.6486486% y 94.8051948% respectivamente.

5.2 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
```

5.3 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.

6. 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
```

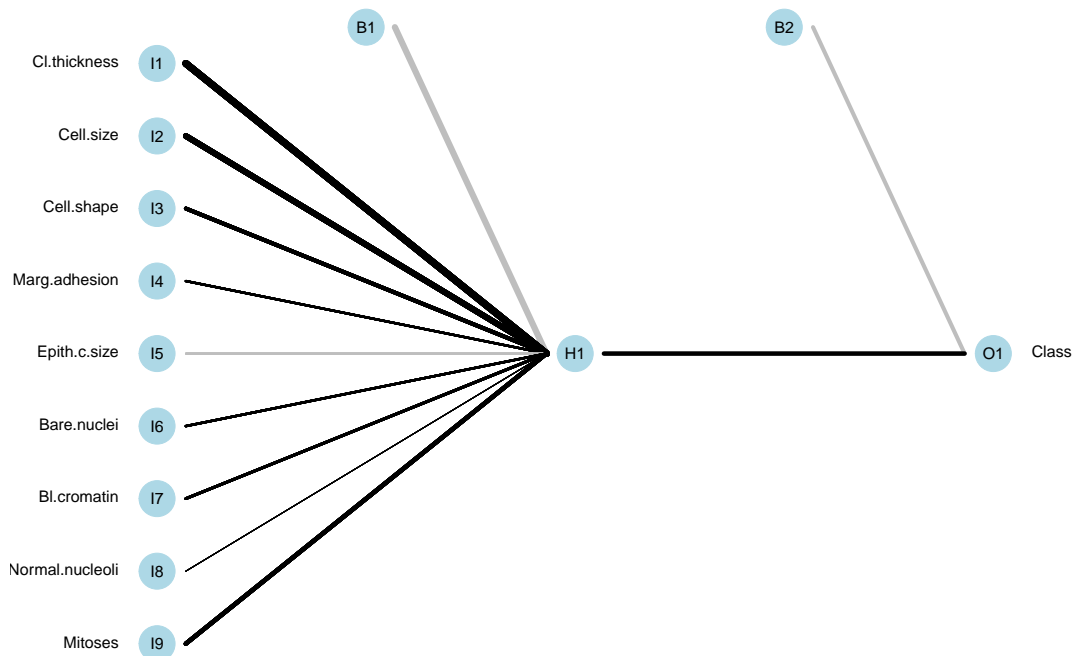
```
> c_train <- c_nrm[idx_train,]
> c_test <- c_nrm[-idx_train,]
> dim(c_train)
```

```
[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
```

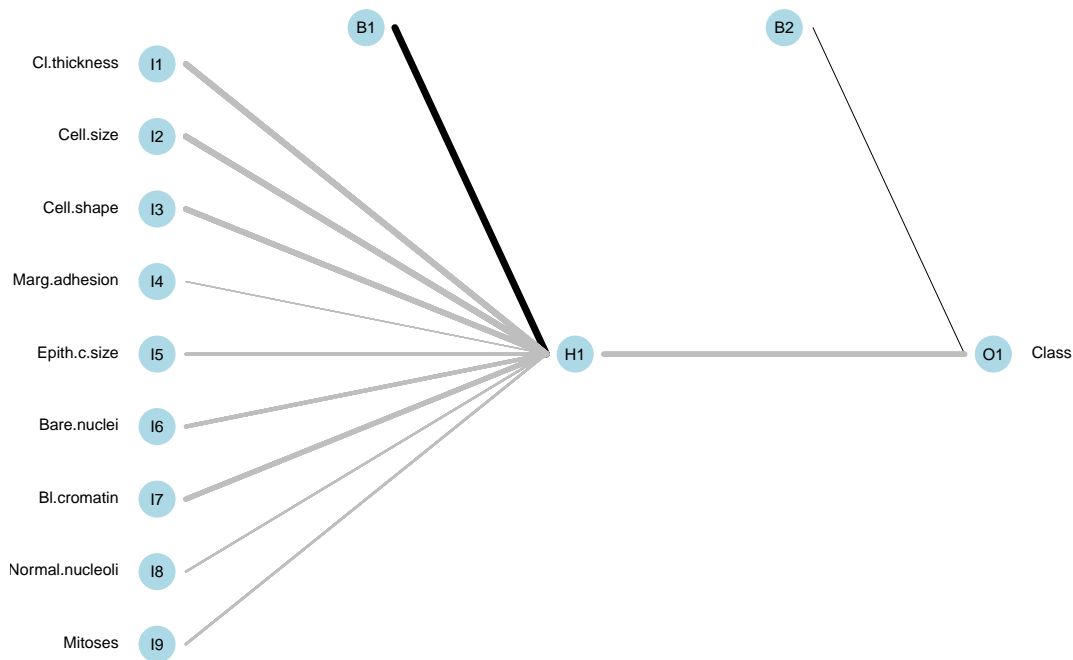
```
b->o h1->o
-29.31 32.60
```

```
> 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
```

6.1.- 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.