Informe de práctica II
Redes Neuronales
Predicción de supervivencia en casos de cáncer de mama

Ana Medina García

9 de marzo de 2016

Introducción

El problema de la predicción de metástasis

Distintas pacientes de cáncer de mama en un mismo estadio de la enfermedad pueden tener respuestas muy distintas al tratamiento y resultados globales muy diferentes.Uno de los problemas más importantes en este campo es la predicción del riesgo de metástasis; la quimioterapia o la terapia hormonal reducen este riesgo, sin embargo, el 70-80 % de las pacientes que reciben este tratamiento, podrían haber sobrevivido sin el mismo.

En esta práctica realizaremos análisis de datos de microarrays en pacientes jóvenes de cáncer de mama, y aplicaremos clasificación supervisada para identificar una una firma de expresión génica fuertemente predictiva de una metástasis a corto plazo.

Redes Neuronales

Aplicaremos aprendizaje supervisado, mediante el diseño de una red neuronal, para realizar la predicción de metástasis en un período de cinco años en casos de tumor de mama.

Las redes de neuronas artificiales son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas que colaboran entre sí para producir un estímulo de salida.

Las características fundamentales de las redes neuronales son son:

- **Aprenden de la experiencia:** Las RNA pueden modificar su comportamiento como respuesta a su entorno. Dado un conjunto de entradas (quizá con las salidas deseadas), las RNA se ajustan para producir respuestas consistentes. Una amplia variedad de algoritmos de entrenamiento se han desarrollado, cada uno con sus propias ventajas e inconvenientes.

- **Generalizan de ejemplos anteriores a los ejemplos nuevos:** Una vez que la RNA esté entrenada, la respuesta de la red puede ser, hasta un cierto punto, insensible a pequeñas variaciones en las entradas, lo que las hace idóneas para el reconocimiento de patrones.
- **Abstracción de la esencia de las entradas:** Algunas RNA son capaces de abstraer información de un conjunto de entradas. Por ejemplo, en el caso de reconocimiento de patrones, una red puede ser entrenada en una secuencia de patrones distorsionados de una letra. Una vez que la red sea correctamente entrenada será capaz de producir un resultado correcto ante una entrada distorsionada, lo que significa que ha sido capaz de aprender algo que nunca había visto.

Sistemas y Métodos

Conjunto de datos de cáncer de mama

Trabajaremos sobre un conjunto de 78 muestras de pacientes con cáncer de mama: 34 que desarrollaron metástasis en un periodo de 5 años, y 44 que continuaron libres de la enfermedad pasados los 5 años. Todas las pacientes dieron negativo en la afeccción de nodos linfáticos y eran menores de 55 años.

Para ello, en primer lugar, debemos crear el paquete con los datos de microarrays de ADN del estudio realizado por Van't Veer *et al.*, y guardarlo de forma local. Una vez hecho esto, ya podemos importar este paquete y cargar los datos para trabajar con ellos. Se trata de un conjunto de datos de expresión génica de 98 muestras y 24481 genes para cada una.

El paquete contiene datos fenotípicos, datos de expresión y del experimento. Por lo tanto, lo siguiente que haremos será extraer los datos de expresión de las 78 muestras que nos interesan (las 34 clasificadas como metástasis y las 34 como no metástasis), y los datos de la variable de estudio (la metástasis) que los estraemos de los datos fenotípicos.

