INFORME
REDES NEURONALES

TÉCNICAS ESTADÍSTICAS EN ANÁLISIS DE MERCADOS

17/06/2013

VICTORIA FORNÉS FERRER
GRADO EN ESTADÍSTICA EMPRESARIAL

UNIVERSITAS Miguel Hernández

ÍNDICE

BIBLIOGRAFÍA

Aplicaciones de las redes neuronales artificiales a la estadística

Mª Luisa Pérez Delgado, Quintín Martín Martín

Editorial: La Muralla, S.A.

INTRODUCCIÓN

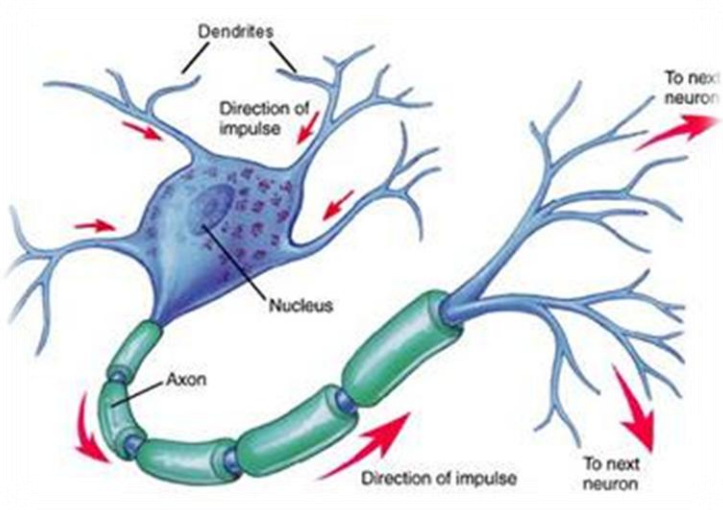
Durante este curso hemos visto y aplicado diferentes métodos para el análisis de datos y la extracción de información en la asignatura de Técnicas Estadísticas en Investigación de Mercados; y por último vamos a utilizar la modelización matemática basada en las Redes Neuronales Artificiales.

Empezaremos por la base, explicando el funcionamiento de las redes neuronales biológicas, y con esto podremos entender la teoría de las redes neuronales artificiales. A continuación utilizaremos la base de datos CSS de la librería BCA del programa R-project. De este modo analizaremos dicha base de datos con una de las técnicas estadísticas más modernas de esta ciencia.

Introducción a las redes neuronales biológicas

Las Redes Neuronales Artificiales (RNA) se inspiran en el funcionamiento de las Redes Neuronales Biológicas. Comenzaremos con una breve descripción de la estructura y funcionamiento de los sistemas biológicos. Esto servirá para comprobar el paralelismo que se intenta establecer en los sistemas artificiales.

El sistema nervioso es un sistema complejo, formado por unidades individuales denominadas neuronas, unidas entre sí por una malla de fibras nerviosas. Las neuronas están separadas estructural, metabólica y funcionalmente y pueden tener tamaños y formas muy variados. Forman redes entretejidas muy elaboradas, cuyas funciones varían en diferentes regiones del sistema. Estas diferencias de formas y tamaños reflejan el modo en que las neuronas procesan la información. Aunque, como hemos dicho, las neuronas puedan presentar múltiples formas, muchas tienen un aspecto similar al representado en esta figura.



En todas las neuronas se identifican 3 partes principales:

- El cuerpo celular, donde se encuentra el núcleo; es el centro de síntesis de la célula, procesa las señales que le llegan de otras células, en forma de impulsos, generando un nuevo impulso si se cumplen ciertas condiciones.
- Las dendritas: ramas fibrosas que emanan del cuerpo celular.
- El axón: fibra principal que emana del cuerpo celular. Es el canal transmisor de los impulsos generados por la célula. Se ramifica en su extremo final para conectar con otras neuronas, a través de las dendritas de éstas, que actúan como canales receptores de información.

