University of Cape Town

Machine Learning

Assignment 2

Author:
Luke Barnes

Student Number:
BRNLUK005

October 27, 2023

Contents

1 Question 1

A.

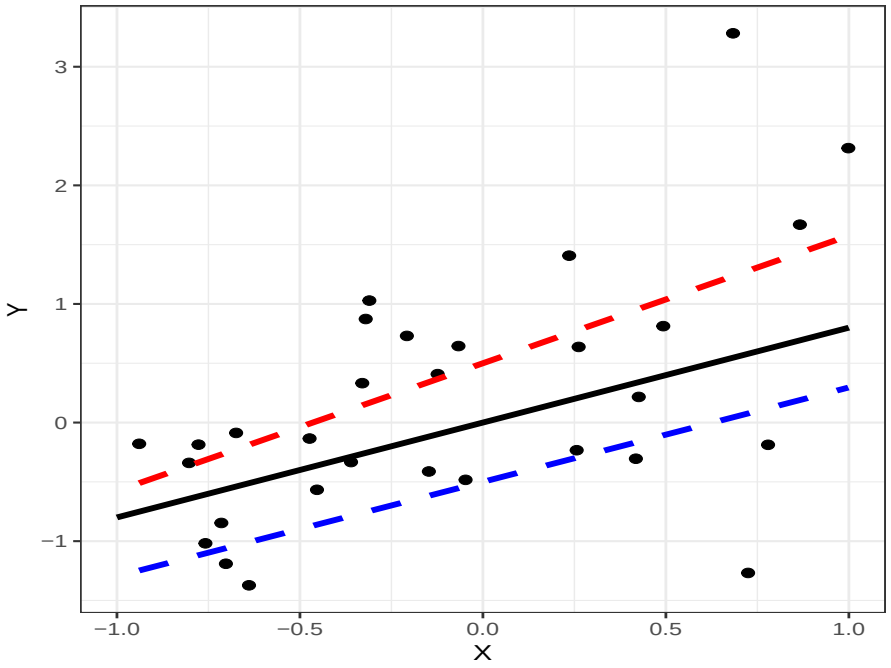

Figure 1: The simulated data (black points), true underlying model (black line) and the fitted models $g_1(x)$ (red dashed line) and $g_2(x)$ (blue dashed line) visualised together.

Figure 1 suggests that the fitted model $g_1(x)$ is going to perform better on this simulated data as it is closer to the majority of the data points. Table 1 shows comparison of the Mean Square Error (MSE) calculated for both fitted models. It shows that $g_1(x)$ has a smaller MSE than $g_2(x)$. However, a comparison of the fitted models to the true underlying model in figure 1 suggests that these models are both incredibly similar to the true underlying model, and the better MSE performance of $g_1(x)$ is a symptom of the simulated standard normal errors and its effect on the data points.

	$g_1(x)$	$g_2(x)$
MSE	0.801	1.318

Table 1: Mean Square Error of Fitted Models.

B.

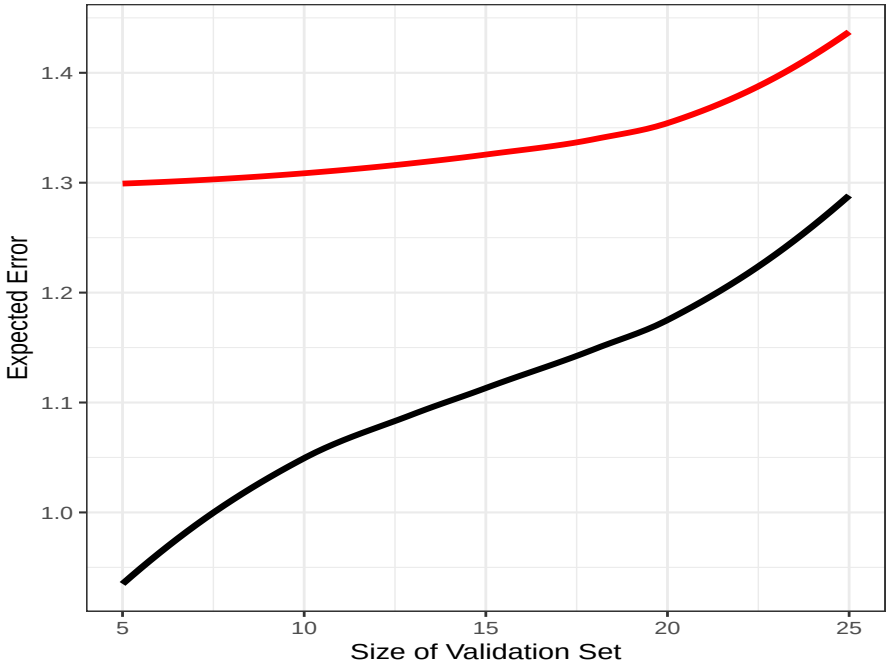

Figure 2: Expected out of sample error, $E_{out}(g^*)$ (red line), and the expected best validation error, $E_{val}(g^*)$ (black line) as the size of the validation data sets increase.

