STA4026S CA2

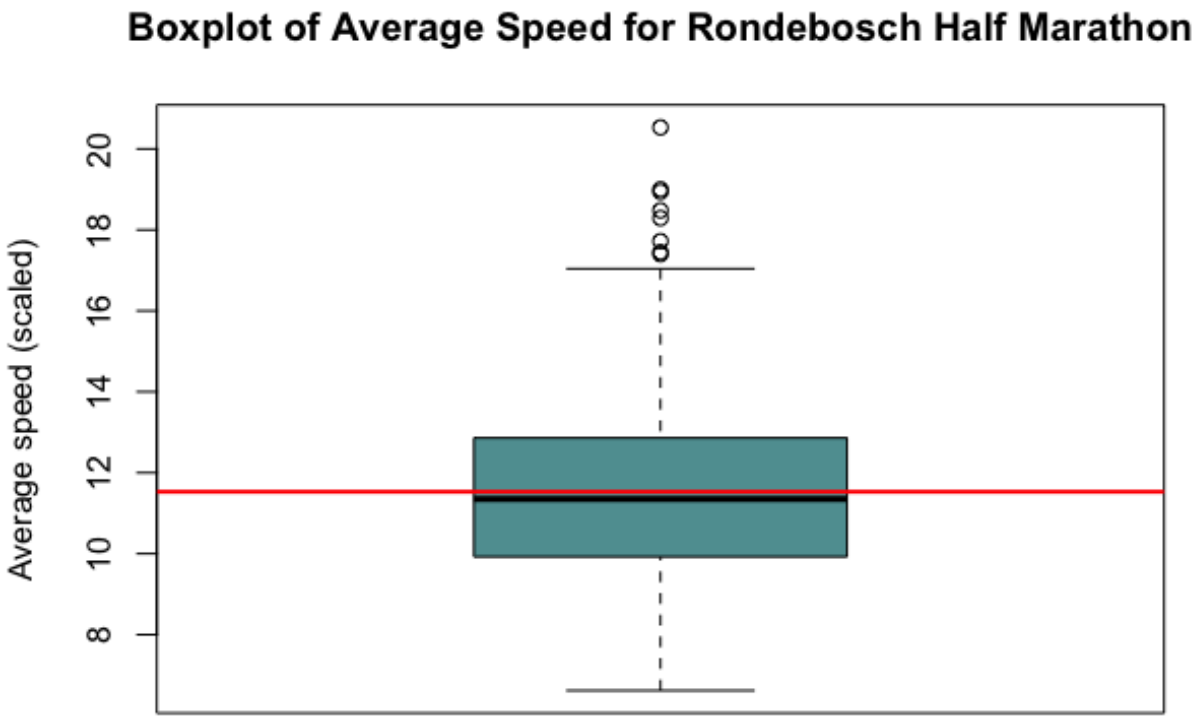
Antony Fleischer

02/10/2021

Question 1

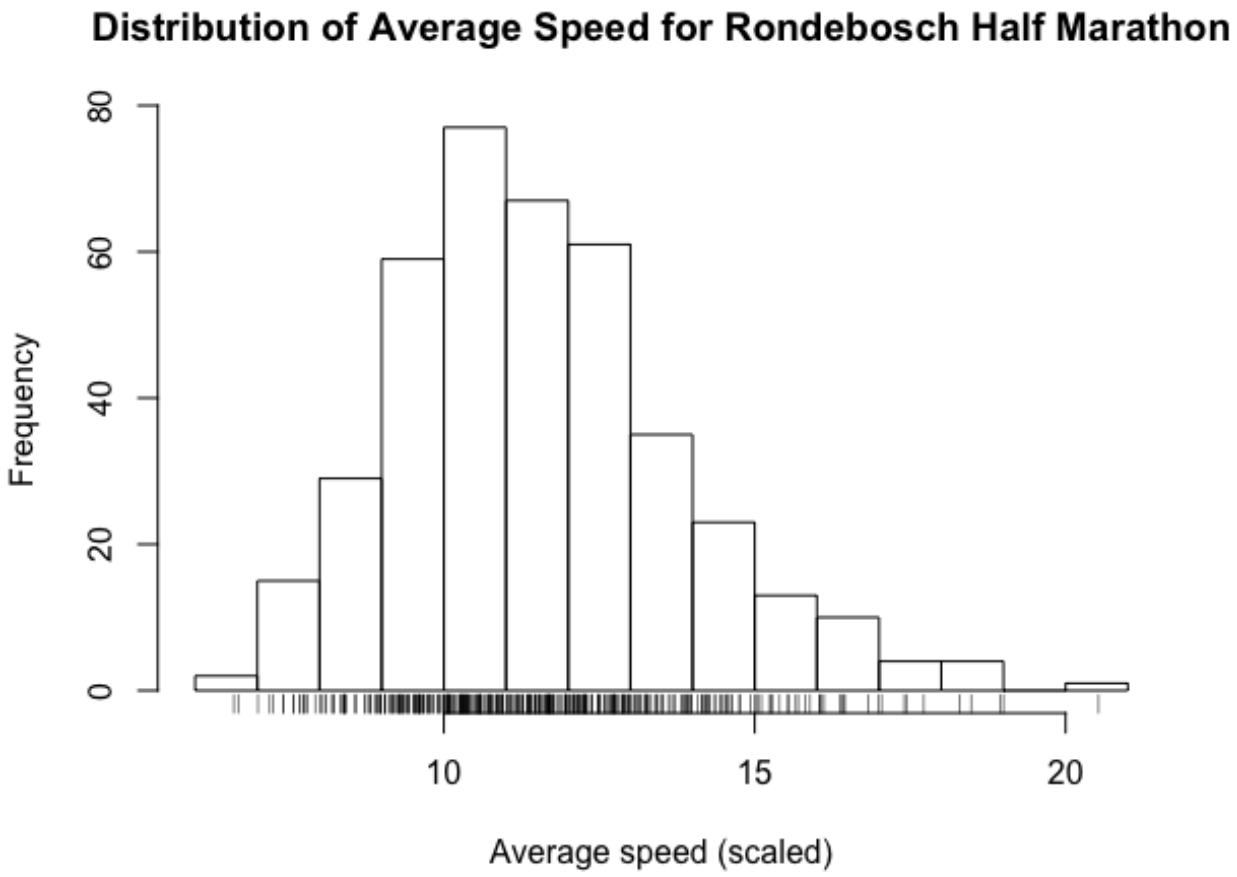
a)

To explore the empirical relationship between the response - scaled average speed (notated as AvgSpeed from here) - and the predictors, several plots were constructed. To examine the distribution of the response, a boxplot and a density histogram were plotted. Both of the plots show that the data is slightly positively skewed, with the red line in the boxplot showing that the mean is slightly larger than the median for AvgSpeed.