La conexión entre neuronas se realiza por medio de uniones especiales denominadas sinapsis. La transmisión de un lado a otro de estas uniones es de naturaleza química. La cantidad de señal transferida depende de la cantidad de química aportada por el axón y recibida por las dendritas. La intensidad sináptica es la que resulta modificada cuando decimos que el cerebro aprende. Las sinapsis, combinadas con el proceso de información de la neurona, forman el mecanismo básico de la memoria.

Introducción a las redes neuronales artificiales

Una Red Neuronal Artificial (RNA) se puede definir como un dispositivo diseñado a imitación de los sistemas nerviosos de los humanos, consistente en una interconexión de unidades, denominadas neuronas artificiales o elementos de proceso, cuyo funcionamiento se inspira en el de las neuronas biológicas.

Simplificando, podemos decir que la función básica de una neurona es sumar sus entradas y producir una salida si la suma es mayor que un umbral determinado (a través de la función de activación). Cada elemento de proceso tiene un conjunto de entradas y una sola salida por las que circulan las señales. Tanto las entradas como la salida dependen del instante de tiempo considerado.

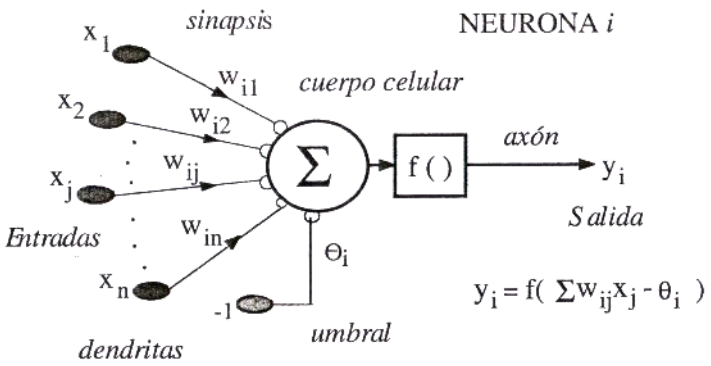
Las variables de entrada y de salida de una neurona pueden ser discretas o continuas, dependiendo del modelo de neurona considerada y de la aplicación que se le vaya a dar. Cuando las salidas pueden tomar valores continuos, se suelen limitar a un intervalo definido, como [-1,+1] o [0,1].

Las entradas a una neurona pueden ser las salidas de otras neuronas conectadas a ella. Así mismo, su salida puede ser una entrada a otros elementos de proceso (neuronas).

Cada conexión de entrada tiene asociado un número, denominado peso o fuerza de conexión, que determina el efecto cuantitativo de unas unidades sobre otras y corresponde a las sinapsis biológicas. Los pesos se suelen representar con una "w" y 2 subíndices que indican la neurona y la entrada a dicha neurona a la que están asociados, respectivamente. Por tanto, cada neurona tendrá tantos pesos como entradas.

La entrada total a un elemento de proceso, o entrada neta, y_i, se determina aplicando una determinada función, se basa en la suma ponderada de las entradas por los pesos, donde i representa el elemento de proceso cuya entrada neta se calcula, n (la parte superior del sumatorio) es el número de entradas de dicho elemento de proceso, las entradas se representan con una x y los pesos con w.

En esta figura podemos ver representada una neurona artificial:



Se pueden distinguir dos fases en la operación de la red: la fase de aprendizaje y la fase de recuerdo o ejecución. En la primera, la red aprende a resolver el problema para el que se ha diseñado. Para ello, los pesos de las conexiones se irán actualizando, permitiendo así aprender a la red. En la segunda fase los pesos permanecen fijos, se presentarán entradas a la red y ésta dará salidas.

Existen dos capas típicas en toda red: la capa de entrada y la capa de salida. La capa de entrada generalmente sirve para distribuir las entradas de la red, por lo que no se tiene en cuenta a la hora de contabilizar el número de capas de ésta. El resto de capas existentes entre ambas (entrada y salida) se denominan capas ocultas. Pueden existir varios tipos de conexiones entre neuronas:

- Conexiones intracapa: entre las neuronas de una misma capa.
- Conexiones intercapa: entre neuronas de diferentes capas.
- Conexiones realimentadas: van en sentido contrario a entrada-salida.

