1

Neural Networks

- Technique for discrimination & regression problems
- More mathematical theoretical foundation
- Works well on many practical problems
- Same problems when there are large number of variables and small numbers of observations

2

- **Mathematical description**
  - A '***neural network'*** is a particular type of non-linear regression model
  - Typically it has very many parameters
  - Sometimes even more parameters than observations
- **Aside:**
  - Usually in applied statistics more parameters than observation ⇒ **TROUBLE**
  - Can be trouble here as well
    - Important to check performance of neural networks with randomization, X-validation etc

3

- Note that the technique was developed by Computer Scientists and terminology may not appear to be standard statistical terms

  e.g.
  - 'inputs' ≡ data (independent variables)
  - 'targets' ≡ data (dependent variables)
  - 'outputs' ≡ predicted values
  - 'weights' ≡ unknown parameters
  - 'training a network' ≡ estimating unknown parameters

4

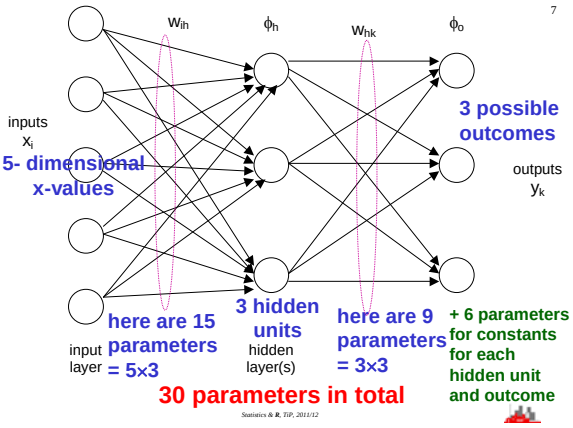
- Many of the methods developed for neural networks came from algorithmic ideas.
- Many also have general statistical justification
- Many are very successful
- Some problems remain
  - e.g. *over-parameterization*
    - (no free lunches even with neural nets)

5

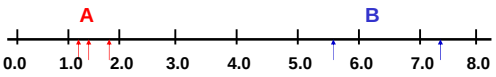
- Outline of steps
  - For each value of **x** there is a target **y**
    - e.g. x=(sepal.length, ……,petal width)
    - y = type "S" – coded as (1,0,0) or type "C" – coded as (0,1,0) or type "V" – coded as (0,0,1) say
  - Calculate several different linear functions $a_i(x)$ of x using weights (parameters) $w_{i1},w_{i2},w_{i3},w_{i4}$ for a suitable number of functions or '*units in a hidden layer*'
  - Use an *activation function* $\phi(.)$ on each $a_i(x)$ to get $\phi(a_i(x))$

6

- Take another linear function of the $\phi_i(a_i(x))$ using weights (or parameters) & feed through another *activation function* to get each element of output **y**
- i.e. three separate (activated) linear functions to get a value (1,0,0) or …. or (0,0,1)
- Now estimate parameters w_{ij} to make outputs match targets as closely as possible e.g. *least squares minimization*
- Evaluate rule and perhaps *tune network* by changing number of hidden units

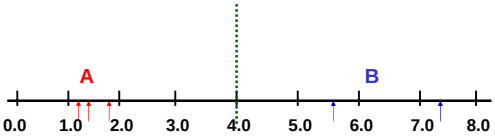


- **Simple example:** (see P142)
  - Univariate data; 2 categories A and B
  - 3 samples from A and 2 from B
  - A samples: 1.1, 1.7, 1.3
  - B samples: 5.6, 7.2



- Easy to discriminate between A & B with a simple rule such as

  If $x < 4.0$ then classify as A, else B



However, neural nets work differently…

- Aim is to find two formulas $f_1(x)$ and $f_2(x)$ such that

  $f_1(x) \approx 1$ when x is of type A
  and ≈ 0 when x is of type B

  $f_2(x) \approx 1$ when x is of type B
  and ≈ 0 when x is of type A

So want $f_1(x) \approx 1$ for x= 1.1, 1.3, 1.7
≈ 0 for x= 5.6, 7.2

Impossible with a linear function of x

- **Neural net solution** with one hidden layer of 2 units and logistic activation functions
  - Calculate, for i = 1 and 2
    $g_i(x) = \exp\{b_i + w_i x\}/[1+\exp\{b_i + w_i x\}]$
  - Calculate, for i = 1 and 2
    $f_i(x) = \exp\{c_i+w_{1i}g_1(x)+w_{2i}g_2(x)\} / [1 + \exp\{c_i+w_{1i}g_1(x)+w_{2i}g_2(x)\}]$
  - This has 10 unknown parameters
  - Estimate these by minimizing
    $(f_1(1.1) -1)^2+\ldots\ldots\ldots\ldots\ldots+(f_2(7.2) -1)^2$

- Note, 10 parameters but only 5 observations, though each observation used twice in minimization since we have terms $(f_1(1.1) -1)^2$ and $(f_2(1.1) -0)^2$

- Surprisingly this works
  - Also solution does not seem to be ***data dependent***
    - i.e. it does not seem to be very dependent on the particular data we use.

