CODATA – RDA

Data Schools

Artificial Neural Networks
1 : Summary of the course

Roger Barlow Roger.Barlow@hud.ac.uk

What you learnt

1. We are very good at pattern recognition
2. Pattern recognition is used in all branches of science
3. We can emulate the brain in software as a neural network
4. Running the network is easy. Training the network is hard
5. Performance is measured by ROC plots
6. You wrote a program, either with its own network software or downloading it, and evaluated performance for various configurations
7. And you gave a talk about it

Pattern recognition

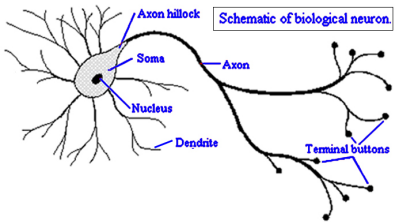
The human brain can recognise and distinguish cats from dogs

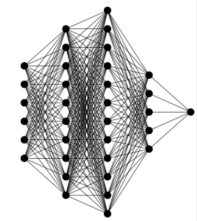
- Swiftly - in a fraction of a second
- Accurately - get it right pretty much every time
- Reliably - even when we're tired or cross or distracted
- Robustly - from partial or even misleading information

Every expert needs this ability

- Particle physicists: signal and background events
- Astronomers: starts and galaxies
- Doctors: sick and healthy patients
- Seismologists: natural earth tremors from human activity
- Data scientists: online robots from real people
- Military: friendly tanks from hostile tanks
- Financiers: good investments from bad investments
- *Your field*: *Your examples*

From neurons to networks





Networks have a few layers, several nodes per layer, and many weights

```
nodes=c(5,7,10,1)       # 5 inputs, 2 hidden layers, with 7 and 10 nodes , 1 output
nlayers=length(nodes) -1          # 3  sets of weights

net=list() # set up empty list
#  net[[ j ]] holds weight matrix feeding nodes of layer j+1 from nodes in layer j

#   make weights and fill with random numbers
for(j in 1:nlayers) net[[ j ]] <- matrix(runif(nodes[ j ]*nodes[ j +1 ]),nodes[j+1],nodes[j])

netsays <- function(x) { #  Returns net output for some input vector x
      for(j in 1:nlayers) x <- 1/(1+exp(-net[[ j ]] %*% x))
      return(x)
       }
```

Training!!!
'Supervised Learning'

You need samples of data where the species is known

Present these to the network (alternating or random sequence) and insist on a decision

Reward (increase weights) for correct answers and punish (decrease weights) for wrong answers, using the formulæ for back propagation

Repeat MANY times. Re-cycle training samples.

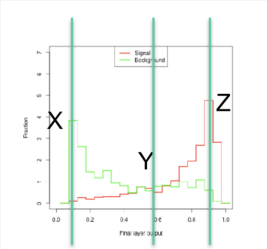
But do not over-train - separate samples for training and testing

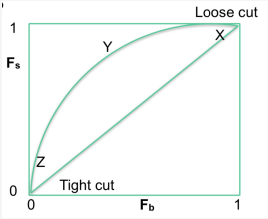
Performance - ROC plots

Take NN output (between 0 and 1)
Loose cut (X) - accept all signal and all background events
Tighter cut (Y) - get fewer signal and fewer background, but hopefully background suffers more
Very tight cut (Z) - get a very pure signal sample but with very low efficiency

Diagonal line - no power at all.
The further from the diagonal, the better

Choosing where to put your cut also needs
(i) relative numbers of S and B in real life
(ii) costs of Type I and Type II errors





You made it work

As part of a group you analysed data made of 5 numbers, either in a 1-2-3-2-1 or a 0-4-1-4-0 pattern using either

Your own ANN network program
If you did this, you learnt a lot about R

The Fritsch and Günther `neuralnet` package
If you did this, you got some pretty pictures of networks and theur trained weights

looking at 3 different cases: easy separation, moderate separation and hard separation.

Studying what parameters gave the best performance

And you put together a group presentation – pretty good for an afternoon's work, on a subject that you were (probably) unfamiliar with

Summary and future talks

1. You learnt what an ANN was and you set one up and used it
2. What's next? What's new? What's changed?
3. Tips on teaching your own course