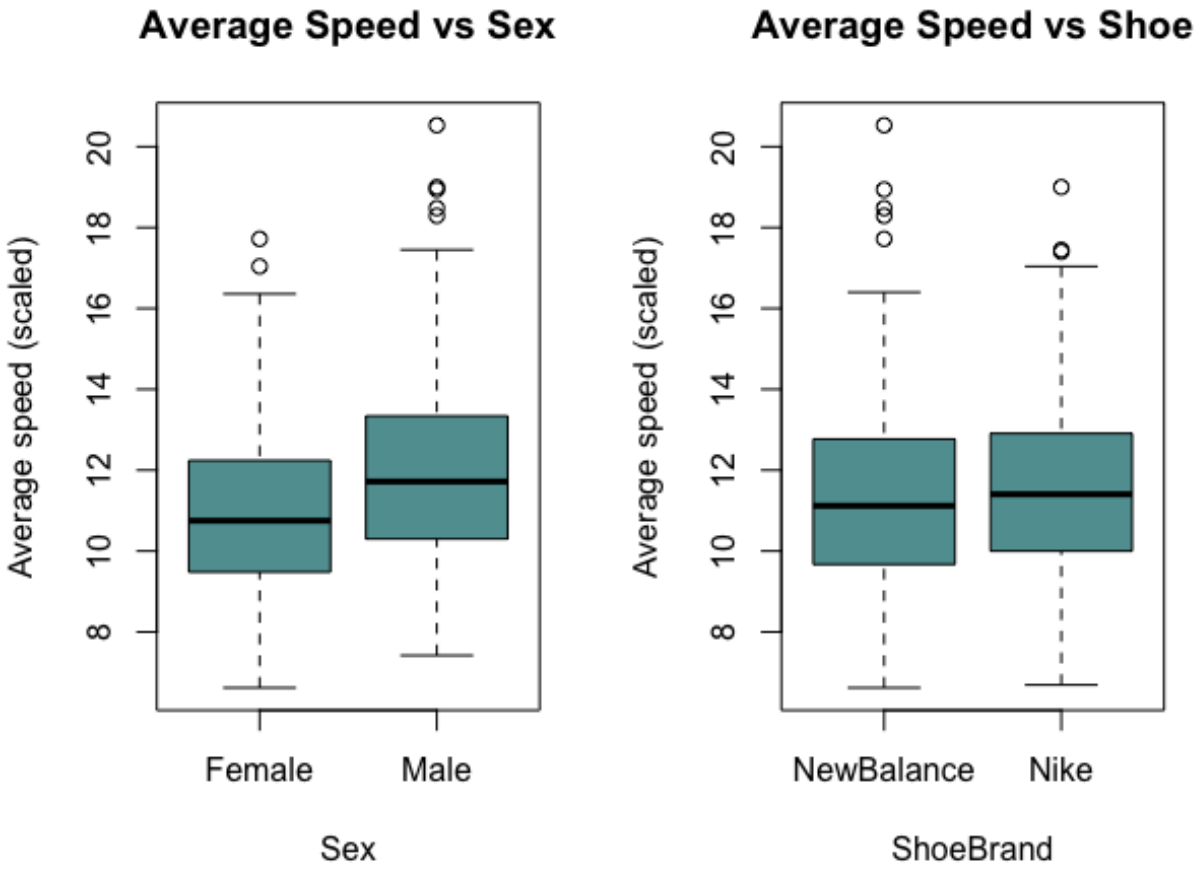


Both plots show that about 50% of AvgSpeed values in the distribution lie between an AvgSpeed value of 10 - 13. This is reinforced by the one-dimensional rug at the bottom of the distribution plot, which shows

that the number of responses is much denser around AvgSpeedValues of 10-13. The boxplot adds some additional information, by showing that there are several statistical outliers. These outliers represent the fastest performers at the half marathon.

