R Markdown_Neural Networks (Intro) Part 1

Dominique Tanner

Background:

Neural Network Example 1: * input 2 * output 7

How to manipulate the input and output: * Take the weight w = 2 and bias b = 3: check w*Input + b = Output

Neural Network Example 2: * input1 2 * output1 7

- input2 3
- output2 21

How to manipulate the input and output: * Find a weight w and bias b, so that w*Input1 + b = Output1 and w*Input2 + b = Output2.

Training Data:

Inputs Outputs Predicted outputs I1 O1 P1 = w*I1 + b I2 O2 P2 = w*I2 + b … … … In On Pn = w*In + b

Note: Calculate the Mean Square Error of Prediction = MSE = (1/n)Σ (Oi – Pi)2. * The MSE will depend on the weight w and bias b chosen. You can prescribe what you want MSE to be. * If not satisfied with the MSE that was received, change the weight and bias. * Keep experimenting with weight w and bias b until you have gotten the MSE below the level you prescribed.

Neural network pursues the following route: * For abstraction, let x = 2 and y = 7.

Step 1: Choose weight w and bias b. Calculate u = w*x + b.

Step 2: Apply the activation (logistic) function f(u) = 1/1 + exp(-u) to what is calculated in Step 1 [i.e., x' = 1/ 1 + exp(-(w*x+b))] This is a non-linear transformation of the input x.

Step 3: Choose another weight w' and bias b'. Calculate y' = w'*x' + b'. Calculate y" = f(y'). This is a linear + non-linear transformation of the contrived output x'.

Step 4: Is y" close to the output y? Spell out how close you want the neural net output to your y. If it is, it is the end of the pursuit. If not, go back to Steps 1, 2, and 3. Modify the weights and biases. You control four entities. Look at the new y". Repeat (if necessary) until satisfied

Note: it is necessary that the inputs are located in the interval [0, 1]. Likewise, the outputs are adjusted to [0, 1]. Why? The activation function 0 < 1/(1 + exp(-x)) < 1.

- How to modify the weight and bias? There is a mathematical way to do it. The Back propagation method …
- There are two R packages, nnet and neuralnet, that can be used to fit a neural network model.

- Download and activate the package ‘neuralnet.’

```
library(nnet)
library(neuralnet)

In <- 2
Out <-7
Data <- data.frame(In, Out)
Data
```

```
##   In Out
## 1  2   7
```

Step 1: Invoke the ‘neuralnet’ function.

```
Net1 <- neuralnet(Out ~ In, data = Data, hidden = 1)
Net1
```

```
## $call
## neuralnet(formula = Out ~ In, data = Data, hidden = 1)
##
## $response
##   Out
## 1   7
##
## $covariate
##
## [1,] 2
##
## $model.list
## $model.list$response
## [1] "Out"
##
## $model.list$variables
## [1] "In"
##
##
## $err.fct
## function (x, y)
## {
##     1/2 * (y - x)^2
## }
## <bytecode: 0x00000000150e6200>
## <environment: 0x00000000150e48a0>
## attr(,"type")
## [1] "sse"
##
## $act.fct
## function (x)
## {
##     1/(1 + exp(-x))
## }
## <bytecode: 0x00000000150e22e0>
## <environment: 0x00000000150e1630>
## attr(,"type")
## [1] "logistic"
##
```

```
## $linear.output
## [1] TRUE
##
## $data
##   In Out
## 1  2   7
##
## $exclude
## NULL
##
## $net.result
## $net.result[[1]]
##          [,1]
## [1,] 6.991231
##
##
## $weights
## $weights[[1]]
## $weights[[1]][[1]]
##          [,1]
## [1,] 2.320889
## [2,] 2.085726
##
## $weights[[1]][[2]]
##          [,1]
## [1,] 3.858378
## [2,] 3.137599
##
##
##
## $generalized.weights
## $generalized.weights[[1]]
##               [,1]
## [1,] -0.0002359839
##
##
## $startweights
## $startweights[[1]]
## $startweights[[1]][[1]]
```

```
##             [,1]
## [1,] -0.3291111
## [2,] -0.5642741
##
## $startweights[[1]][[2]]
##            [,1]
## [1,] 1.2083783
## [2,] 0.4875987
##
##
##
## $result.matrix
##                                [,1]
## error                  3.845042e-05
## reached.threshold      8.769312e-03
## steps                  3.000000e+01
## Intercept.to.1layhid1 2.320889e+00
## In.to.1layhid1         2.085726e+00
## Intercept.to.Out       3.858378e+00
## 1layhid1.to.Out        3.137599e+00
##
## attr(,"class")
## [1] "nn"
```

```
plot(Net1)
```

