# Artificial Neural Networks 4

Roger Barlow
The University of Huddersfield

CODATA School on Data Science

Autumn 2021

University of
HUDDERSFIELD
International Institute
for Accelerator Applications

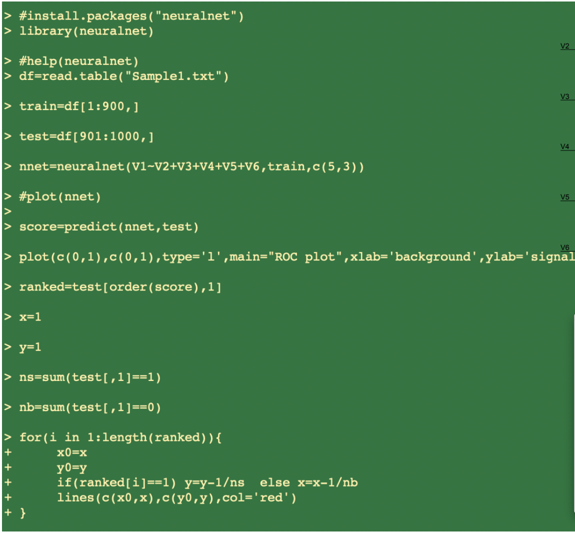
# A professional network program

The R package `neuralnet`, Written by Fritzsch and Günther

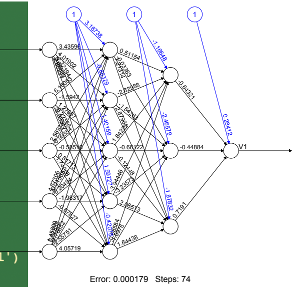
- `install.package('neuralnet')`. You just do this once
- `library(neuralnet)`. Do this once per session
- Prepare a data frame and partition into training and testing samples
- Compose a model using the data frame column names, e.g. `V1` $\sim$ `V2+V3+V4+V5+V6`
- Decide the hidden layer structure, e.g. 2 layers of 5 and 3 nodes
- Create and train the network using `nnet=neuralnet(model,trainingsample,c(5,3))`
- (Optional) call `plot(nnet)` to see the weights - impressive but not really useful
- Evaluate using `score=predict(nnet,testingsample)`

For more information type `help(neuralnet)` or see `https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf`

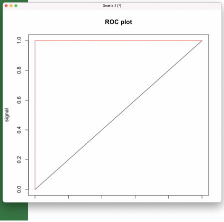
# Look at data on Sample1.txt



```
> #install.packages("neuralnet")
> library(neuralnet)

> #help(neuralnet)
> df=read.table("Sample1.txt")

> train=df[1:900,]

> test=df[901:1000,]

> nnet=neuralnet(V1~V2+V3+V4+V5+V6,train,c(5,3))

> #plot(nnet)
>
> score=predict(nnet,test)

> plot(c(0,1),c(0,1),type='l',main="ROC plot",xlab='background',ylab='signal')

> ranked=test[order(score),1]

> x=1

> y=1

> ns=sum(test[,1]==1)

> nb=sum(test[,1]==0)

> for(i in 1:length(ranked)){
+       x0=x
+       y0=y
+       if(ranked[i]==1) y=y-1/ns  else x=x-1/nb
+       lines(c(x0,x),c(y0,y),col='red')
+ }
```

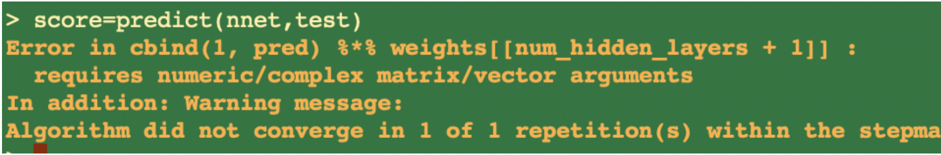


Error: 0.000179   Steps: 74



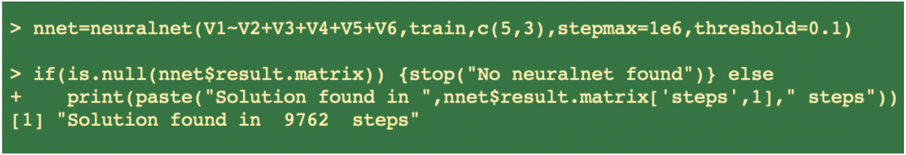
# But it's not always that easy

When that works, try the same program but using Sample2.txt



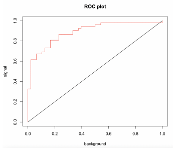
```
> score=predict(nnet,test)
Error in cbind(1, pred) %*% weights[[num_hidden_layers + 1]] :
  requires numeric/complex matrix/vector arguments
In addition: Warning message:
Algorithm did not converge in 1 of 1 repetition(s) within the stepma
```

