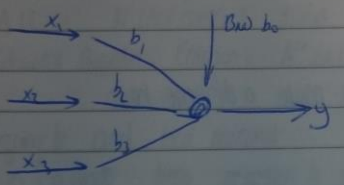
28/04/16

DA - NN

- Classification, prediction, clustering
- Map a series of inputs onto an output

- Feed forward multilayer perception
- Radial basis function networks
- Bayesian NN
- Kohonen self organising maps - NN for clustering

Multilayer perceptron - Feed forward



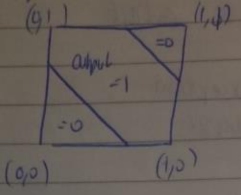
$= b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3$ - weighted summation

Transfer or activation function
- identity function $g(a) = a$
- linear function - $g(a) = \alpha + \beta^* a$ for some α, β
- Threshold function $g(a) = \begin{cases} 1 & a \geq 0 \\ 0 & a < 0 \end{cases}$
- Sigmoid function $g(a) = 1/1+e^{-a}$

Error = target value - predicted value
- Just a linear combination of the x's with lots of parameters
- With identity function - no different to linear regression

What makes them more powerful?
- non linear activation function
- Model can handle more complicated situations
- Turns out to be a non-linear model with lots of parameters - weights

- linear seperability - being able to split into 2 groups without many mixed members in group



Input data: - what variables - previous variables
- Scale of data - affects weights.
- Standardise data
- Missing data
- Transformation of data - outliers may make a difference here
- Interactions, may be important.

Calculation of Weights
- For a given NN
  Randomly initialise weights - should be done carefully
  Calculate output y
  Compare to target value t
  Define an error function e.g. $E = \sum_{\text{all obs}} \frac{1}{2}(y-t)^2$

Which weights are responsible?

$$\Delta w_{ji} = \frac{\delta E}{d w_{ji}}$$

~~A~~ $w_{ji}' = w_{ji} + \Delta w_{ji}$

- Run another record or (all records) through and update again
- Calculate E using training set
- Stop when E changes only a little

- Danger of finding local minima for weights

28/04/16

DA - NN

Hessian Matrix

- Matrix that contains $\frac{d^2E}{dw_i dw_j}$

- Eigen values should all > 0 for global maximum
- Can look at these for each solution

Evaluation

p = # parameters N = # cases

SSE = Sum of Squared Errors

AIC : $N \ln(SSE/N) + 2p$

Schwartz Bayesian Criterion: $N^* \ln(SSE/N) + p\ln(N)$

- Trying to avoid local minima
- Run a simple linear regression to get a baseline for error
- Look at eigenvalues of Hessian Matrix and weights
- Scale data beforehand - mean of 0 sd: 1
- Start with linear Regression and run a few times
- Use a 'For loop' to see variation in Error
- Cannot compare models with unscaled and scaled data

Classification Problems

- R uses logistic function as activation function
- K classes, K output variables y_k $\sum_{i=1}^{K} y_i = 1$
- Range y_n $[0,1]$
- Interpret the output as probability of belonging to class K
- Use Softmax transformation
- Softmax function to map Ts into $[0,1]$ range

$$y_i = \frac{e^{T_i}}{\sum_{j=1}^{k} e^{T_j}} \quad \text{ensures} \quad \sum_{i=1}^{K} y_i = 1$$

Interpret y_is as probabilities

- Objective function = Error function + penalty
- Penalty = $d * \sum w_i^2$
- keeps weights from flying off to infinity
- Values for d are usually in region of 0.001

Number of Hidden layers
- No hidden layers - linear separability

1 hidden layer with x nodes should approximate most functions

2 hidden layers will introduce more complexity at a cost sometimes

Number of Weights to estimate

n - inputs, m hidden nodes in first layer, p hidden nodes in second layer.

No hidden layers: $n+1$ weights

1 hidden layer: $m*n + m = m*(n+1)$ weights

2 hidden layers: $p*[m*(n+1)+1]$

Growing
- Start with no hidden nodes or layers.
- Put 1 hidden layer with one hidden node and look at AIC/BIC of training set
- At each stage use weights already determined as initial weights - can do this in R
- At some stage add another layer with 1,2,3 hidden nodes

Pruning
- Complicated network trained
- Which weights/paths can be deleted?
- What are the least important weights?
- little theoretical motivation
- Performs poorly in practice

28/04/16

DA - NN

EXAM QUESTIONS

→ Overview of NN

- NN are learning models that attempt to map a series of inputs to an output
- Typically comprised of an input layer, a number of hidden layers and output layer
- Input layer contains the input variables that are being pulled to [illegible]