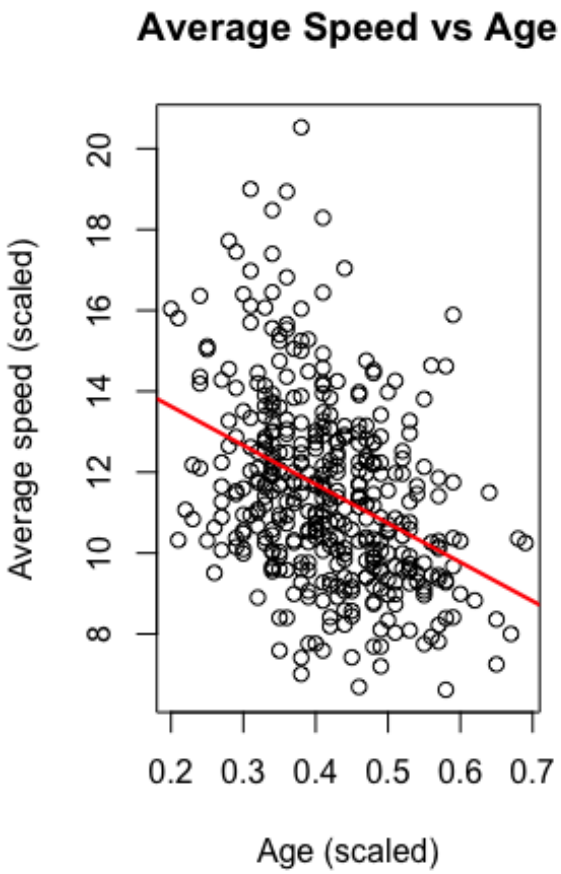


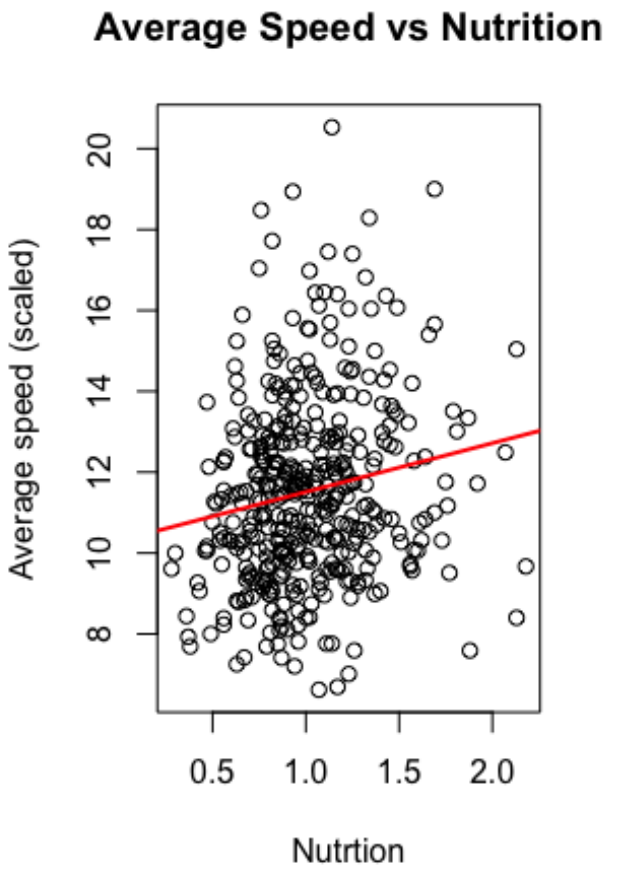
Having explored the response independently, the relationships between different predictors and the response was plotted. The left-hand side of the plot below shows the box plots of AvgSpeed vs Female and Male, respectively. It is immediately clear that most of the outliers from the previous boxplot are Male, as only two Female athletes recorded an average speed close to the upper extreme of the previous boxplot. This is contrasted by the higher upper extreme of the Male boxplot, and the higher number of outliers. Additionally, the lower, middle and upper quartiles are all higher for Male, and so one could hypothesise that on average a Male ran the half-marathon faster than a Female. However, the averages of AvgSpeed are closer than expected, with a Female average of 10.955 and a Male average of 11.995. The sample average was 11.530.



The right-hand boxplots show AvgSpeed vs New Balance and Nike, respectively. The faster athletes in the sample tend to wear New Balance over Nike. However, there is not much inference to be drawn from these two plots. In general, Shoe Brand does not seem to have a strong correlation with AvgSpeed. To confirm this, the linear correlation value was calculated. $cor_{(AvgSpeed,Shoe)=0.0290451}$, which further reinforces the negligible effect of Shoes on AvgSpeed. For comparison, the correlation between Avg Speed and Sex was 0.2241107.

Lastly, the relationship between the response and the continuous predictors was investigated. There is no obvious linear relationship between either of the predictors and AvgSpeed, but by constructing the linear regression of $lm(AvgSpeed\ Age)$ and $lm(AvgSpeed\ Nutrition)$, the linear trend is presented. This does not mean that a linear model captures the relationship best, but the linear model attempts to describe the relationship. There is an inverse relationship between AvgSpeed and Age, which is as expected. A younger person is generally fitter and more athletic. The correlation between the two variables is -0.383. Conversely, there is a positive correlation between AvgSpeed and Nutrition, but this is also as expected. The linear correlation between the values is 0.169. This suggests that an athlete who took in more water during the race ran a faster half marathon.





b)

The input vector, x_i, for a neural network is encoded as: $x_i = [x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}]$, where i is the encoding of the observation and $1, 2, 3, 4$ are the encoding of the predictors for the i_{th} observation.

c)

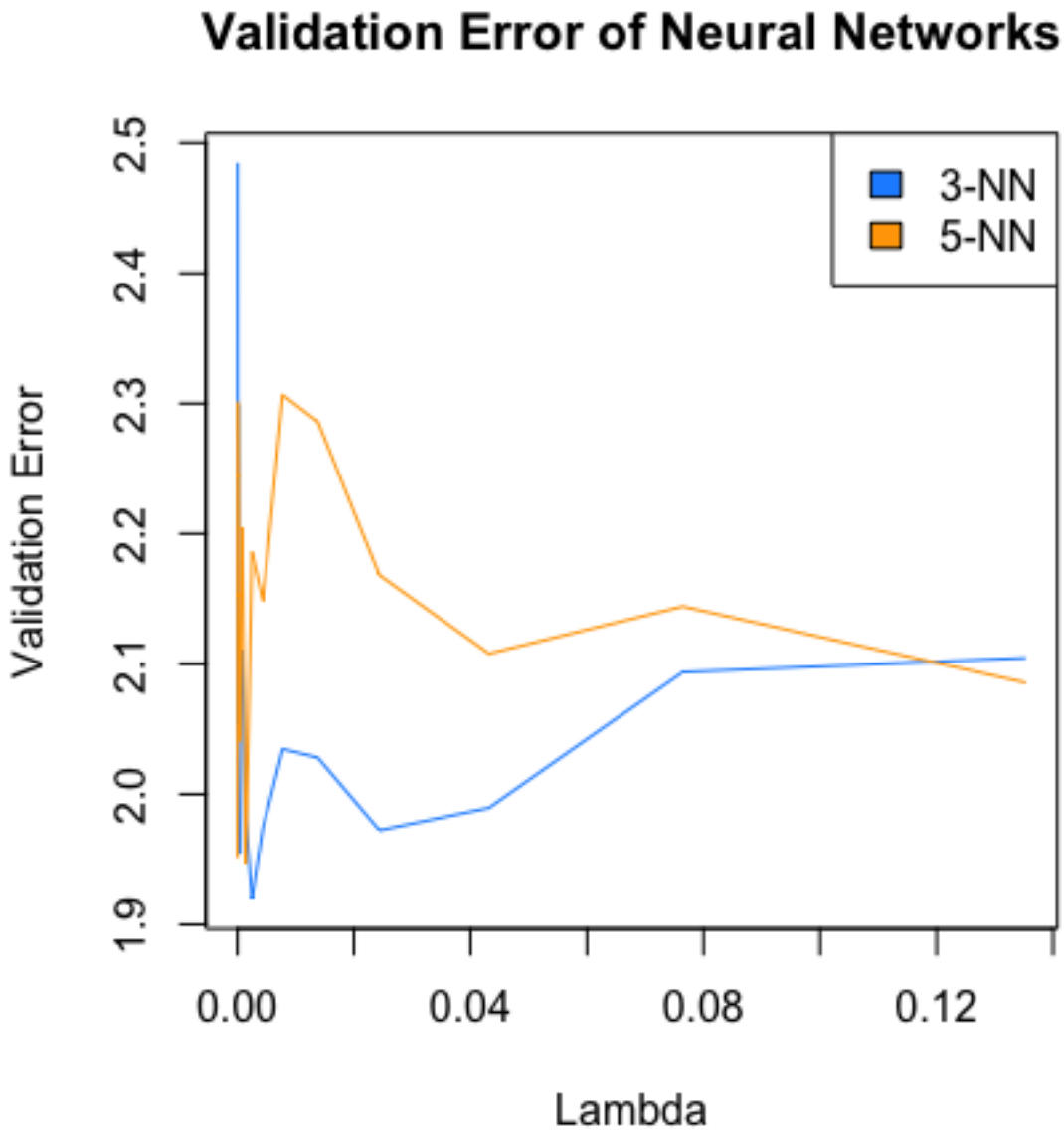
Considering the Rondebosch Half Marathon dataset has a continuous response, this neural network is being used for regression. Consequently, I have used a logistic/sigmoid activation function for the hidden layer and a linear activation for the output layer. The linear activation function on the output layer is the obvious choice because of the continuous and regression qualities of the neural network. A linear activation does not contain the output either, so the range of outputs is not constrained. The hidden layer activation decision is slightly more arbitrary, and I have chosen the logistic activation because I am familiar with using its derivative for backpropagation. Realistically, using the *ReLU* or *tanh* activations would not cause a noticeable difference in the neural network's output or performance.

