# Assignment 3

## Task 1.

A Neural Network (NN) model was trained to learn the sine function (sin(x)). The data set consisted of 500 samples with two columns, ‘var’ and ‘sin’. Var is the value of x which is a value in a range from 1 to 10 and sin is sin(x), sin of x.

25 samples were used for training and 475 for test. Starting weights were in the range -1 to 1 and random.

Plot of the training data (black), the test data (blue) and the predictions (red) of the NN model.

Chart, line chart

Description automatically generated

The results look good. But as seen in the plot there are some error, the prediction doesn’t follow the data perfectly. But very well.

## Task 2.

Three new neural networks were trained, with different custom activation functions. One linear (h1=x), one ReLU (h2=max{0,x}) and one softplus (h3=ln(1+exp(x))). The previous task used the default sigmoid function, also known as the “logistic”. Beyond the activation function, nothing was changed from task 1.

Chart, line chart

Description automatically generated

Linear activation function. As seen, it gives a straight line. Also, it can’t approximate a non-linear function.

Chart, line chart

Description automatically generated

Softplus activation function. Fits the best, because to the difference of a linear activation function it considers unknown variables.

ReLU was not implementable in the same way as the previous two custom activation functions. Because {0,x} is not a differentiable function, it cannot be derived at zero. In other words the left side and right side of the derivative is not identical and this is not something rstudio could handle. There are workarounds, however we were not tasked with implementing the workarounds.

## Task 3.

Same steps as in task 1, however the range for x was increased to 0 to 50.

Black is training, blue is test and red is prediction.

Chart, line chart

Description automatically generated

The model looks good up to x=10, after that it starts to look worse, so mixed results. Then it seems to converge to some value.

## Task 4.

The prediction in the previous plot seems to converge to some value. NN and NN$weights were looked at to explain why.

The activation function for the NN was the default sigmoid function, 1/(1 + exp(-x))=.

Looking at the graph the value being converged to is approximately somewhere between -0.3 and -0.4 or -0.5.

The prediction for a large x value, 50, was calculated with the bias, so a vector of 1 and 50. The activation function was used on the bias and x\*weights which looked like this 1/(1 + exp(-x\*weights)). The result was a vector of 1s and almost zeroes, first position in the vector corresponds to bias and the rest to each hidden node in turn. Based on this result it is possible to see which hidden units were used in the output, those that had the value of 1. Those with value of zero were not used. Those that had a value of 1 were the bias and input variable 4, 6, 7, 9 and 10.

Summing up these values from nn$weights, or multiplying the previously mentioned vector with the vector for nn$weights [[1]][[2]] we got the value the models prediction is converging towards, -0.3474075. nn$weights [[1]][[2]] is the values of weights for the output layer in the NN.

This happens because as x grows the sigmoid function approaches 1 or 0 depending on if x is positive or negative, positive x goes towards 1 and negative x towards 0.

## Task 5.

Same as task 1 but with a twist. So, the range for x is 1 to 10 again. Instead of trying to predict sin(x) based on x, it is the other way around, predicting x based on sin(x).

Black is training data, test is blue and predictions red.

Chart

Description automatically generated

Result is bad, prediction doesn’t match the data. Because the model is trying to get a linear prediction, x, from a non-linear function, sin(x). For example for sin(0) there can be 4 different values for x in the range [1,10] (zero, pi, 2\*pi and 3\*pi for the curious). However, the model can’t see that.

# Appendix

library(neuralnet)

#####Task 1#####

set.seed(1234567890)

Var <- runif(500, 0, 10)

mydata <- data.frame(Var, Sin=sin(Var))

tr <- mydata[1:25,] # Training

te <- mydata[26:500,] # Test

# Random initialization of the weights in the interval [-1, 1]

winit <- runif(25, -1, 1)

# Training model

nn <- neuralnet(Sin~Var, data=tr, hidden=10, startweights=winit)

# Plot of the training data (black), test data (blue), and predictions (red)

plot(tr, cex=2)

points(te, col = "blue", cex=1)

points(te[,1],predict(nn,te), col="red", cex=1)

#####Task 2#####

winit <- runif(25, -1, 1)

linear <- function(x) x

nnLinear <- neuralnet(Sin~Var, data=tr, hidden=10, act.fct = linear, startweights=winit)

winit <- runif(25, -1, 1)

softplus <- function(x) log(1 + exp(x))

nnSoftPlus <- neuralnet(Sin~Var, data=tr, hidden=10, act.fct = softplus)

#install.packages('sigmoid')

library(sigmoid)

#relu()

winit <- runif(25, -1, 1)

nnReLu <- neuralnet(Sin~Var, data=tr, hidden=10, act.fct =relu, startweights=winit)

#plotting

#linear

plot(tr, cex=2)

points(te, col = "blue", cex=1)

points(te[,1],predict(nnLinear,te), col="red", cex=1)

#softplus

plot(tr, cex=2)

points(te, col = "blue", cex=1)

points(te[,1],predict(nnSoftPlus,te), col="red", cex=1)

#ReLu

plot(tr, cex=2)

points(te, col = "blue", cex=1)

points(te[,1],predict(nnReLu,te), col="red", cex=1)

##### Task 3 #####

set.seed(1234567890)

Var <- runif(500, 0, 50)

mydata <- data.frame(Var, Sin=sin(Var))

tr <- mydata[1:25,] # Training

te <- mydata[26:500,] # Test

# Random initialization of the weights in the interval [-1, 1]

winit <- runif(25, -1, 1)

nntask3 <- neuralnet(Sin~Var, data=tr, hidden=10, startweights=winit)

# Plot of the training data (black), test data (blue), and predictions (red)

plot(tr, cex=2)

points(te, col = "blue", cex=1)

points(te[,1],predict(nntask3,te), col="red", cex=1)

library(Hmisc)

minor.tick(ny = 5, tick.ratio = 1) #line seems to be exactly at -0.347

##### Task 4 #####

print(nntask3$weights)

# Look at some (large) x value (1 is the bias)

x <- c(1, 50)

print(x)

# Print the hidden units

hidden <- c(1, 1/(1+exp(-x %\*% nntask3$weights[[1]][[1]])))

print(hidden)

# Print the prediction

prediction <- hidden %\*% nntask3$weights[[1]][[2]]

print(prediction)

# Print the weights

print(nntask3$weights)

#Plot of neural network

plot(nntask3)

# [4,] -1.2142028

#[6,] 0.6218254

#[7,] -1.2120181

#[9,] -1.2250588

#[10,] 2.1300140

summm<- sum(-1.2142028, 0.6218254,-1.2120181,-1.2250588,2.1300140,0.5520328)

#exactly -0.3474075

##### Task 5 #####

#predict x from sin(x)

set.seed(1234567890)

Var <- runif(500, 0, 10)

mydata <- data.frame(Var, Sin=sin(Var))

# Random initialization of the weights in the interval [-1, 1]

winit <- runif(25, -1, 1)

#NN that predicts x from sin(x) on the whole dataset (all point are training)

nnreversed <- neuralnet(Var~Sin, data=mydata, hidden=10, startweights=winit,threshold = 0.1)

reversedprediction=predict(nnreversed,mydata)

# Plot of the training data (black), test data (blue), and predictions (red)

plot(mydata, cex=2) #black dots = training data

points(reversedprediction,mydata[,2],col="red", cex=1)