Una vez extraídos los datos necesarios, los reorganizamos de acuerdo al número de muestra y creamos un "data frame" con los mismos para facilitar su posterior procesamiento. También comprobamos el número de casos de cada tipo:

```
> ###Carga de los datos de Van't Veer et al.
> library(seventyGeneData)
> data(vantVeer)
> library(Biobase)
> ###Búsqueda de las 78 muestras
> met5y <- pData(vantVeer)$DataSetType
> less5year <- which(met5y=="less_than_5y")
> greater5year <- which(met5y=="greater_than_5y")
> tumors78 <- sort(c(less5year,greater5year))
> ###Subconjunto de datos con sólo esas 78 muestras
> vantVeer78tumors <- vantVeer[,tumors78]
> ###Extracción de los datos fenotípicos
> pData78 <- pData(vantVeer78tumors)
> ###Extracción del vector que determina la clase
> dataType <- pData78$DataSetType
> dataType <- as.factor(dataType)
> ###Reordenación de acuerdo al número de muestra
```

```
> dataType <- dataType[order(pData78$Sample)]
> ####CLASE A ESTUDIAR: variable dataType
> summary(dataType)


greater_than_5y    less_than_5y
             44              34


> ###Extracción de los datos de expresión
> expData78 <- t(exprs(vantVeer78tumors))
> ###Reordenación de los datos (78 observaciones de 24481 variables)
> samples <- rownames(expData78)
> samples.num <- as.numeric(sub(".* ", "", samples))
> rownames(expData78) <- samples.num
> expData78 <- expData78[order(as.numeric(rownames(expData78))), ]
> row.names(expData78) <- seq(nrow(expData78))
>
```

Pre-procesamiento de datos

Eliminación de datos perdidos

Al explorar la estructura de datos nos encontramos con que existen muchos datos perdidos que se registran como *" NaN"*, lo cual, además se suponer un problema para el procesamiento de los mismos, hace, junto con el detalle de utilizar comas para los decimales, que los números se encuentren almacenados como tipo *factor*.

Para poder realizar estudios sobre las mediciones de expresión, debemos eliminar los valores perdidos y modificar los datos para poder almacenarlos como tipo numérico.

Mediante una función, podemos mostrar a coninuación el número de datos perdidos que tenemos por cada muestra:

```
> ###Transformación de los datos de expresión a numérico
> expData78 <- as.data.frame(expData78)
> expData78[expData78=="   NaN"] <- NA
> expData78 <- as.data.frame(lapply(expData78, function(x) sub(",", ".", x)))
> expData78 <- as.data.frame(lapply(expData78, as.character), stringsAsFactors=FALSE)
> expData78 <- as.data.frame(lapply(expData78, as.numeric))
> ###Función para contar los NA en cada fila
> find.na <- function(data){
+   nas <- vector()
+   for(i in 1:78){
+     nas <- c(nas, length(which(is.na(data[i,]))))
+   }
+   return(nas)
+ }
> row.nas <- find.na(expData78)
> row.nas
```

```
 [1]   293   293   300   340   293   293   293   293   293   298   293   377
[13]   293   293   293   293   318   301   298   293   293   295   333   322
[25]   294   309   293   324   295   293   319   353   336   293   293   295
[37]   293   407   301   293   301   327   293   294   293   293   293   293
[49]   293   293   296   293  2397 10896   326   296   431   293   293   296
[61]   326   293   293   293   294   293   325   302   308   293   293   293
[73]   293   298   293   297   312   294

>
```

Podemos darnos cuenta de que en la mayoría de las muestras el número de NA's (valores perdidos) es 293 o muy cercano. Comprobamos si se trata de los mismos genes en todas ellas y descubrimos que sí; las expresiones de esos 293 genes se repiten como valores perdidos para todas, o casi todas, las muestras. Por lo tento eliminaremos estos genes del conjunto quedándonos de esta forma con 24188 genes.

Una vez hecho esto, volvemos a mostrar el número de valores perdidos por muestra:

```
> omitedGenes <- which((is.na(expData78[1,])))
> expData78clean <- expData78[ ,-omitedGenes]
> row.nas.cleaned <- find.na(expData78clean)
> row.nas.cleaned

 [1]     0     0     7    47     0     0     0     0     0     5     0    84
[13]     0     0     0     0    25     8     5     0     0     2    40    29
[25]     1    16     0    31     2     0    26    60    43     0     0     2
[37]     0   114     8     0     8    34     0     1     0     0     0     0
[49]     0     0     3     0  2104 10603    33     3   138     0     0     3
[61]    33     0     0     0     1     0    32     9    15     0     0     0
[73]     0     5     0     4    19     1

>
```