I have used Mean Squared Error as the cost function, because it captures the average error across a multi-valued output, and can be used with a regression problem. It compares the expected value of output with the predicted value. Furthermore, it goes hand-in-hand with a linear activation function. Lastly, I used the L2 penalty to implement regularisation when optimising the model's parameters. The L2 penalty penalises model parameters by reducing their magnitude and reducing overfitting, whilst the L1 penalty reduces 'less important' parameter values towards 0, effectively removing features from the model. The L1 penalty cannot

be used in regression problems, and, additionally, since there are only 4 features in the sample, I did not want to remove any of the features.

d)

To select a prediction model, two models were optimised over a range of L2 penalty λ values. The first model is a 3-network, and the second is a 5-network. A validation error is a proxy of Out-of-Sample Error and so we will select the model and λ value that results in the lowest validation error. The results of running an *nlm* optimisation across 16 λ values from $e^{-}10$ to $e^{-}2$, and including 0, are shown in the plot below.



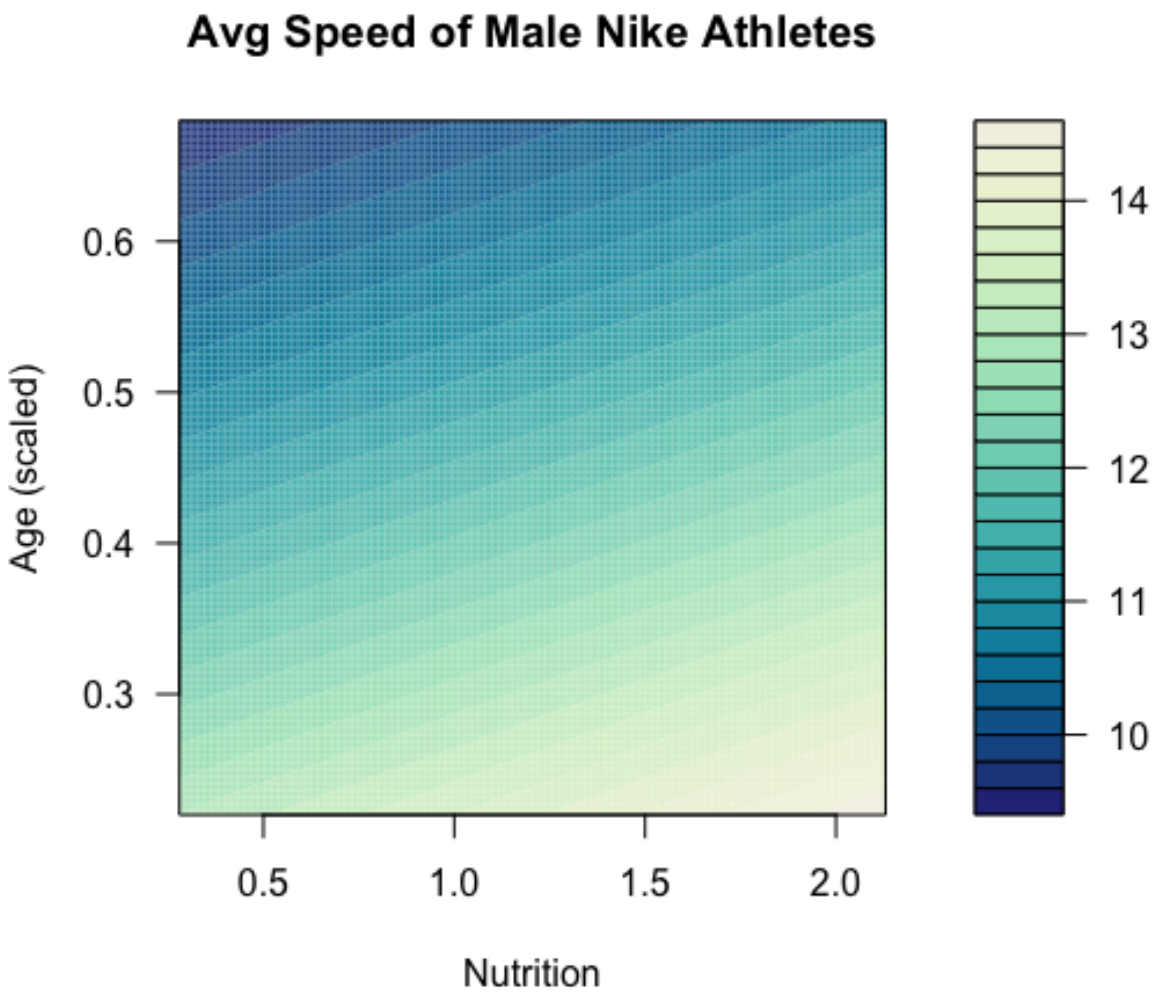
The plot shows that the 3-network consistently has a lower validation error, and reached a validation error minimum when $\lambda = 0.0024$. The MSE error of the 3-model at this point was 1.92. Both models exhibited an improvement in predictive performance when $\lambda > 0$, which shows that both models were slightly overfitting without regularisation.