Con todo un conjunto de neuronas y conexiones formaríamos una red, creada al conectar unos nodos con otros y consiguiendo una arquitectura escrita en forma matemática. Bueno, y podríamos profundizar más, pero con esto ya es suficiente para poder entender el análisis que vamos a hacer a nuestra base de datos.

La estadística y las RNA, dos campos paralelos

Es innegable que existe una estrecha relación entre los campos de la estadística y de las redes neuronales artificiales.

A la hora de relacionar ambos campos se pueden considerar dos perspectivas. Por una parte, el campo de las RNA constituye una nueva herramienta para realizar cálculos de tipo estadístico. Por otra parte, numerosas de las técnicas y cálculos subyacentes al aprendizaje y operación de diversos modelos de RNA siguen métodos estadísticos bien conocidos.

Mientras la estadística se dedica al análisis de datos, en las redes neuronales la inferencia estadística significa aprender a generalizar a partir de datos con ruido.

A veces se afirma que las RNA, a diferencia de los modelos estadísticos, no requieren suposiciones sobre las distribuciones de los datos. Sin embargo, las RNA implican exactamente la misma clase de hipótesis distribucionales que los modelos estadísticos. La diferencia reside en que los estadísticos estudian la importancia y consecuencias de estas suposiciones, mientras que en el campo de las RNA muchas veces se ignoran. Una RNA puede operar sin hacer ese tipo de hipótesis, pero podremos conseguir una red mucho mejor si las consideramos.

Existen redes neuronales que permiten implementar técnicas de uso habitual en el campo estadístico, como el análisis cluster, el análisis discriminante y modelos de regresión.

Aunque para resolver algunos problemas existen métodos estadísticos que ofrecen muy buenos resultados, también se pueden aplicar modelos RNA. Estos modelos pueden suponer una reducción en el tiempo de cálculo, no tener que hacer suposiciones sobre las distribuciones de los datos, no tener que determinar expresiones matemáticas que definan el problema a resolver o, simplemente, ser un método de solución alternativo a las técnicas estadísticas disponibles.

DESARROLLO

Vamos a seguir las instrucciones del script, para modelizar con el programa R, una red neuronal sobre nuestra base de datos que explique la variable respuesta MonthGive.

Conocimientos requeridos

Lo primero que haremos será abrir el programa R-project, instalaremos y cargaremos las librerías BCA, para seleccionar nuestra base de datos CCS en el conjunto de datos activo, la librería neuralnet, necesaria para todas las instrucciones que vamos a usar y la librería Rcmd.Plugin.BCA, necesaria para seccionar la muestra para evitar el problema de overfitting.

El paquete neuralnet permite hacer configuraciones flexibles a través de la opción del error y la función de activación. Por otra parte, también se llevarán a cabo el cálculo de los pesos generalizados. Utilizaremos una serie de funciones como:

- plot.nn para el trazado de la red neuronal
- gwplot para el trazado de los pesos generalizados
- compute para el cálculo de la red ya entrenada
- confidence.interval para el cálculo de un intervalo de confianza para los pesos
- prediction para el cálculo de las predicciones

El **compute**, calcula las salidas de todas las neuronas para los vectores de covarianza arbitrarias específicas dadas una red neuronal entrenada. Pero como nosotros aún no hemos creado la red neuronal (neural network), y no ha pasado por la fase de aprendizaje, no puede estar entrenada. Así que primero nos dedicaremos a la formación de la red neuronal.

Para ello, utilizaremos la función **neuralnet**. Para crear un buen modelo, es necesario conocer bien nuestros datos. Se trata de una base de datos con 20 variables explicativas y 1600 datos. Todas las variables explicativas son características de personas que están en un registro de donaciones caritativas a una asociación canadiense. Y la variable respuesta es MonthGive, binaria, Sí o No. Lo que queremos crear un modelo que describa nuestro conjunto de datos en función de las otras variables, es decir, que en función de las características de cada persona nos diga si ésta hará una donación el próximo mes o no.

Pasos a seguir

Vamos a intentar explicar la variable MonthGive, a partir de las variables explicativas que nos parezcan interesantes, como por ejemplo: DonPerYear, Log.AveDonAmt, Log.LastDonAmt y NewRegionC.