- It means the hidden layer has only one node.
- Look at the output.
- error function = sum(0.5(Observed – Predicted)^2). This is usually the set number
- Will need to keep working until the error is less than what was set. Or, one can set an upper limit of the number of iterations.
- need to display the network graphically
- Let us be more ambitious.
- We want to reproduce the square function by a neural network.
- The square function is not linear.

```
input <- 0:10
output <- input^2
Squares <- data.frame(input, output)
Squares
```

```
##    input output
## 1      0      0
## 2      1      1
## 3      2      4
## 4      3      9
## 5      4     16
## 6      5     25
## 7      6     36
## 8      7     49
## 9      8     64
## 10     9     81
## 11    10    100
```

- The goal is to reproduce the ‘square’ operation by a neural network.
- Let’s try three nodes in the hidden layer.

```
Net <- neuralnet(output ~ input, data = Squares, hidden = 2)
```

```
## Warning: Algorithm did not converge in 1 of 1 repetition(s) within the stepmax.
```

```
Net <- neuralnet(output ~ input, data = Squares, hidden = 3)
Net
```

```
## $call
## neuralnet(formula = output ~ input, data = Squares, hidden = 3)
##
## $response
##    output
## 1       0
## 2       1
## 3       4
## 4       9
## 5      16
## 6      25
## 7      36
## 8      49
## 9      64
## 10     81
## 11    100
##
## $covariate
##
##  [1,]  0
##  [2,]  1
##  [3,]  2
##  [4,]  3
##  [5,]  4
##  [6,]  5
##  [7,]  6
##  [8,]  7
##  [9,]  8
## [10,]  9
## [11,] 10
##
## $model.list
## $model.list$response
## [1] "output"
##
## $model.list$variables
## [1] "input"
##
##
```

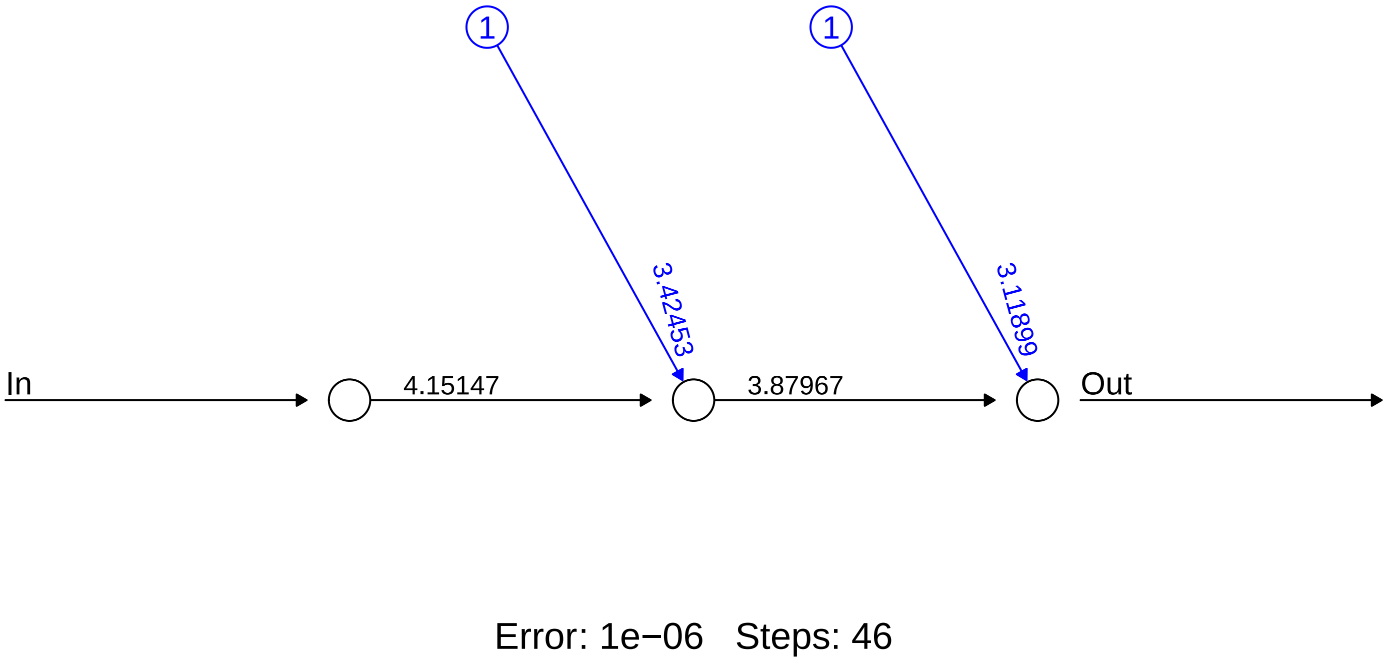
```
## $err.fct
## function (x, y)
## {
##     1/2 * (y - x)^2
## }
## <bytecode: 0x00000000150e6200>
## <environment: 0x0000000021e353d8>
## attr(,"type")
## [1] "sse"
##
## $act.fct
## function (x)
## {
##     1/(1 + exp(-x))
## }
## <bytecode: 0x00000000150e22e0>
## <environment: 0x0000000021e35870>
## attr(,"type")
## [1] "logistic"
##
## $linear.output
## [1] TRUE
##
## $data
##    input output
## 1      0      0
## 2      1      1
## 3      2      4
## 4      3      9
## 5      4     16
## 6      5     25
## 7      6     36
## 8      7     49
## 9      8     64
## 10     9     81
## 11    10    100
##
## $exclude
## NULL
```