Despite its additional depth, the 5-network is overfitting on the training data and as a result, it is not predicting well on 'unseen' data. The general trend shows that the 3-network is a better predictor on the validation set, and so we will select it for prediction. We can also see that increasing λ does apply the brakes on model complexity initially, but validation error does not show a general linear relationship concerning λ. Therefore, we can conclude that the 3-network was not overly complex or overfitting for low values of λ, and so we will select an L2 λ value of 0.0024.

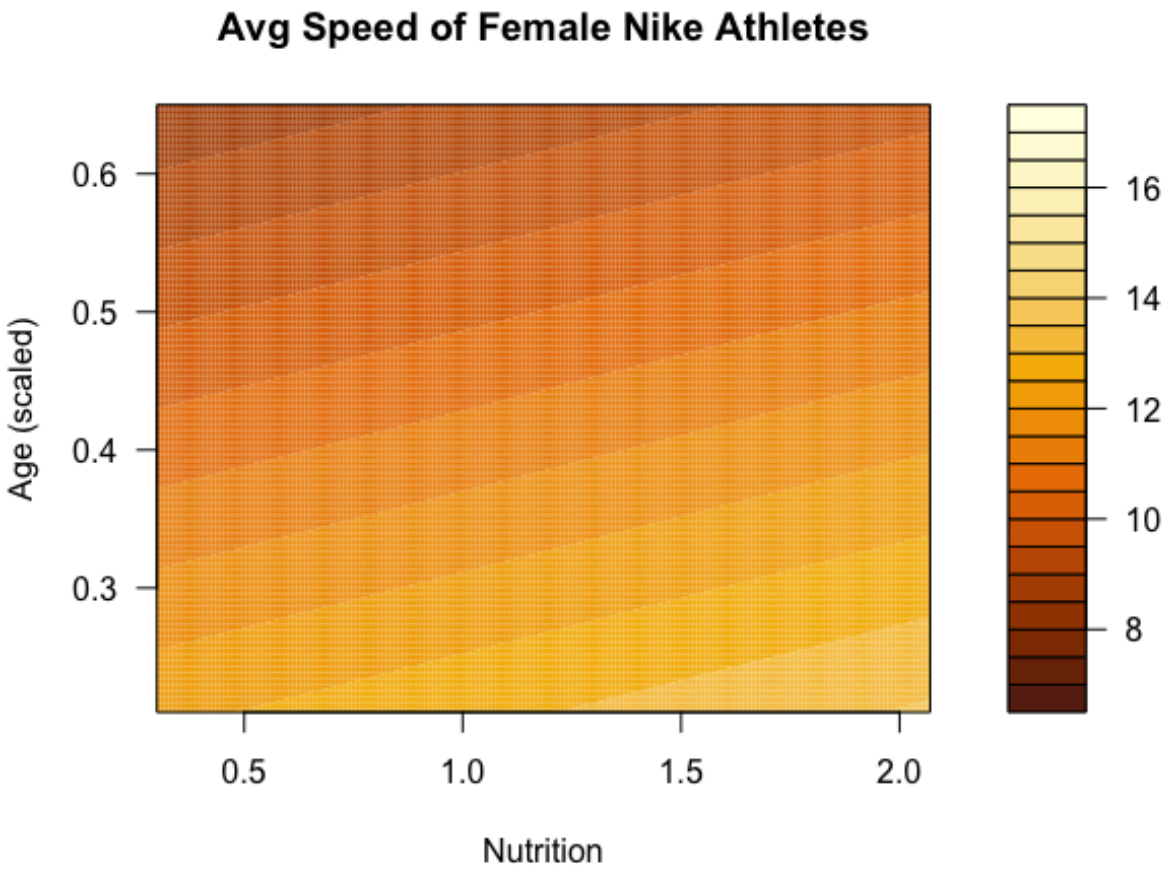
e)

By creating a dense lattice across the range of possible AvgSpeed values that Male, Nike athletes presented in the sample training data, it is possible to view a filled contour plot of the optimal model's predictions for each coordinate of the lattice. The optimal model uses a pocket solution from question d as its parameter estimates, rather than the parameter estimates of the last *nlm* iteration. The plot below shows the same empirical relationships we explored earlier. The relationship between age and AvgSpeed is negative, and the relationship between nutrition and AvgSpeed is positive. The contours are slightly radial, which also reinforces a suggestion from the exploratory data analysis: the positive effect of lower age is stronger than the effect of a higher Nutrition. This can be seen by the gradual gradient from white to blue as X decreases, and the faster transition from white to blue as Y increases. This can be conceptualised by imagining a plot where the contours were completely horizontal, which would show that only Age affects AvgSpeed, or where the contours are vertical, which would show that only Nutrition affects Avg Speed. A contour angle of 45 degrees would show an equal contribution by both predictors.

To create such a dense contour plot, there are $200 * 200$ data points. As a result, the faster predictions do not show clearly on the plot, as they make up a very small proportion of the lattice data set. This is why the z-values in the colour legend do not increase past a 15 for AvgSpeed. Additionally, the outlying points in the training data are likely partially interpreted as noise by the neural network, and as a result, it will attempt to fit the majority of the data and so it will underpredict faster athletes' AvgSpeed.



The same contour plot was constructed for Female Nike athletes. The trend is very similar. Younger females with more liquid intake are predicted to have higher AvgSpeed. The range of AvgSpeed is also larger for Female, so there appear to be more contour levels throughout the plot. This is shown by the larger range of z-values in the contour legend on the right. The angle of the contours is slightly less steep than the Male Nike plot, which suggests that the effect of high Nutrition levels for Female athletes does not increase AvgSpeed as much as it does for Male athletes. This could be interpreted as Age having a more significant contribution on AvgSpeed for females.