El resto de valores perdidos que podemos ver, los eliminamos por imputación, utilizando el valor medio de expresión del gen en cuestión en cada caso. Para ello utilizamos la siguiente función:

```
> for(i in 1:24188){
+   genAux <- expData78clean[,i]
+   meanExp <- summary(genAux)[4]
+   expData78clean[is.na(expData78clean[,i]),i] <- meanExp
+ }
```

Ya podemos comprobar que hemos conseguido eliminar los valores perdidos entre los datos:

```
> row.nas.imp <- find.na(expData78clean)
> row.nas.imp
```

```
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[39] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[77] 0 0

>
```

Normalización

El siguiente paso es la normalización y/o escalado de los datos. En nuestro caso, vamos a aplicar un procedimiento de normalización sencillo de forma que transformamos los datos a una distribución en el intervalo [0,1] según la siguiente fórmula:

$$v = x - min(x)/max(x) - min(x)$$

```
> ###Normalización de datos
> expData78.norm <- sapply(expData78clean[,-24189], Normalize <- function(dataSet){
+   datanorm <-( (dataSet-min(dataSet)) / (max(dataSet)-min(dataSet)) )
+ })
> expData78.norm <- as.data.frame(expData78.norm)
```

Filtrado

A continuación realizaremos un filtrado de los datos con el fin de seleccionar un subconjunto de genes con el que poder construir el modelo basado en redes neuronales para la predicción de metástasis.

Vamos a utilizar para el filtrado, métodos similares a los realizados por Van't Veer *et al.* en su estudio, de tal forma que llevaremos a cabo varios pasos.

En primer lugar, utilizamos un filtro de T-test para escoger los genes diferencialmente expresados para la clase, con un p-valor de 0.1:

```
> #Conversión de la clase a binario ("greater_than_5y" = 0, "less_than_5y" = 1)
> dataType.binary <- (dataType =="less_than_5y")*1
> library(genefilter)
> filter <- ttest(dataType.binary, p=0.1)
> selection <- genefilter(t(expData78.norm), filterfun(filter))
> sum.ttest <- sum(selection)
> sum.ttest

[1] 4096

> expData78.filt <- expData78.norm[,selection]
```

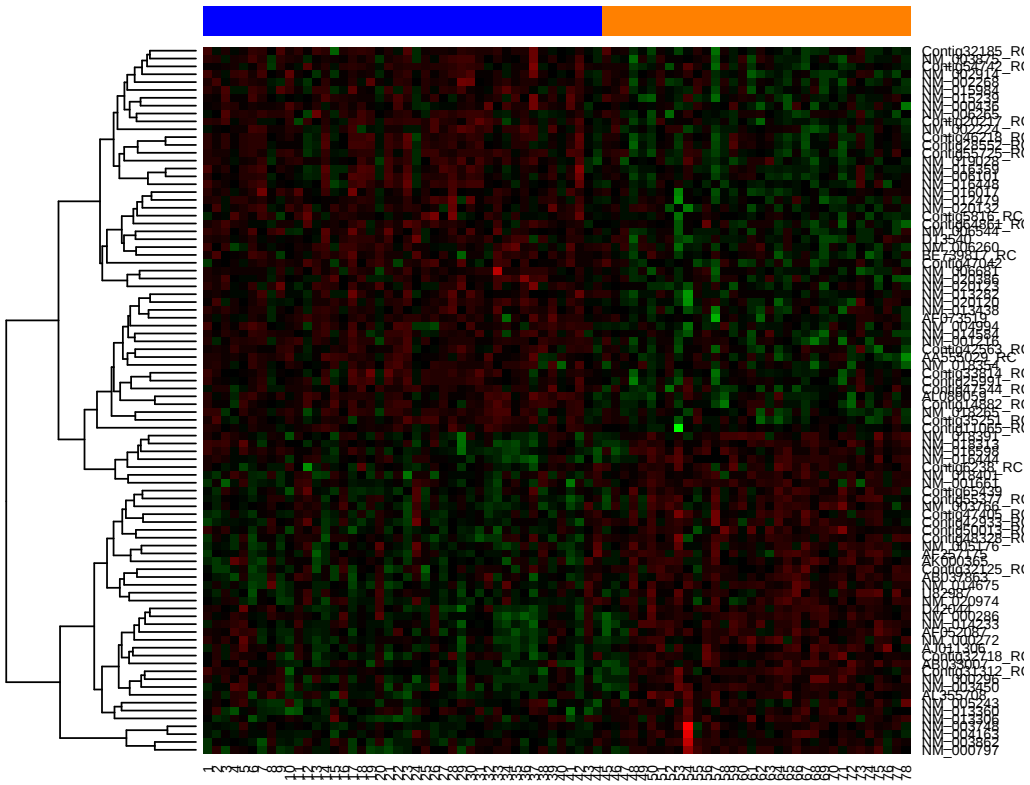
De esta forma, obtenemos un subconjunto de 4096 genes.