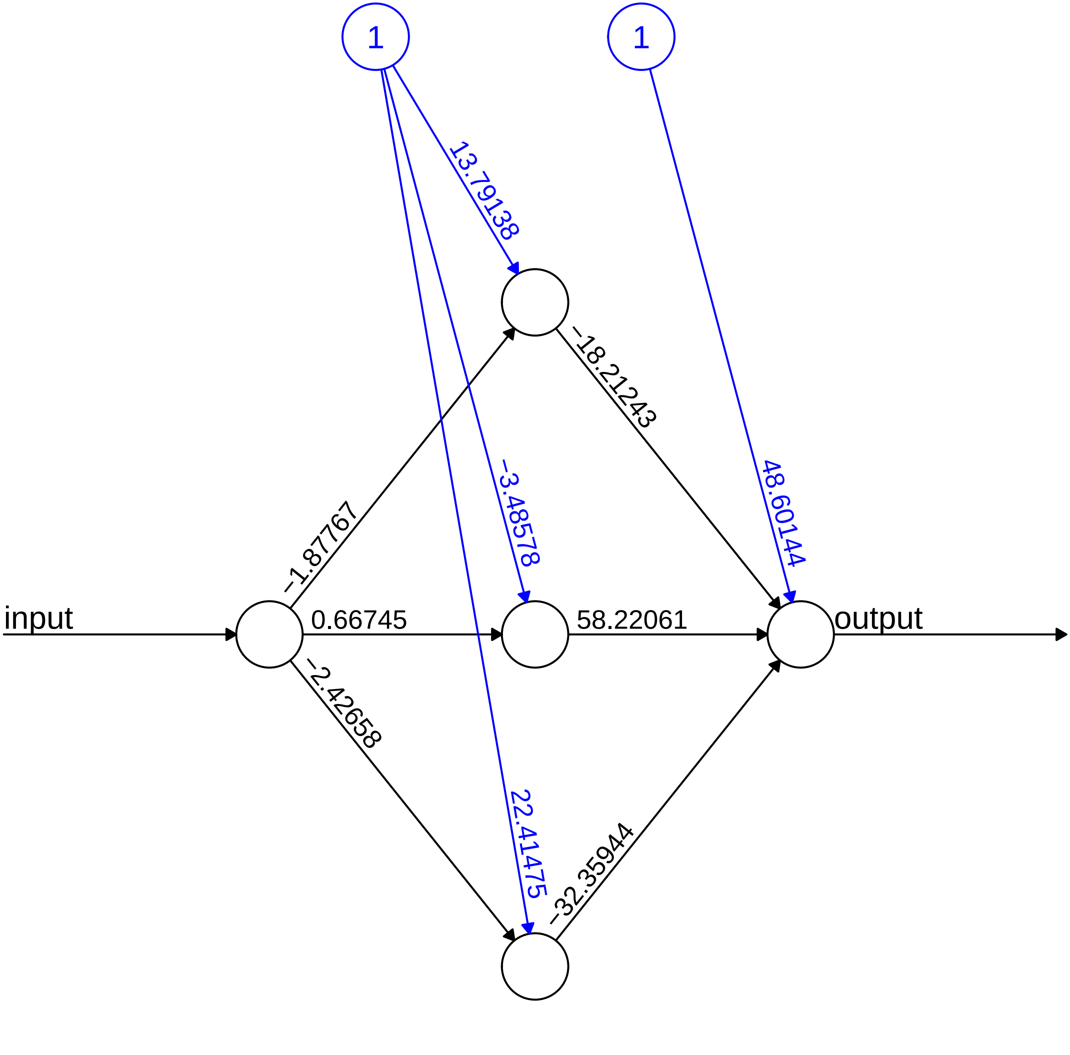
```
##
## $net.result
## $net.result[[1]]
##            [,1]
##  [1,] -0.190604
##  [2,]  1.272455
##  [3,]  4.036176
##  [4,]  8.815340
##  [5,] 16.003092
##  [6,] 25.156029
##  [7,] 35.857024
##  [8,] 49.067465
##  [9,] 63.976821
## [10,] 81.004158
## [11,] 99.999795
##
##
## $weights
## $weights[[1]]
## $weights[[1]][[1]]
##            [,1]      [,2]      [,3]
## [1,]  3.4060658 -17.70588 -9.476564
## [2,] -0.7338856   1.88462  1.291891
##
## $weights[[1]][[2]]
##           [,1]
## [1,]  43.15293
## [2,] -44.78377
## [3,]  36.30255
## [4,]  31.16548
##
##
##
## $generalized.weights
## $generalized.weights[[1]]
##               [,1]
##  [1,] -4.514186090
##  [2,] -5.763797231
##  [3,] -0.298350715
```

```
##  [4,] -0.086812233
##  [5,] -0.034602022
##  [6,] -0.016265300
##  [7,] -0.009401644
##  [8,] -0.006165552
##  [9,] -0.003743972
## [10,] -0.003076394
## [11,] -0.001453333
##
##
## $startweights
## $startweights[[1]]
## $startweights[[1]][[1]]
##           [,1]       [,2]     [,3]
## [1,] -1.056504 0.06167203 1.706800
## [2,]  1.251366 0.36630514 1.201194
##
## $startweights[[1]][[2]]
##           [,1]
## [1,]  1.065220
## [2,] -1.829726
## [3,]  1.171569
## [4,] -1.190415
##
##
##
## $result.matrix
##                                [,1]
## error                  9.793636e-02
## reached.threshold      9.785311e-03
## steps                  2.734200e+04
## Intercept.to.1layhid1  3.406066e+00
## input.to.1layhid1     -7.338856e-01
## Intercept.to.1layhid2 -1.770588e+01
## input.to.1layhid2      1.884620e+00
## Intercept.to.1layhid3 -9.476564e+00
## input.to.1layhid3      1.291891e+00
## Intercept.to.output    4.315293e+01
## 1layhid1.to.output    -4.478377e+01
```

```
## 1layhid2.to.output     3.630255e+01
## 1layhid3.to.output     3.116548e+01
##
## attr(,"class")
## [1] "nn"
```

```
plot(Net)
```


1
1
3.42453
3.11899
In
4.15147
3.87967
Out
Error: 1e−06 Steps: 46


1
1
13.79138
−18.21243
−3.48578
48.60144
−1.87767
input
0.66745
58.22061
output
−2.42658
22.41475
−32.35944
Error: 0.132579 Steps: 40552