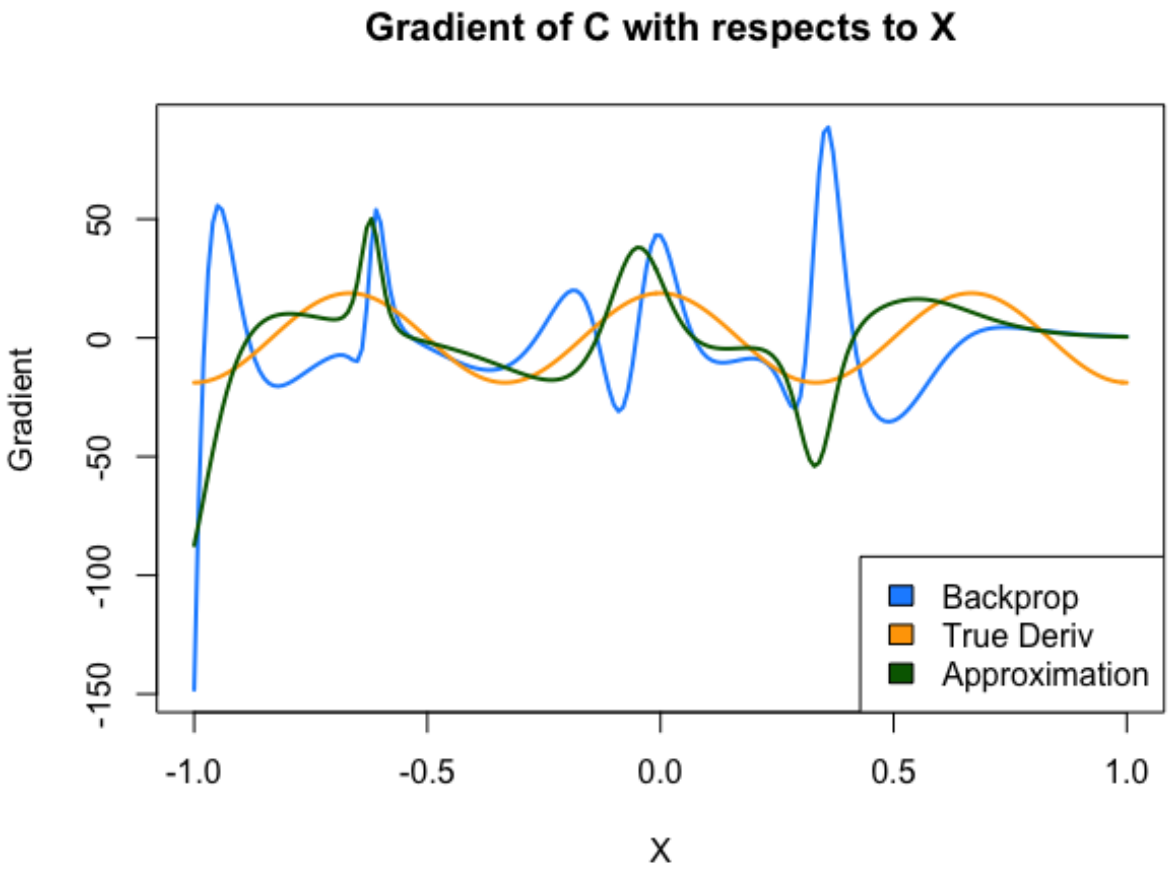
f)

I have predicted the test set response using my optimal model.

Question 2

a)

By using the central limit theorem, we can approximate the gradient of the cost function concerning the inputs. This is shown by the dark green line in the plot.



b)

Yes, the plot (above) does suggest a means for conducting regularisation which does not involve penalising the parameters. The approximated gradient, when compared to the true derivative of the cost function - shown in orange, is highly sensitive to specific data points. I am not entirely sure how one would do this, but I have an idea: by applying a linear transformation to the inputs, particularly those that are beyond a certain number of standard deviations from the sample mean, one could reduce the sensitivity of the model to inputs and scale the gradients to within a more accurate range.

c)

The actual gradients of the cost function for the inputs have been superimposed on the plot above. By backpropagating from the first hidden layer gradients, one can derive the true gradient with respect to the inputs.