En segundo lugar, ordenamos los genes de este subconjunto por su grado de correlación con la clase, y escogemos los 80 primeros:

```
> #cálculo de la correlación
> correlation <- sapply(expData78.filt, function(i){
+   cor(i,dataType.binary)
+ })
> #Número de genes que queremos seleccionar
> gene.number <- 90
> #Ordenación por correlación (en valor absoluto) y selección de los 200 primeros
> best.correlation <- sort(abs(correlation), decreasing = TRUE)[1:gene.number]
> #Nombres de los genes seleccionados
> best.correlation.names <- names(best.correlation)
> #Extracción de la matriz de datos de los genes seleccionados
> expData78.best90.corr <- expData78.filt[,best.correlation.names]
>
```

Podemos visualizar la expresión de estos genes mediante el siguiente mapa de calor. En la barra superior los casos de metástasis son los representados con el color naranja (34 casos) y los que no desarrollaron metástasis son los representados con el color azul (44 casos).

```
> ## Visualización la expresión de los genes seleccionados en un heatmap
> library(gplots)
> color.map <- function(class) { if (class=="greater_than_5y") "#0000FF" else "#FF8000" }
> classcolors <- unlist(lapply(dataType, color.map))
> heatmap(t(as.matrix(expData78.best90.corr)), Colv=NA, col=redgreen(75), ColSideColors=classc
>
```



Redes Neuronales para predicción

Conjunto de 90 genes seleccionado

Diseñamos el modelo de red neuronal, lo entrenamos y realizamos la predicción con el mismo:

```
> expData78.best90.corr$Class <- dataType
> library(nnet)
> nn.fit <- nnet(Class~., data=expData78.best90.corr, size=3, maxit=20)

# weights:  277
initial  value 54.261409
iter  10 value 32.014922
iter  20 value 22.184137
final  value 22.184137
stopped after 20 iterations

> nn.pred <- predict(nn.fit, expData78.best90.corr, type="raw")
```

A continuación mostramos la curva ROC:

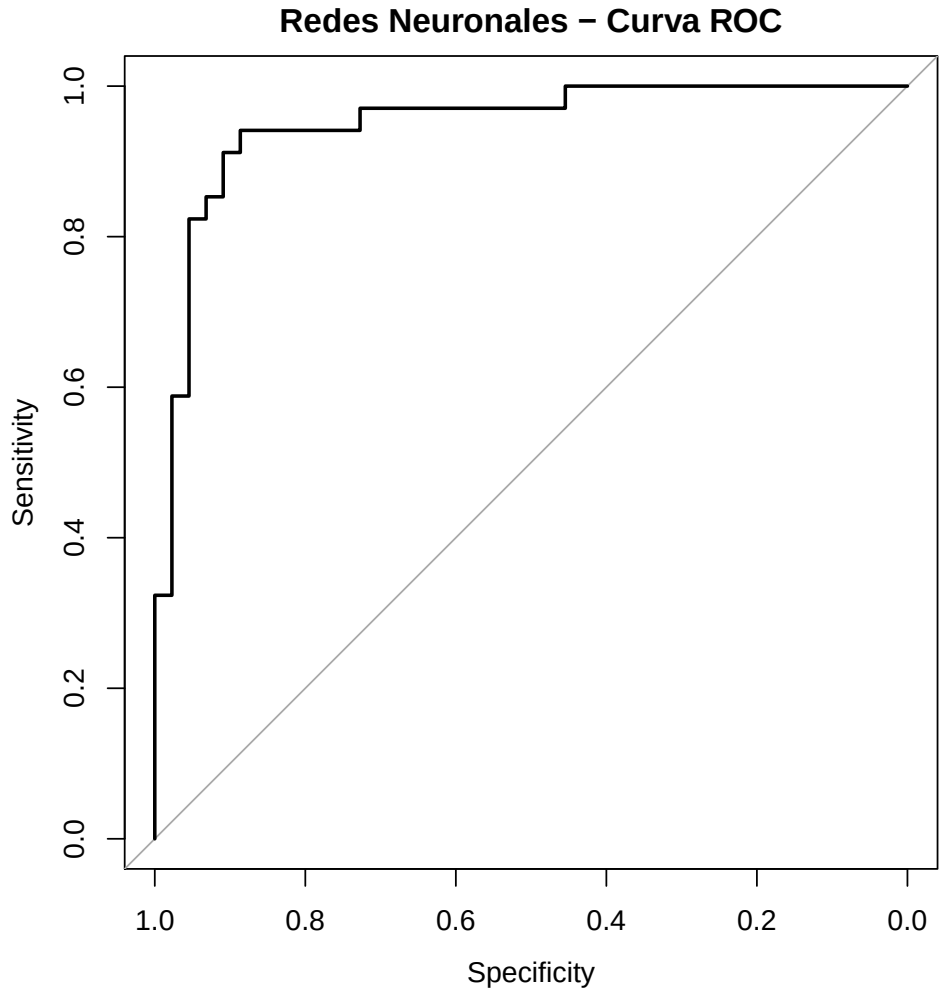
```
> library(pROC)
> nn.roc <- roc(expData78.best90.corr$Class, nn.pred, smooth=FALSE, auc=TRUE)
> auc(nn.roc)

Area under the curve: 0.9485

> plot(nn.roc, main="Redes Neuronales - Curva ROC")

Call:
roc.default(response = expData78.best90.corr$Class, predictor = nn.pred,     smooth = FALSE, a

Data: nn.pred in 44 controls (expData78.best90.corr$Class greater_than_5y) < 34 cases (expData
Area under the curve: 0.9485
```



Conjunto de 70 genes de Van't Veer

Extraemos el conjunto de datos de los 70 genes seleccionados por Van't Veer *et al.* y los preprocesamos y normalizamos:

```
> gene70 <- which(featureData(vantVeer)@data$genes70)
> expData.vantveer70 <- expData78[,gene70]
```

```
> for(i in 1:70){
+   genAux <- expData.vantveer70[,i]
+   meanExp <- summary(genAux)[4]
+   expData.vantveer70[is.na(expData.vantveer70[,i]),i] <- meanExp
+ }
> expData.vv70.norm <- sapply(expData.vantveer70, Normalize <- function(dataSet){
+   datanorm <-( (dataSet-min(dataSet)) / (max(dataSet)-min(dataSet)) )
+ })
> expData.vv70.norm <- as.data.frame(expData.vv70.norm)
> expData.vv70.norm$Class <- dataType
```

Diseñamos el modelo de red neuronal y llevamos a cabo la predicción con el mismo:

```
> nn70.fit <- nnet(Class~., data=expData.vv70.norm, size=3, maxit=20)
```

```
# weights:  217
initial  value 56.431869
iter  10 value 29.429179
iter  20 value 16.259365
final  value 16.259365
stopped after 20 iterations
```

```
> nn70.pred <- predict(nn70.fit, expData.vv70.norm, type="raw")
```