Primero deberemos recodificar la variable categórica Region para simplificar:

```
##Variables R1 y R3###

CCS$NewRegion <- Recode(CCS$Region, '"R1"="A"; "R2"="A"; else="B"', as.factor.result=TRUE)

table(CCS$NewRegion)
```

A continuación crearemos logaritmos de las variables AveDonAmt y LastDonAmt porque tienen unos datos muy disperses y con éstas nuevas podremos trabajar mejor.

```
######Creo logaritmos#####

CCS$Log.AveDonAmt<- with(CCS,log(AveDonAmt))

CCS$Log.LastDonAmt<- with(CCS,log(LastDonAmt))
```

Antes de continuar dividiremos nuestra muestra en 3 partes, porque tiene muchos datos (1600), y así evitamos el problema de sobreajuste. Así podemos hacer con una muestra la fase de aprendizaje del modelo, y con otra la fase de ejecución.

```
CCS$Sample <- create.samples(CCS, est = 0.34, val = 0.33, rand.seed = 1)
```

El modelo al que llamaremos CCS.RNA, debe incluir algunos retoques; por ejemplo, a la variable explicada y a la variable NewRegionC le añadiremos as.numeric antes de llamarlas, porque son variables categóricas (factores), y hacemos esto para que no cree incoherencias y trabaje sólo con variables numéricas:

```
CCS$dependiente<-(as.numeric(CCS$MonthGive)-1)

CCS$NewRegionC<-as.numeric(CCS$NewRegion)
```

Lo siguiente que haremos sera indicarle dónde comienza mi semilla:

```
set.seed(467)
```

Creación del modelo RNA

Ahora sí, vamos a crear el modelo:

```
CCS.RNA<-neuralnet(dependiente~ DonPerYear + Log.AveDonAmt + Log.LastDonAmt+NewRegionC, data=CCS[CCS$Sample=="Estimation",],hidden=4, rep=9,act.fct="logistic", err.fct="ce", linear.output=FALSE)
```

Y este será el resultado:

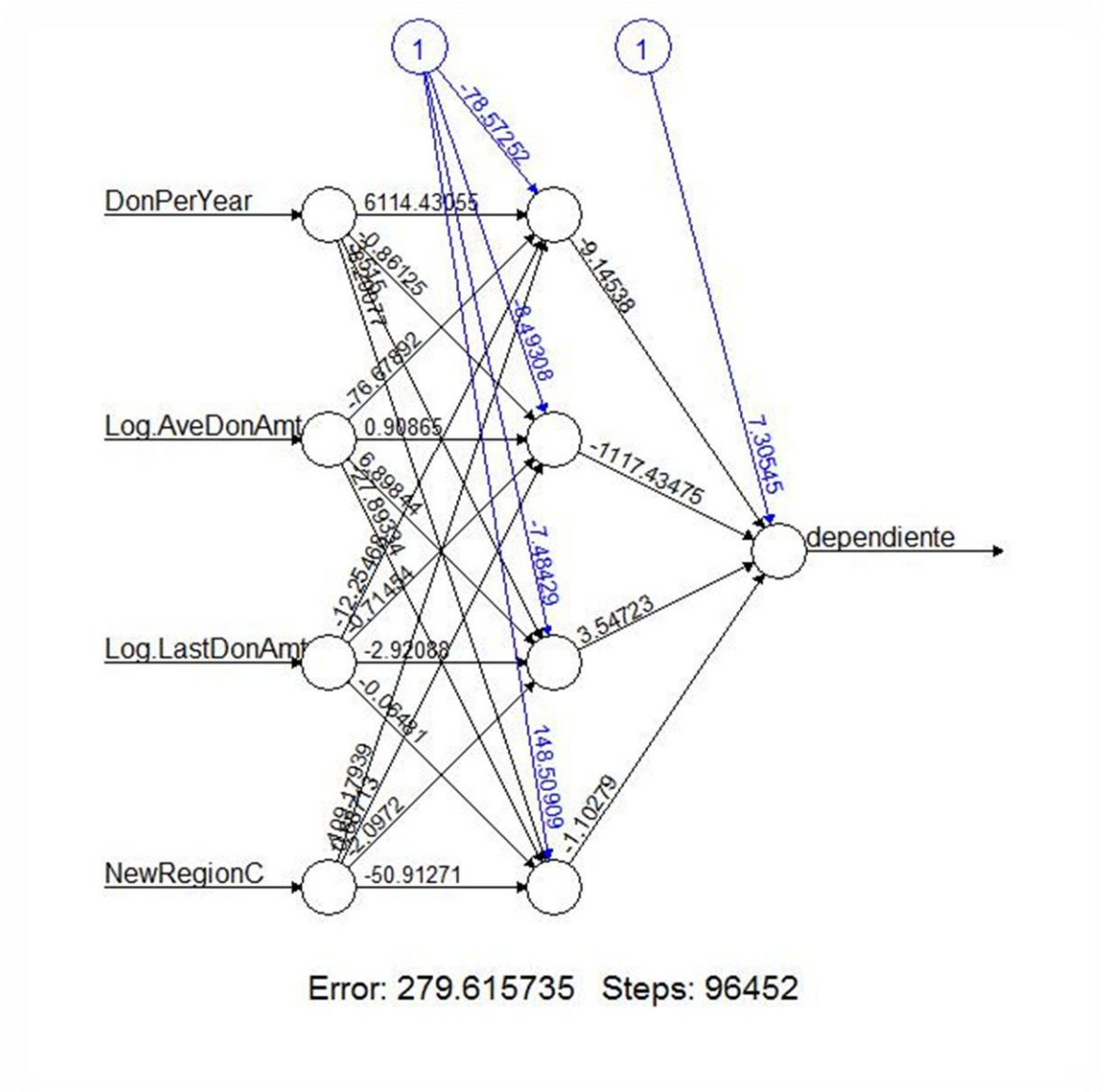
```
> CCS.RNA<-neuralnet(dependiente~ DonPerYear +Log.AveDonAmt  +Log.LastDonAmt+Ne$
Aviso: algorithm did not converge in 3 of 9 repetition(s) within the stepmax
> CCS.RNA
Call: neuralnet(formula = dependiente ~ DonPerYear + Log.AveDonAmt +     Log.La$

6 repetitions were calculated.

        Error Reached Threshold Steps
2 279.6157348    0.007801167759 96452
4 289.5715417    0.009764548848 33238
6 291.3668852    0.009790091582 41274
5 294.2696122    0.009894962387 27334
3 296.0318280    0.009810336225 82535
1 298.4271858    0.009944581696 22258
```

Hemos cambiado la instrucción en 6 repeticiones en vez de 9 porque no convergía nuestra red neuronal. En esta última tabla nos muestra las 6 repeticiones, las cuales están ordenadas según el error cometido por la red neuronal en cada entrenamiento (de menor a mayor), la segunda columna es el umbral alcanzado en la función de activación y la tercera columna son los escalones (vueltas) que necesita la red para alcanzar los pesos óptimos. La dibujamos y este es el resultado:

```
plot(CCS.RNA)
```



Escogemos la primera fila de la tabla porque es la que tiene el error más bajo, y este es el gráfico que le corresponde. Como podemos ver tiene 4 entradas, las 4 variables escogidas en el modelo y una única salida, dependiente.

Además existe 4 capas ocultas, puesto que se lo hemos indicado en la instrucción: hidden=4.