Appendix

Question 1

```
### STA4026S - Analytics - CA2
### FLSANT005 - Antony Fleischer

dat_train = read.table('Rondebosch21km_2021_Train.txt', h = TRUE)
dat_val = read.table('Rondebosch21km_2021_Validate.txt', h = TRUE)
dat_test = read.table('Rondebosch21km_2021_Test.txt', h = TRUE)
attach(dat_train)

### Question 1
### ====================
## a)
#Five number summary
boxplot(Speed_21km, data = dat_train, ylab = "Average speed (scaled)", main = "Boxplot of Average Speed
summary(Speed_21km)
subset(dat_train, dat_train$Speed_21km > 17)
abline(h = mean(Speed_21km), col = "red", lwd = 2)

#Distribution and skewness
par(mfrow = c(1, 1))
hist(Speed_21km, xlab = "Average speed (scaled)", main = "Distribution of Average Speed for Rondebosch
rug(Speed_21km)

#Speed vs Sex
par(mfrow = c(1, 2))
plot(Speed_21km ~ Sex, col = "cadetblue", main= "Average Speed vs Sex", ylab = "Average speed (scaled)"
cor(Speed_21km, as.numeric(Sex))

mean(Speed_21km)
mean(subset(Speed_21km, Sex == 'Male'))
mean(subset(Speed_21km, Sex == 'Female'))

#Speed vs Shoe
plot(Speed_21km ~ ShoeBrand, col  = "cadetblue", main = "Average Speed vs Shoe", ylab = "Average speed
cor(Speed_21km, as.numeric(ShoeBrand))

#Speed vs Age
plot(Speed_21km ~ Age_Scl, main = "Average Speed vs Age", ylab = "Average speed (scaled)", xlab = "Age
age_lm = lm(Speed_21km~Age_Scl)
abline(age_lm, col = 'red', lwd = 2)
cor(Speed_21km, Age_Scl)


#Speed vs Nutrition
plot(Speed_21km ~ Nutrition,  main = "Average Speed vs Nutrition", ylab = "Average speed (scaled)", xla
age_lm = lm(Speed_21km~Nutrition)
abline(age_lm, col = 'red', lwd = 2)
cor(Speed_21km, Nutrition)

## b)
X_train = matrix(cbind(Nutrition,Age_Scl,as.numeric(Sex)-1, as.numeric(ShoeBrand)-1), ncol = 4)
```

```
Y_train = matrix(dat_train$Speed_21km, ncol = 1)

# Specify activation functions for the hidden and output layers:
sig1 = function(x) #Logistic activation function for hidden layers
{
  1/(1+exp(-x))
}

sig2 = function(x) #Logistic activation function for output layer.
{
  x
}

g = function(Yhat,Y) #Cross-entropy cost function
{
  0.5*(Yhat-Y)^2
}

neural_net = function(X,Y,theta, m, lam)
{
  # Relevant dimensional variables:
  N = dim(X)[1]
  p = dim(X)[2]
  q = dim(Y)[2]

  # Indexing of theta parameter vector
  index = 1:(p*m)
  W1    = matrix(theta[index],p,m)
  index = max(index) + 1:(m*q)
  W2    = matrix(theta[index],m,q)
  index = max(index) + 1:m
  b1    = matrix(theta[index],m,1)
  index = max(index) + 1:q
  b2    = matrix(theta[index],q,1)
  out   = rep(0,N)
  error = rep(0,N)

  #Matrix Approach
  S1 = X%*%W1
  b1_t = rep(b1, dim(X)[1])
  b1_t = matrix(t(b1_t), nrow = dim(X)[1], byrow = TRUE)
  S1 = S1 + b1_t
  Z1 = sig1(S1)

  S2 = Z1%*%W2
  b2_t = rep(b2, dim(X)[1])
  b2_t = matrix(t(b2_t), nrow = dim(X)[1], byrow = TRUE)
  S2 = S2 + b2_t
  Z2 = sig2(S2)

  E1 = (0.5*sum((Z2-Y)^2))/N
  E2 = E1 + lam/N*(sum(W1^2)+sum(W2^2))

  return(list(out = Z2, E1 = E1, E2 = E2))
```

```
}

obj = function(pars)
{
  res = neural_net(X, Y, pars, m, l)
  return(res$E2)
}

## c)
#Most work here is done in the neural_net function
par(mfrow = c(1,1))

p = dim(X_train)[2]
q = dim(Y_train)[2]
m = 4
np = p*m+m*q+m+q
theta = runif(np,-2,2)
nn = neural_net(X_train, Y_train, theta, m, 0)

## d)
X_val = matrix(cbind(dat_val$Nutrition, dat_val$Age_Scl, as.numeric(dat_val$Sex)-1, as.numeric(dat_val$
Y_val  = matrix(dat_val$Speed_21km, ncol = 1)

#Number of params for 3-network.
a = 3
np_a = p*a+a*q+a+q

#Number of params for 5-network.
b = 5
np_b = p*b+b*q+b+q

#Create empty variables to store min errors, respective lambda values and weight estimates.
#Define lambda sequence
lams    = c(0,exp(seq(-10,-2, length =15)))

ValError_A = rep(NA,length(lams))
ValError_B   = rep(NA,length(lams))
estimateA = NA
estimateB = NA
minErrorA = 999
minErrorB = 999
lambdaA = NA
lambdaB = NA


 for(i in 1:16)
 {
   X = X_train
   Y = Y_train
   l = lams[i]
   theta_a = runif(np_a,-1,1)
   theta_b = runif(np_b, -1,1)
   m = a
   res_opt_a   = nlm(obj,theta_a, iterlim = 400)
```

```
   m = b
   res_opt_b   = nlm(obj,theta_b, iterlim = 400)
   X = X_val
   Y = Y_val
   resA = neural_net(X_val,Y_val,res_opt_a$estimate,3,lams[i])
   resB = neural_net(X_val,Y_val,res_opt_b$estimate,5,lams[i])
   ValError_A[i] = resA$E2
   ValError_B[i] = resB$E2


   # plot(Y_val)
   # points(resA$out, col = 'dodgerblue1')
   # points(resB$out, col = 'orange')
   #
   # print(res_opt_a$code)
  if (resA$E2 < minErrorA)
   {
    print("Min error found A")
    print(i)
     print(resA$E2)
     minErrorA = resA$E2
     estimateA = res_opt_a$estimate #Usi
     lambdaA = lams[i]
   }
   if (resB$E2 < minErrorB)
   {
     print("Min error found B")
      print(i)
    # print(res_opt_b$estimate)
     print(resB$E2)
     minErrorB = resB$E2
     estimateB = res_opt_b$estimate
     lambdaB = lams[i]
   }



}

resB = neural_net(X_val,Y_val,res_opt_a$estimate,3,lams[16])
resC = neural_net(X_val,Y_val,estimateA,3,lambdaA)
resB$E2
resC$E2

#Min and max values help when plotting val_error vs lambda.
min = min(c(minErrorA, minErrorB))
max = max(c(max(ValError_A), max(ValError_B)))
plot(ValError_A~lams, type = 'l', col = 'dodgerblue1',
     ylim = c(min, max),
     lwd = 1,
     ylab = "Validation Error",
     xlab = "Lambda",
     main = "Validation Error of Neural Networks")
```

```
lines(ValError_B~lams, lwd = 1, col = 'orange')
legend(x = 'topright', legend=c("3-NN", "5-NN"),
       fill = c("dodgerblue1","orange"))


which.min(c(minErrorA, minErrorB))
minErrorA
lambdaA

## e)
#Define male Nike lattice.
male = subset(dat_train, (dat_train$ShoeBrand=='Nike'))
male = subset(male, male$Sex=='Male')

M = 200
x1 = seq(min(male$Nutrition), max(male$Nutrition), length = M)
x2 = seq(min(male$Age_Scl), max(male$Age_Scl), length = M)
N = rep(x1, M)
A = rep(x2, each = M)
lat = data.frame(N = N, A = A, S = 1, Sh = 1)
Ylat =  cbind(rep(x2*0, each = M))

#Fit the lattice to the model and return predictions.
res_fitted = neural_net(as.matrix(lat), Ylat, estimateA, 3, lambdaA)
z = matrix(res_fitted$out, nrow = M, ncol = M)
z
filled.contour(x = seq(min(male$Nutrition), max(male$Nutrition), length = M),
               y = seq(min(male$Age_Scl), max(male$Age_Scl), length = M),
               z,
               color = function(n) hcl.colors(n, "Blue-Yellow"),
               nlevels = 30,
               #zlim = range(min(male$Speed_21km), max(male$Speed_21km)),
               plot.title = title(main = "Avg Speed of Male Nike Athletes",
                                  xlab = "Nutrition", ylab = "Age (scaled)"))



#Define female Nike lattice.
female = subset(dat_train, (dat_train$ShoeBrand=='Nike'))
female = subset(female, female$Sex=='Female')

M = 200
x1 = seq(min(female$Nutrition), max(female$Nutrition), length = M)
x2 = seq(min(female$Age_Scl), max(female$Age_Scl), length = M)
N = rep(x1, M)
A = rep(x2, each = M)
lat = data.frame(N = N, A = A, S = 0, Sh = 1)
Ylat =  cbind(rep(x2*0, each = M))

res_fitted_female = neural_net(as.matrix(lat), Ylat, estimateA, 3, lambdaA)
z = matrix(res_fitted_female$out, nrow = M, ncol = M)

filled.contour(x = seq(min(female$Nutrition), max(female$Nutrition), length = M),
               y = seq(min(female$Age_Scl), max(female$Age_Scl), length = M),
```

```
              z,
              color = function(n) hcl.colors(n, "YlOrBr"),
              nlevels = 30,
              zlim = range(min(female$Speed_21km), max(female$Speed_21km)),
              plot.title = title(main = "Avg Speed of Female Nike Athletes",
                                 xlab = "Nutrition", ylab = "Age (scaled)"))


## f)
#Predictions saved to .csv file.
X_test = matrix(cbind(dat_test$Nutrition, dat_test$Age_Scl, as.numeric(dat_test$Sex)-1, as.numeric(dat_
Y_test = matrix(rep(0, dim(X_test)[1]), ncol = 1)

test_model = neural_net(X_test, Y_test, estimateA, 3, lambdaA)
test_model$out

test_model$E2
pred = data.frame(predictions = matrix(test_model$out, ncol = 1))
write.table(pred,'FLSANT005_STA4026S_CA2.csv', quote = F, row.names = F, sep = ',')
```