13

- Implementation in **R**:

```
> nick<-
+
  data.frame(x=c(1.1,1.7,1.3,5.6,7.2,8.1,1.8,3.0)
  ,
+ targets=c(rep("A",3),rep("B",2),rep("U",3)))
> attach(nick)
```

Sets up a data set with values for training samples for A and B and test samples for U (8.1, 1.8, 3.0)

```
> nick.net<-
  nnet(targets~.,data=nick[1:5,],size=2)
```

Calculates a neural net from first 5 data points (i.e. the *training data*)

14

```
> predict(nick.net,nick[1:5,])
              A            B
1 1.000000e+00 2.286416e-18
2 1.000000e+00 3.757951e-18
3 1.000000e+00 2.477393e-18
4 1.523690e-08 1.000000e+00
5 2.161339e-14 1.000000e+00
```

Uses the estimated functions and evaluates them for the training data:

Note that $f_1(x)=1$ & $f_2(x)=0$ for x= 1.1, 1.7 & 1.3 and $f_1(x)=0$ & $f_2(x)=1$ for x=5.6 and 7.2

15

```
> predict(nick.net,nick[1:5,],type="class
  ")
[1] "A" "A" "A" "B" "B"
```

Checks that the categories are correct using the type="class" option

```
> predict(nick.net,nick[6:8,])
              A            B
6 1.364219e-15 1.000000e+00
7 1.000000e+00 4.659797e-18
8 1.000000e+00 1.769726e-08
```

Looks at numerical predictions on test data U

16

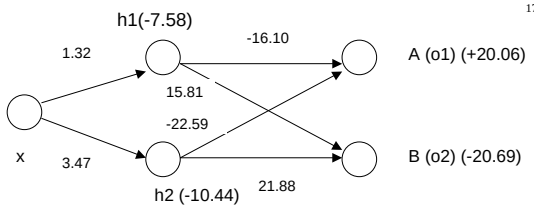
```
> predict(nick.net,nick[6:8,],type="class
  ")
[1] "B" "A" "A"
```

Gives classifications for new data.

```
> summary(nick.net)
a 1-2-2 network with 10 weights
options were - softmax modelling
 b->h1 i1->h1
 -7.58   1.32
 b->h2 i1->h2
-10.44   3.47
 b->o1 h1->o1 h2->o1
 20.06 -16.10 -22.59
 b->o2 h1->o2 h2->o2
h1(-7.58)-20.69  15.81  21.88
```

Estimates of parameters in the functions $g_i(.)$ and $f_i(.)$

17



Gives a pictorial representation of the two functions $f_1(.)$ and $f_2(.)$ with values of the estimates of the parameters

18

- **Notes**
  - Sophisticated numerical optimization in calculations in **R**
    - some parameters to `nnet(.)` control this
  - key ***statistical*** parameter in is **size**
    - controls number of units in hidden layer
    - i.e. number of parameters to be estimated
  - increasing **size** gives a better fit
    - does well on training data

  but may make the prediction less reliable on new data

19

- Notes (continued)
  - i.e. may be dangers of ***overfitting***
  - aim is to get a model with as few parameters as possible that still performs well
- **Strategy:**
  - fit model & investigate statistical properties
    - permutations
    - random sampling
    - applications to new data

20

- Lecture notes give some examples of this approach on iris data and data set *book*
  - NB 'book' is just a code name & the variable names are not available
    - (current research data set with a company)
- **NOTE:-**
  - **Output in notes has been edited**
  - **Not all of the R commands are given**
    - **should be sufficient to see the type of approach used**

21

- **Summary**
  - Very simple introduction to neural networks
    - Many other types of networks available
    - References by Ripley (1996) & Bishop (1995)
  - Technique for classification based on constructing numerical formula
    - Can also be applied to regression problems
      - (compare tree-based methods)
  - General statistical principles of high number of parameters and overfitting
    - Less serious than could be expected
    - But still a problem to consider

22

- Final Comments & Summary
  - Introduction to the language **R**
    - Provides facilities for learning new statistical techniques in all areas
    - All new statistical developments in the next few years will be available in **R**
  - Ideas of what to do if assumption of Normality is not sensible
    - Ideas of **robust methods**
    - Ideas of cross-validation
    - Ideas of random resampling and permutations
    - **Bootstrapping**

23

- Final Comments & Summary (ctd)
  - Regression methods
    - Interactive identification of points
    - Regression diagnostics
    - Robust regression
  - Multivariate methods
    - Based on traditional mathematical & statistical theory
    - Problems of discrimination
    - Methods for evaluating performance based on random resampling, permutation, etc

24

- Final Comments & Summary (ctd)
  - Other methods of classification etc
    - Based on algorithmic ideas
    - Tree-based methods
    - Neural networks
  - Evaluate statistical properties by statistical experiment
    - Random resampling, permutation etc etc.
- **Aristotle:** for the things we have to know before we can do them, we ***learn*** by doing them
- i.e. **TRY IT & SEE WHAT HAPPENS**