A continuación podríamos a mirar el resultado de la red desde otros puntos de información:

```
names(CCS.RNA)

CCS.RNA$result.matrix
```

```
>
> CCS.RNA$result.matrix

                                                    2
error                                   279.615734768146
reached.threshold                         0.007801167759
steps                                   96452.000000000000
Intercept.to.1layhid1                     -78.572520709043
DonPerYear.to.1layhid1                   6114.430551078977
Log.AveDonAmt.to.1layhid1                 -76.678916442349
Log.LastDonAmt.to.1layhid1                -12.254683476475
NewRegionC.to.1layhid1                   -109.179393061445
Intercept.to.1layhid2                      -8.493078863810
DonPerYear.to.1layhid2                     -0.861251636846
Log.AveDonAmt.to.1layhid2                   0.908654333662
Log.LastDonAmt.to.1layhid2                 -0.714543573794
NewRegionC.to.1layhid2                      0.687132647479
Intercept.to.1layhid3                      -7.484289877424
DonPerYear.to.1layhid3                      3.514999819425
Log.AveDonAmt.to.1layhid3                   6.898437662192
Log.LastDonAmt.to.1layhid3                 -2.920881952834
NewRegionC.to.1layhid3                     -2.097196471984
Intercept.to.1layhid4                     148.509093346051
DonPerYear.to.1layhid4                     -8.290770192666
Log.AveDonAmt.to.1layhid4                 -27.893343744859
Log.LastDonAmt.to.1layhid4                 -0.064812297142
NewRegionC.to.1layhid4                    -50.912711330715
Intercept.to.dependiente                    7.305446699099
1layhid.1.to.dependiente                   -9.145377117659
1layhid.2.to.dependiente                -1117.434746708098
1layhid.3.to.dependiente                    3.547229674063
1layhid.4.to.dependiente                   -1.102786405454
```

Esta matriz nos muestra los coeficientes que viajan en dirección entrada-salida.

```
CCS.RNA$weights

CCS.RNA$err.fct

CCS.RNA$act.fct

CCS.RNA$call

CCS.RNA$response

CCS.RNA$covariate

CCS.RNA$net.result

CCS.RNA$generalized.weights

CCS.RNA$startweights
```

Esto es lo que quiere decir cada apartado:

[1] Call: la instrucción de llamada del modelo.

[2] Response: extracción del valor de la variable respuesta.

Nos da valores 1 o 2 (significa que nos devuelve un Sí o un No)

[3] Covariate: el valor de las variables extraídas de la muestra utilizada.

[4] Model.list: una lista que contiene el nombre de la variable respuesta (response) y las variables explicativas (covariates).

[5] Err.fct: la función del error.

[6] Act.fct: la función de activación.

[7] Linear.output: nos dice si la salida es lineal.

[8] Data: nos muestra la base de datos.

[9] Net.result: una lista que contiene el resultado global de la red neuronal para cada repetición.

El valor de la primera repetición para todos los datos es 1.485297798

[10] Weights: una lista que contiene los pesos ajustados de la red neuronal para cada repetición.

[11] Startweights: una lista con los pesos iniciales de la red neuronal para cada repetición.

[12] Generalized.weights: una lista que contiene los pesos generalizados de la red neuronal para cada repetición.

[13] Result.matrix: una matriz que contiene el umbral alcanzado, las medidas necesarias, el error, el AIC y el BIC, y los pesos de cada repetición. Cada columna representa una repetición.