Question 2

```
# Let's fake a dataset and see if the network evaluates:
set.seed(2020)
N = 50
x = runif(N,-1,1)
e = rnorm(N,0,1)
y = 2*sin(3*pi*x)+e
plot(y~x, pch = 16, col = 'blue')

# Get the data in matrix form:
X = matrix(x,N,1)
Y = matrix(y,N,1)

# Specify activation functions for the hidden and output layers:
sig1 = function(x)
{
  1/(1+exp(-x))
}
sig1. = function(x)
{
  1/(1+exp(-x))*(1-1/(1+exp(-x)))
}
sig2 = function(x)
{
  x
}
sig2. = function(x)
{
  1+0*x
}

g = function(Yhat,Y)
{
  0.5*(Yhat-Y)^2
}
g. = function(Yhat,Y)
{
  (Yhat-Y)
}
# Write a function that evaluates the neural network (forward recursion):
# X     - Input matrix (N x p)
# Y     - Output matrix(N x q)
# theta - A parameter vector (all of the parameters)
# m     - Number of nodes on hidden layer
# lam   - Regularisation parameter (see later)
neural_net = function(X,Y,theta, m, lam)
{
  # Relevant dimensional variables:
  N = dim(X)[1]
  p = dim(X)[2]
  q = dim(Y)[2]
```

```
  # Populate weight-matrix and bias vectors:
  index = 1:(p*m)
  index
  W1    = matrix(theta[index],p,m)
  W1
  index = max(index) + 1:(m*q)
  W2    = matrix(theta[index],m,q)
  W2
  index = max(index) + 1:m
  b1    = matrix(theta[index],m,1)
  index = max(index) + 1:q
  b2    = matrix(theta[index],q,1)

  # Evaluate network:
  out   = rep(0,N)
  error = rep(0,N)

  dW1 = W1*0
  dW2 = W2*0
  dX1 = X*0
  db1 = b1*0
  db2 = b2*0

  i = 1
  for(i in 1:N)
  {
    a0 = matrix(X[i,],ncol = 1)
    z1 = t(W1)%*%a0+b1
    a1 = sig1(z1)
    z2 = t(W2)%*%a1+b2
    a2 = sig2(z2)

    d2 = g.(a2,Y[i])*sig2.(z2) #working gradient of layer 2 derivative of C * sigma(z2)
    d1 = (W2%*%d2)*sig1.(z1) #working gradient of layer 1
    d0 = (W1%*%d1) #working gradient of layer 0 (inputs)
    db2 = db2+d2
    db1 = db1+d1
    dX1[i] = d0 #assign
    dW2 = dW2+a1%*%t(d2)
    dW1 = dW1+a0%*%t(d1)

    out[i]   = a2
    error[i] = g(a2,Y[i])
  }

  # Calculate error:
  E1 = sum(error)/N
  #E2 = ...

  # Return predictions and error:
  return(list(out = out, E1 = E1, grad = c(dW1,dW2,db1,db2)/N, gradX = dX1))
}
```

```
# We need to know the number of parameters in the network:
m          = 10
p          = dim(X)[2]
q          = dim(Y)[2]
npars      = p*m+m*q+m+q
npars
theta_rand = runif(npars,-2,2)
theta = theta_rand
res        = neural_net(X,Y,theta_rand,m,0)

# Set an objective and minimize
obj = function(pars)
{
  res = neural_net(X,Y,pars,m,0)
  return(res$E1)
}


res_opt = nlm(obj,theta_rand, iterlim = 250)


# res_rc     = neural_net(x_lat,y_dummy,res_opt$estimate,m,0)
#
# # Plot response curve and fitted values
# plot(y~x,pch = 16,col = 'blue')
# points(res_fitted$out~X,col = 'red')
# lines(res_rc$out~x_lat)


#=====================================================
# Continue here:
#=====================================================

### Question 2
## a)
x_lat = cbind(seq(-1,1,1/100))
y_dummy = cbind(x_lat*0)
h     = 0.01
grad_check = c()
for(k in 1:length(x_lat))
{
  x_kp = x_lat
  x_km = x_lat
  x_kp[k] = x_kp[k]+h/2
  x_km[k] = x_km[k]-h/2

  res_kp  =  neural_net(x_kp,y_dummy,res_opt$estimate,m,0)
  res_km  =  neural_net(x_km,y_dummy,res_opt$estimate,m,0)

  grad_check[k] = (res_kp$out[k]-res_km$out[k])/h
}
```

```
#The partial derivative of the cost function with respect to X. So simple!
CpartialX = 6*pi*cos(3*pi*x_lat)

#Plot approximate gradient vs regularly spaced x
res_fitted = neural_net(x_lat, y_dummy, res_opt$estimate, 10, 0)
plot(res_fitted$gradX~x_lat, type = 'l', lwd = 2, col = 'dodgerblue1', xlab = 'X', ylab = 'Gradient', ma
#add actual cost function derivative/gradient
lines(x_lat, CpartialX, lwd = 2, col = 'orange')

lines(grad_check~x_lat, type = 'l', lwd = 2, col = 'darkgreen')
legend(x = 'bottomright', legend=c("Backprop", "True Deriv", "Approximation"),
       fill = c("dodgerblue1","orange", "darkgreen"))
```