`neuralnet` call has failed - and only given a warning. Need to catch it. And either increase the number of steps or lessen the threshold required for solution - or both



```
> nnet=neuralnet(V1~V2+V3+V4+V5+V6,train,c(5,3),stepmax=1e6,threshold=0.1)

> if(is.null(nnet$result.matrix)) {stop("No neuralnet found")} else
+    print(paste("Solution found in ",nnet$result.matrix['steps',1]," steps"))
[1] "Solution found in  9762  steps"
```

That gives a satisfactory ROC plot



# neuralnet Parameters

- `threshold` used in convergence (based on magnitudes of differentials) . Default 0.01
- `stepmax` number of steps allowed before failure. Default 1E5
- `rep` Repetitions. NOT training cycles. Will build separate networks `rep` times. Useful to check your result is unique.
- `linearoutput` needs to be set FALSE if you want the smoothing function. Which you probably do. So over-ride the default here. If applied, logistic function is default but others are available.
- `algorithm` default is `rprop+`, 'resilient' back propagation with weight backtracking. Others are available including standard `backprop` in which case `learningrate` needs to be set

You can give it starting weights - I have yet to find this useful. And several other parameters you can tweak.

# Some real data

Download the file called `Sample4.txt`. Originally called `ticdata2000.txt` and made public by Peter van der Putten of Sentient Machine Research.

*Please quote this reference to refer to the TIC Benchmark / CoiL Challenge 2000 data: P. van der Putten and M. van Someren (eds) . CoIL Challenge 2000: The Insurance Company Case. Published by Sentient Machine Research, Amsterdam. Also a Leiden Institute of Advanced Computer Science Technical Report 2000-09. June 22, 2000. See http://www.liacs.nl/∼putten/library/cc2000/*

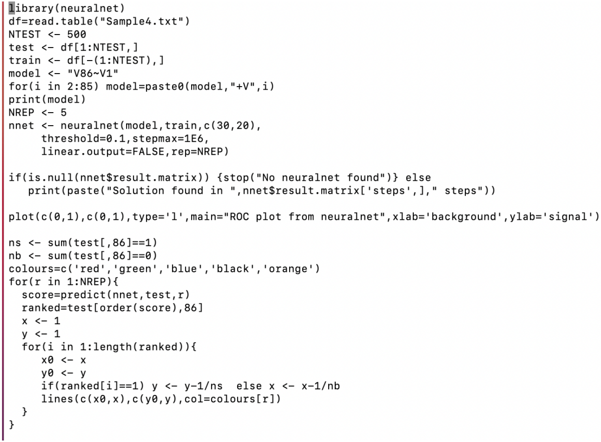
Data table has 5822 rows (each row is 1 person) and 86 columns.

Columns give lots of numeric (integer) data on people: education, income, religion, family circumstances... and insurance policies. For full details see `https://liacs.leidenuniv.nl/∼puttenpwhvander/library/cc2000/`

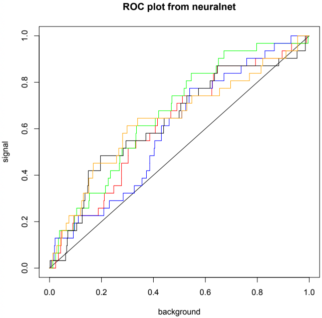
Column 86 tells whether they have insured a caravan or not.

The challenge is to predict Column 86 from the other 85 columns.