A continuación mostramos la curva ROC:

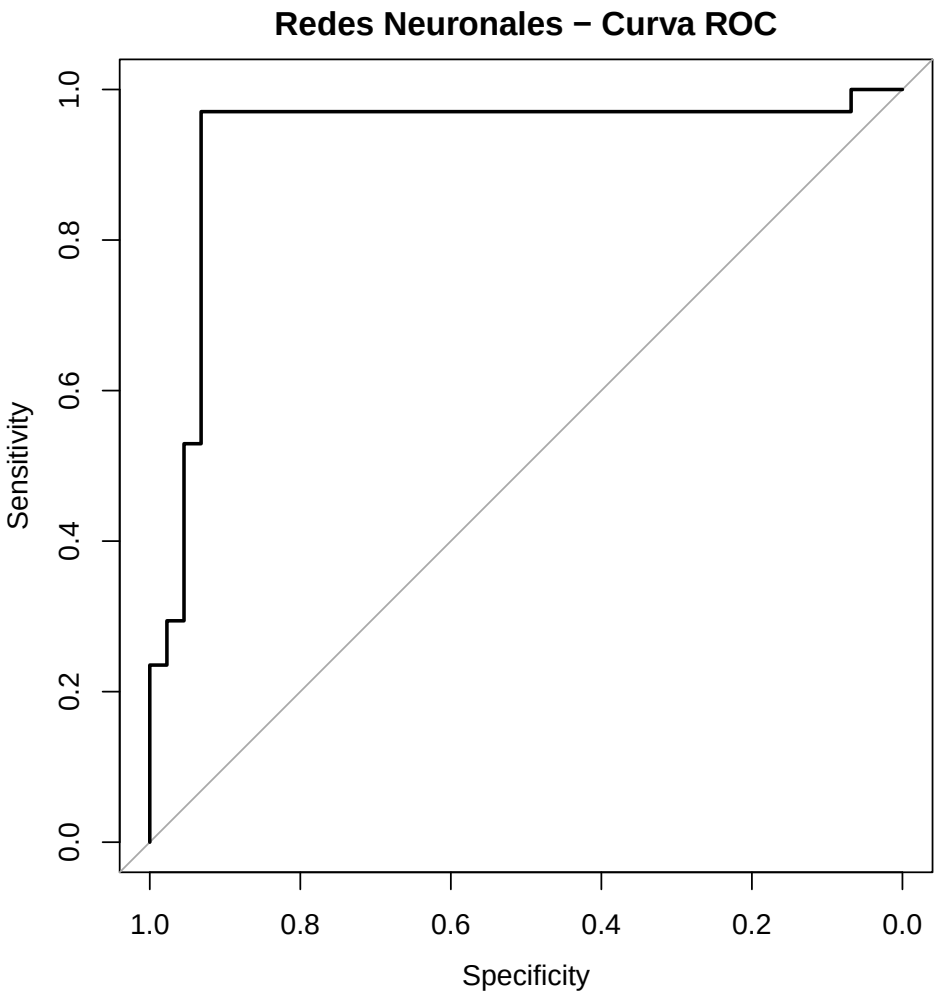
```
> nn70.roc <- roc(expData.vv70.norm$Class, nn70.pred, smooth=FALSE, auc=TRUE)
> auc(nn70.roc)
```

```
Area under the curve: 0.9305
```

```
> plot(nn70.roc, main="Redes Neuronales - Curva ROC")
```

```
Call:
roc.default(response = expData.vv70.norm$Class, predictor = nn70.pred,     smooth = FALSE, auc

Data: nn70.pred in 44 controls (expData.vv70.norm$Class greater_than_5y) < 34 cases (expData.v
Area under the curve: 0.9305
```


Redes Neuronales − Curva ROC
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.8
0.6
0.4
0.2
0.0
Specificity