(a) Comment each section labelled “SECTION ” , be sure to provide enough information of

what is going on in the section.

SECTION 1 :

mnist\_raw[,-1] = as.data.frame(lapply(mnist\_raw[,-1], min\_max\_normal))

This code performs min-max normalization on all columns in mnist\_raw except for the first column. Finally, use the as.data.frame() function to convert back to a data frame.

mnist\_raw <- replace(mnist\_raw, is.na(mnist\_raw), 0)

This code replaces missing values (NA) in the mnist\_raw data with 0. In neural networks, missing values can cause computational errors, so replacing them with 0 can avoid such problems.

SECTION 2 :

mnist\_raw[, "zero"] = as.numeric(mnist\_raw[,"X5"] == 0)

mnist\_raw[, "one"] = as.numeric(mnist\_raw[,"X5"] == 1)

mnist\_raw[, "two"] = as.numeric(mnist\_raw[,"X5"] == 2)

mnist\_raw[, "three"] = as.numeric(mnist\_raw[,"X5"] == 3)

mnist\_raw[, "four"] = as.numeric(mnist\_raw[,"X5"] == 4)

mnist\_raw[, "five"] = as.numeric(mnist\_raw[,"X5"] == 5)

mnist\_raw[, "seven"] = as.numeric(mnist\_raw[,"X5"] == 7)

mnist\_raw[, "eight"] = as.numeric(mnist\_raw[,"X5"] == 8)

mnist\_raw[, "nine"] = as.numeric(mnist\_raw[,"X5"] == 9)

This code creates several columns by column X5 in mnist\_raw. It translates specific data to a related column( For example: if the data is 0, it would create a new column called “zero” to mark 1, or if the column already exist, it will mark one in that column)

SECTION 3 :

mnist\_raw$actual = mnist\_raw[,1]

This code assigns the value of the first column in the mnist\_raw data set to the new variable actual.

mnist\_raw = as.data.frame(mnist\_raw[,-1])

This procedure removes the first column of the mnist\_raw data box (i.e.,column “actual”) and replaces it with a new data frame.

SECTION 4 :

frm = 'zero + one + two + three + four + five + six + seven + eight + nine ~ '

for (i in 1:784) {

frm = paste(frm, names(mnist\_raw)[i],sep="+")

}

This code creates a formula for Neural Network model that the target is 'zero + one + two + three + four + five + six + seven + eight + nine ,and the variables are based on the mnist\_raw’s columns name.

SECTION 5 :

mnist\_raw\_train = mnist\_raw[1:4000,]

This code creates a new data set based on mnist\_raw, and only pick up first 4000 rows.

SECTION 6 :

neuron\_estimate = 3

This code is to Choose how many Number of Hidden Layers.

SECTION 7 :

nn = neuralnet(frm,data=mnist\_raw\_train[,-795],hidden=c(2),linear.output=F)

This code uses the neuralnet() function to create a neural network model. The parameter frm contains a formula. data specifies the dataset used to train the model, where mnist\_raw\_train[,-795] indicates the removal of the last column. hidden specifies the hidden layer size of the network, here set to c(2), i.e. a hidden layer containing 2 neurons.

predict\_num = compute(nn, covariate = mnist\_raw\_train[,-(785:795)])$net.result

This code uses the trained neural network nn to predict the classification result of each data in the mnist\_raw\_train[,-(785:795)] data set (indicates the removal of the last 11 fields in the mnist\_raw\_train dataset). And compute the output of the neural network. Finally, the prediction results of the neural network are obtained using $net.result and stored in the predict\_num variable.

tmp\_output = cbind(mnist\_raw\_train[1:100,"actual"], round(predict\_num[1:100,],2))

This code stores the predicted and actual values of the first 100 data in the training set mnist\_raw\_train in a table in tmp\_output variables. Only pick up the first 100 rows of the actual column from mnist\_raw\_train and the first 100 rows of predict\_num, then Round the result to 2 decimal places. Finally, use the cbind() function to combine the actual value and the prediction result into a table with columns.

(b) The code needs to work on your machine, so you will need to get it to work.There may be a few errors. Indicate what you needed to do to get the code to work.

1. ~~mnist\_raw[, "zero"] = as.numeric(mnist\_raw[,"X1"] == 0)~~

mnist\_raw[, "zero"] = as.numeric(mnist\_raw[,"X5"] == 0)

mnist\_raw[, "six"] = as.numeric(mnist\_raw[,"X5"] == 6)

I change the target variable’s column, because in my dataset, the target column is “X5”. In addition, I add the new target variable’s column”6”, which is missing.

1. r~~m = 'zero + one + two + three + four + five + six + seven + eight + nine ~ '~~

target = 'zero + one + two + three + four + five + six + seven + eight + nine ~ '

vec = str\_split(target, " ~ ", simplify = TRUE)[[1]]

variables = str\_split(vec, " \\+ ", simplify = TRUE)

target\_t = paste(paste(variables[variables %in% n], collapse = " + "),"~")

I create a formula though target variable based on what “actual” column’s data. If the variable is not appeared in actual column, I will not include it.

1. ~~neuron\_estimate = 3~~

inputs = ncol(mnist\_raw\_train) - 1

outputs = 1

neuron\_estimate = ceiling(nrow(mnist\_raw\_train) / (2 \* (inputs + outputs)))

Instead of indicate a actual number of neuron estimation, I use upper bound equation to check the maximum number.