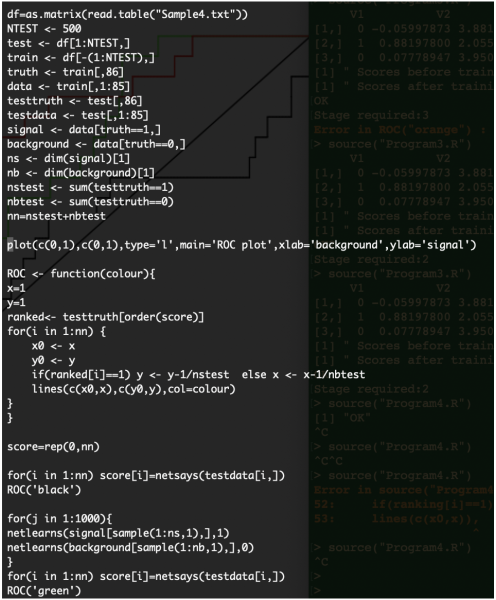
# Using neuralnet



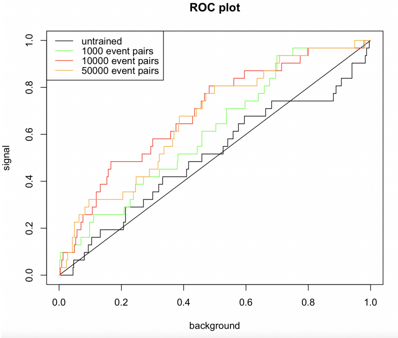
```
library(neuralnet)
df=read.table("Sample4.txt")
NTEST <- 500
test <- df[1:NTEST,]
train <- df[-(1:NTEST),]
model <- "V86~V1"
for(i in 2:85) model=paste0(model,"+V",i)
print(model)
NREP <- 5
nnet <- neuralnet(model,train,c(30,20),
     threshold=0.1,stepmax=1E6,
     linear.output=FALSE,rep=NREP)

if(is.null(nnet$result.matrix)) {stop("No neuralnet found")} else
   print(paste("Solution found in ",nnet$result.matrix['steps',]," steps"))

plot(c(0,1),c(0,1),type='l',main="ROC plot from neuralnet",xlab='background',ylab='signal')

ns <- sum(test[,86]==1)
nb <- sum(test[,86]==0)
colours=c('red','green','blue','black','orange')
for(r in 1:NREP){
  score=predict(nnet,test,r)
  ranked=test[order(score),86]
  x <- 1
  y <- 1
  for(i in 1:length(ranked)){
     x0 <- x
     y0 <- y
     if(ranked[i]==1) y <- y-1/ns  else x <- x-1/nb
     lines(c(x0,x),c(y0,y),col=colours[r])
  }
}
```



# Using our home-made package



```
df=as.matrix(read.table("Sample4.txt"))
NTEST <- 500
test <- df[1:NTEST,]
train <- df[-(1:NTEST),]
truth <- train[,86]
data <- train[,1:85]
testtruth <- test[,86]
testdata <- test[,1:85]
signal <- data[truth==1,]
background <- data[truth==0,]
ns <- dim(signal)[1]
nb <- dim(background)[1]
nstest <- sum(testtruth==1)
nbtest <- sum(testtruth==0)
nn=nstest+nbtest

plot(c(0,1),c(0,1),type='l',main='ROC plot',xlab='background',ylab='signal')

ROC <- function(colour){
x=1
y=1
ranked<- testtruth[order(score)]
for(i in 1:nn) {
    x0 <- x
    y0 <- y
    if(ranked[i]==1) y <- y-1/nstest  else x <- x-1/nbtest
    lines(c(x0,x),c(y0,y),col=colour)
}
}

score=rep(0,nn)

for(i in 1:nn) score[i]=netsays(testdata[i,])
ROC('black')

for(j in 1:1000){
netlearns(signal[sample(1:ns,1),],1)
netlearns(background[sample(1:nb,1),],0)
}
for(i in 1:nn) score[i]=netsays(testdata[i,])
ROC('green')
```



The training data only has ∼16% signal, with ∼86% background. Training must use a 50:50 mix So training does one random signal and one random background event, many times. Selected using the `sample` function.

# Conclusion

The `neuralnet` package is available and well documented, but needs some care in handling

Real (or realistic) data can be challenging

### Activity for you

Either: see what you can do with Sample4.txt, using `neuralnet` or your own network program

Or: Apply a neural network to a classification problem in your own data

Present some result(s) as a plot and upload it for a class zoom meeting