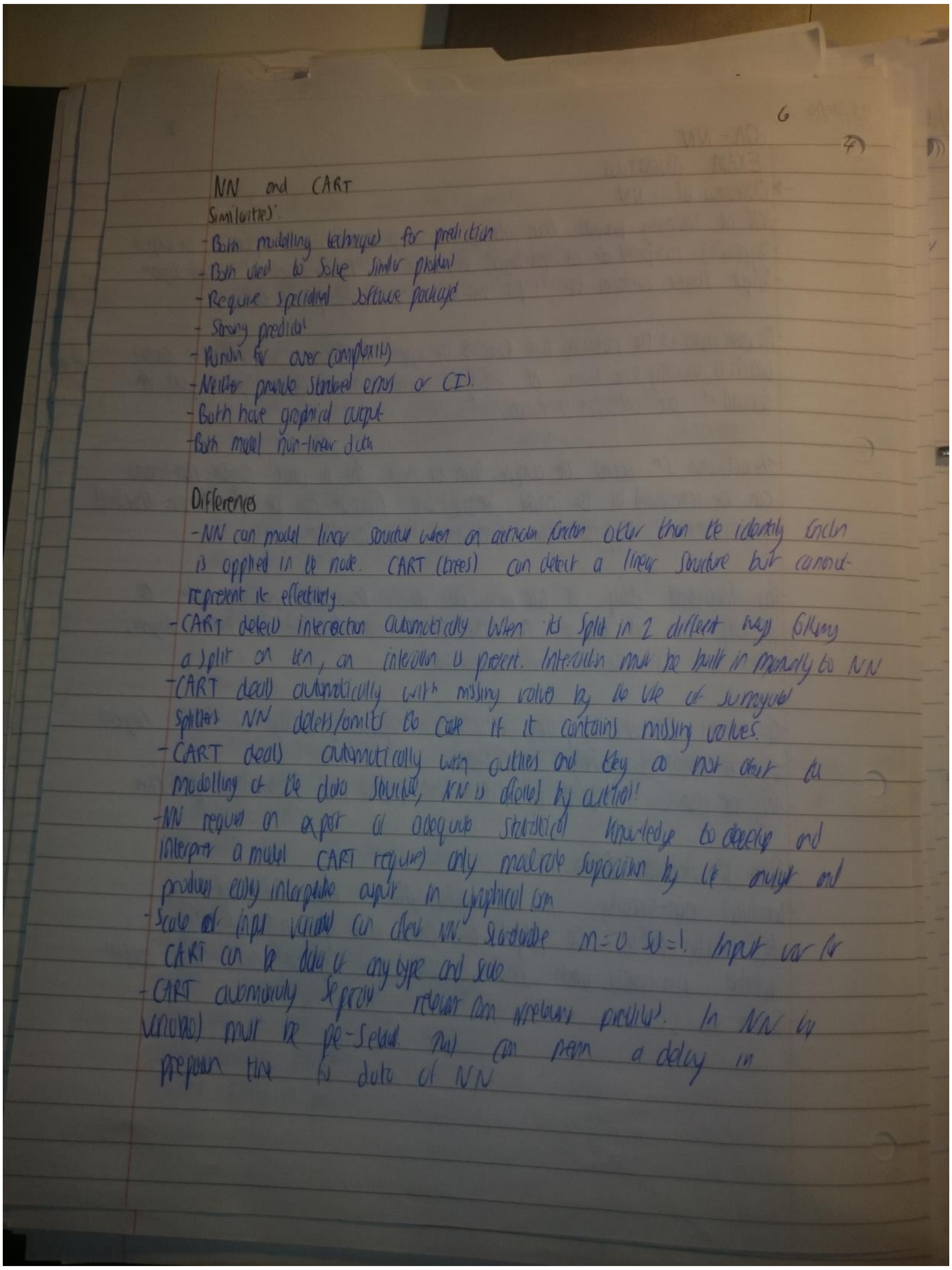
- At each nodes in the network, two functions are applied. The first is a combination function which is usually some form of weighted summation of the multiple inputs. The second is the activation function

- The activation f^n defines the output from the node. This is where complex non-linearity can be introduced to the model because this function can be non-linear. Threshold, piecewise, and hyperbolic-tan functions are all non-linear.

- This fundamental ability of NN make them dissimilar from linear regression. When the identity function is used as the activation function and there are no hidden layers, the NN is like a linear regression

- The nodes interact by pulling their values sequentially through the hidden layers via a series of weights and biases
- They are often considered a "black box" method of classifying or prediction

Role of Activation Function
- Introduces non-linearity into the modelling system at each hidden node.
- Act function defines the output of that node given a set of inputs.
- Introduces non-linearity which make NN more powerful than linear models such as LR.



NN and CART

Similarities:

- Both modelling techniques for prediction
- Both used to solve similar problems
- Require specialised software package
- Strong predictors
- Punish for over complexity
- Neither provide standard errors or CI's.
- Both have graphical output
- Both model non-linear data

Differences

- NN can model linear structure when an activation function other than the identity function is applied in the node. CART (trees) can detect a linear structure but cannot represent it effectively.
- CART detects interaction automatically when its split in 2 different ways following a split on ten, an interaction is present. Interaction must be built in manually to NN
- CART deals automatically with missing values by the use of surrogate splitters NN deletes/omits the case if it contains missing values.
- CART deals automatically with outliers and they do not affect the modelling of the data structure, NN is affected by outliers!
- NN requires an expert or adequate statistical knowledge to develop and interpret a model CART requires only moderate supervision by the analyst and produces easily interpretable output in graphical form
- Scale of input variables can affect NN. Standardise $m = 0$ $sd = 1$. Input var for CART can be data of any type and scale.
- CART automatically separates relevant from irrelevant predictors. In NN the variables must be pre-selected. This can mean a delay in preparation time for data of NN