1. ~~names(tmp\_outpt) = c("Actual", "Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine")~~

tmp\_output = cbind(mnist\_raw\_train[,"actual"], round(predict\_num,2))

tmp\_target = variables[variables %in% n]

tmp\_target = c("actual",tmp\_target)

colnames(tmp\_output) = tmp\_target

tmp\_output = as.data.frame(tmp\_output)

features = tmp\_output[,-1]

numbers = c(0,1,2,3,4,5,6,7,8,9)

for (i in seq\_len(nrow(features))){

max\_col = apply(features[i,], 1, which.max)

tmp\_output[i,"pre"] = numbers[max\_col]

}

tmp\_output = tmp\_output[,c(1,ncol(tmp\_output))]

Instead of rename the column manually, I decide to rename the column based on what “actual” column’s data. And encode it to one column.

(c) Analyze what your colleague did and explain their process (from the technique perspective) and whether the code could be improved. Indicate the number of hidden layers and number of neurons. Do you think this might be a problem, if so why?

First, Colleague normalized the dataset without target variable to make sure this dataset was appropriate to put into the neural network model. In order to increase the accuracy, the colleague divides the target variable into several columns. Because with several outputs, the difference of each predict variable will be more precise.

Second, the colleague creates a formula for the neural network model.

Finally, colleague used 4000 observation for the training set and put it into a neural network model with 2 hidden nodes and 1 hidden layer. Then use this model to predict the training set.

In addition, the number of neurons can not be higher than 2. Because, in order to avoid overfitting, we can should the upper bound by this equation.

(d) Use the table below to provide level of accuracy of your colleagues, code. If it doesn’t not run, make any necessary modifications you feel necessary to get the code to run, while

keeping in line with your colleagues ”approach”.

|  |  |  |
| --- | --- | --- |
| Observations | # Accurate | Accurate |
| 1000 | 605 | 0.687 |
| 1500 | 801 | 0.820 |
| 2000 | 1210 | 0.653 |

(e) Next, modify the code in a way that you think is more efficient and for the same observations,

or even more provide a similar table. Indicate the number of hidden layers, you

choose and the number of hidden nodes in each layer.

I use 2 hidden nodes in my model. If we maximize the accuracy, we should use higher hidden nodes it can. However, in order to avoid overfitting, we can limit the upper bound by this equation.

This is the case. We can write this code to represent it.

neuron\_estimate = ceiling(nrow(mnist\_raw\_train) / (2 \* (inputs + outputs)))

and the result is 2, which means the upper bound of hidden nodes to prevent overfitting is 2 nodes.

|  |  |  |
| --- | --- | --- |
| Observations | # Accurate | Accurate |
| 1000 | 801 | 0.778 |
| 1500 | 1176 | 0.828 |
| 2000 | 1639 | 0.805 |

(f) Provide an interpretative summary of the differences between your final code and your

colleagues code. Provide any explanations about the approach and your thoughts about

the different methods.

First, I add the new column”six”, which is based on actual number and the colleague had missed.

mnist\_raw[, "six"] = as.numeric(mnist\_raw[,"X5"] == 6)

second, We could build a target based on target variables when creating a formula. If there’s a missing target, we should removed it.

target = 'zero + one + two + three + four + five + six + seven + eight + nine ~ '

vec = str\_split(target, " ~ ", simplify = TRUE)[[1]]

variables = str\_split(vec, " \\+ ", simplify = TRUE)

target\_t = paste(paste(variables[variables %in% n], collapse = " + "),"~")

for (i in 1:784) {

target\_t = paste(target\_t, n[i],sep="+")

}

frm = target\_t

Second, for neuron estimate, we should consider whether it will be overfitting by the upper bound equation.

inputs = ncol(mnist\_raw\_train) - 1

outputs = 9

neuron\_estimate = ceiling(nrow(mnist\_raw\_train) / (2 \* (inputs + outputs)))

Finally, using neuronal network to predict data and for measuring the prediction accuracy, instead of showing all the predictions and actual numbers, we can count all correct predictions and divide the total number to measure the prediction accuracy.

start\_time = system.time({

nn = neuralnet(frm,data=mnist\_raw\_train[,-795],

hidden=c(2),

linear.output=F,

stepmax = 1000000,

rep = 1)

})

plot(nn)

tmp\_output = cbind(mnist\_raw\_train[,"actual"], round(predict\_num,2))

tmp\_target = variables[variables %in% n]

tmp\_target = c("actual",tmp\_target)

colnames(tmp\_output) = tmp\_target

# tmp\_output

tmp\_output = as.data.frame(tmp\_output)

features = tmp\_output[,-1]

numbers = c(0,1,2,3,4,5,6,7,8,9)

# features = is.na(features)

for (i in seq\_len(nrow(features))){

max\_col = apply(features[i,], 1, which.max)

tmp\_output[i,"pre"] = numbers[max\_col]

}

tmp\_output = tmp\_output[,c(1,ncol(tmp\_output))]

accuracy = 0

for (i in 1:nrow(tmp\_output)){

if (!is.na(tmp\_output[i,2]) && tmp\_output[i,1] == tmp\_output[i,2]){

accuracy = accuracy + 1

}

}

print(paste("accuracy : ",accuracy/nrow(tmp\_output)))