A continuación vamos a calcular la variable respuesta según nuestra red. Y veremos las cuál es la predicción con los datos originales.

```
calcula<-
compute(CCS.RNA,CCS.RNA$covariate,rep=4)$net.result
```

```
compuOut<-cbind(
CCS$dependiente[CCS$Sample=="Estimation"], calcula)

colnames(compuOut)<-c("MonthGive", "Predicción")

compuOut[ 30:50,]

compuOut[ 230:250,]

compuOut[ 430:450,]
```

```
> compuOut[ 30:50,]
      MonthGive   Predicción
 [1,]         1 0.6268885304
 [2,]         1 0.6169349660
 [3,]         1 0.6023456034
 [4,]         1 0.3724460373
 [5,]         1 0.6988030973
 [6,]         1 0.8974342067
 [7,]         1 0.6278093668
 [8,]         1 0.8983046692
 [9,]         1 0.7213928998
[10,]         1 0.6241838503
[11,]         1 0.6275640157
[12,]         1 0.5160215357
[13,]         1 0.6230814707
[14,]         1 0.5493904691
[15,]         1 0.1631933341
[16,]         1 0.6391956397
[17,]         1 0.1631933341
[18,]         1 0.8853148893
[19,]         1 0.5493904691
[20,]         1 0.6049145655
[21,]         1 0.2533560361
```

```
> compuOut[ 230:250,]
      MonthGive    Predicción
 [1,]         0 0.60954069056
 [2,]         0 0.62680767600
 [3,]         0 0.55610249850
 [4,]         0 0.63300943002
 [5,]         0 0.15198539888
 [6,]         0 0.07004948063
 [7,]         0 0.87628545601
 [8,]         0 0.44565545925
 [9,]         0 0.44565545925
[10,]         0 0.21896128956
[11,]         0 0.49929181453
[12,]         0 0.16523811426
[13,]         0 0.07148474223
[14,]         0 0.61884135817
[15,]         0 0.63264083714
[16,]         0 0.22094799512
[17,]         0 0.07004948063
[18,]         0 0.02803274440
[19,]         0 0.03062682539
[20,]         0 0.62678335406
[21,]         0 0.48350955070
```

```
> compuOut[ 430:450,]
      MonthGive      Predicción
 [1,]         0 0.2084915479459
 [2,]         0 0.0690218480331
 [3,]         0 0.1343287400548
 [4,]         0 0.6040808645113
 [5,]         0 0.5907733655322
 [6,]         0 0.6059493239771
 [7,]         0 0.0008430576929
 [8,]         0 0.4940458921369
 [9,]         0 0.4520570921354
[10,]         0 0.5922648720324
[11,]         0 0.6329738780627
[12,]         0 0.5939251079890
[13,]         0 0.6128844331650
[14,]         0 0.5235231020359
[15,]         0 0.1767127581636
[16,]         0 0.5493904690995
[17,]         0 0.2201205771378
[18,]         0 0.0039091601458
[19,]         0 0.1675367326100
[20,]         0 0.4456554592451
[21,]         0 0.5493904690995
```

Estas salidas son las predicciones, que podrían variar entre 0 y 1, 0 sería No y 1 sería Sí. Si el modelo fuese bueno los datos serían muy próximos a cero o muy próximos a 1, alejandose del 0.5, esto explicaría mejor si la persona hará la donación el próximo mes si o no.

Los gráficos gwplot representan los pesos generalizados (generalized weights) de una variable explicativa frente a la variable explicada. Instrucción:

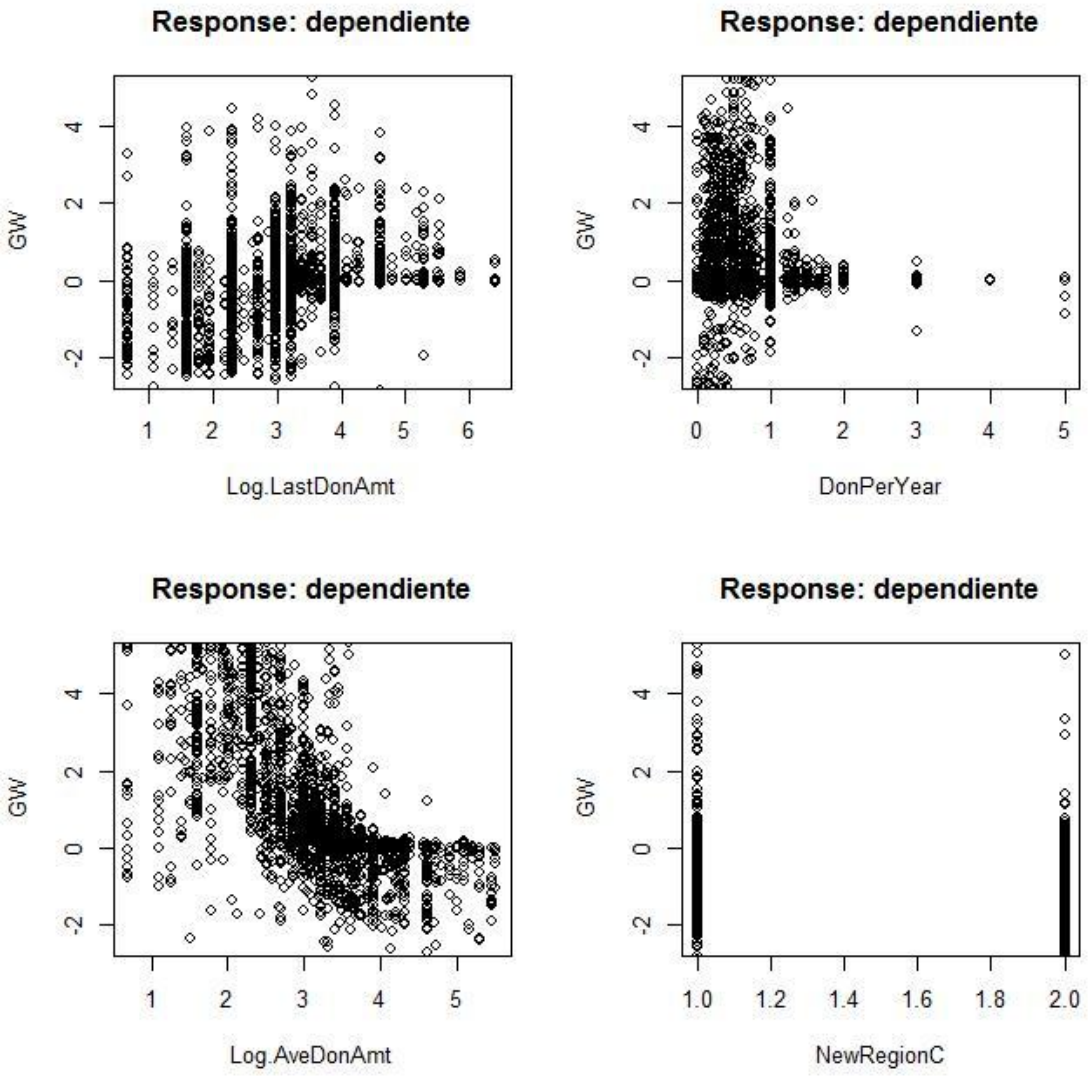
```
par(mfrow=c(2,2))

gwplot(CCS.RNA,selected.covariate="Log.LastDonAmt",min=-
2.5, max=5)

gwplot(CCS.RNA,selected.covariate="DonPerYear",min=-2.5,
max=5)

gwplot(CCS.RNA,selected.covariate="Log.AveDonAmt",min=-2.5,
max=5)

gwplot(CCS.RNA,selected.covariate="NewRegionC",min=-2.5,
max=5)
```



Como la red neuronal que hemos creado tiene solamente 4 variables, los gráficos estan dispersos. Necesitaríamos implementar más variables en el modelo para que los datos se ajustasen más.

Predicción

Lo último que vamos a hacer será la predicción, para ello ya hemos pasado la RNA por la fase de entrenamiento, y ahora será la fase de ejecución.

```
salPred<-prediction(CCS.RNA)

names(salPred)
```

```
rep2<-cbind (CCS$MonthGive[CCS$Sample=="Estimation"]
,round(salPred[[2]][,"dependiente"],2))
colnames(rep2)<-c("Real","Pred")
rep2
round(salPred[[2]][,"dependiente"],0)
```

Dato	Valor Predicho	Correspondencia
1	0.03	0
2	0.26	0
3	0.04	0
4	0.04	0
5	0.04	0
6	0.05	0
7	0.04	0
8	0.04	0
9	..	…
	..	
540	0.83	1
541	0.84	1
542	0.77	1
543	1.00	1
544	0.6	1

Las que tienen correspondencia 0, no harán la donación el próximo mes, y los que tienen correspondencia 1, si que la harán. Si queremos ver el resultado de las otras repeticiones que tenían un error más alto, utilizaríamos estas instrucciones.

```
round(salPred[[4]][,"dependiente"],0)
round(salPred[[6]][,"dependiente"],0)
round(salPred[[5]][,"dependiente"],0)
round(salPred[[3]][,"dependiente"],0)
round(salPred[[1]][,"dependiente"],0)
```

CONCLUSIÓN

Tras conocer la teoría de las redes neuronales artificiales como modelos estadísticos explicativos, hemos diseñado un modelo neural network a través de unos sencillos pasos.

Así hemos podido explicar si una persona de un registro de donaciones caritativas hará una donación el próximo mes o no, con tan solo 4 variables. Cabe decir que si hubiésemos tenido más variables, tal vez las predicciones hubieran salido más ajustadas.