Figure 2 shows the expected out of sample error ($E_{out}(g^*)$) and the expected best validation error ($E_{val}(g^*)$) as the size of the validation data sets increase. As the size of the validation set increases, the expected errors increase. The reason for this is that as the size of the validation set increases, the size of the training set decreases. Models that are trained on larger training sets should better predict on unseen data. Therefore, even the best model, $g^*(x)$ is trained on less data and is a worse predictor of the response, in comparison to the models with larger training sets and smaller validation sets. As the size of the validation set becomes really large, the expected best validation error converges towards $E_{out}(g^*)$. The reason for this is that after a certain split, there are too few observations in the training set to train the model and as such the models, and subsequently $g^*(x)$, become worse at generalising on new data. With fewer training observations, the range for the validation errors and consequently $E_{val}(g^*)$, is smaller. As the validation errors are unbiased estimates of the out of sample error, when the range of validation errors decrease, the best validation error, $E_{val}(g^*)$, is closer to the expected out of sample error $E_{out}(g^*)$ and converges to it..

2 Question 2

A.

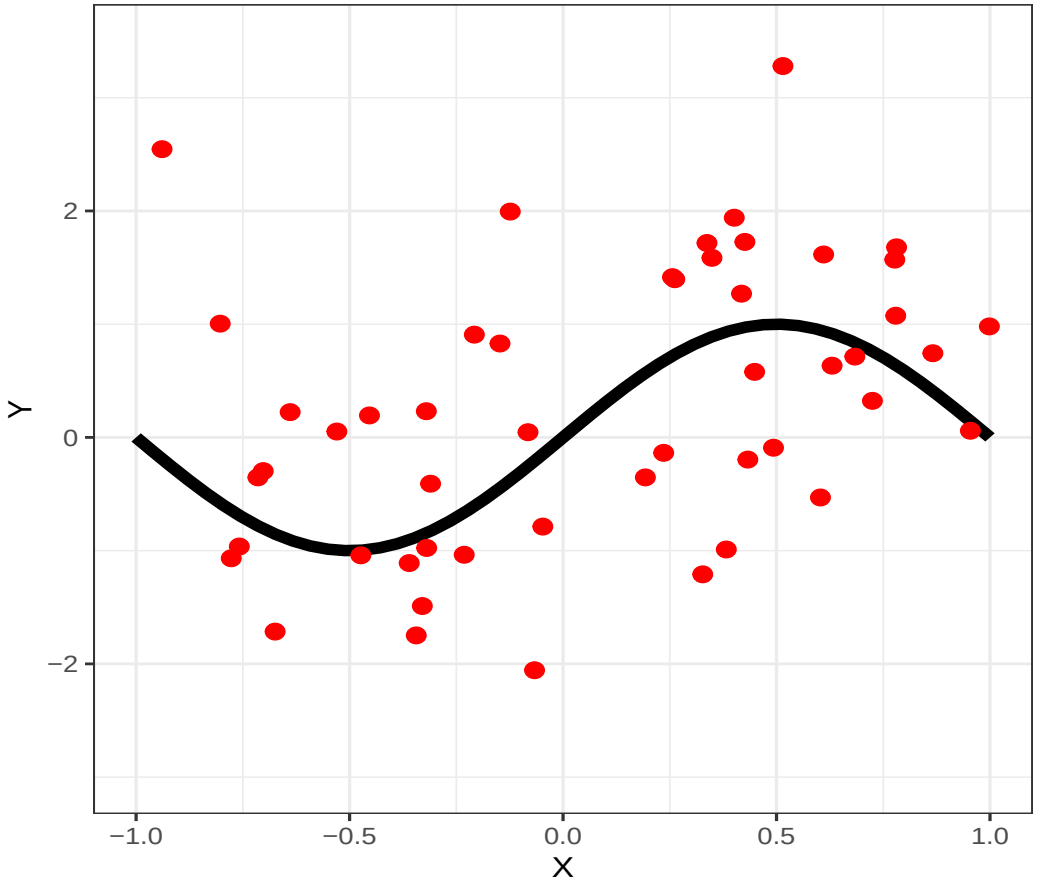

Figure 3: Simulated data (red points) and the underlying model (black line).

B.

The fitted curves in Figure 6 suggest that both the unregularised and regularised legendre models find the underlying pattern in the data. The curves lay close to that of the underlying model. The unregularised model is less smooth, in comparison to the regularised model, as it attempts to fit the curve through as many points as possible. This is additionally seen for one of the smallest x values. The unregularised curve starts at exactly that point. In contrast, the regularised model's fitted curve is more smooth and flatter, as expected.

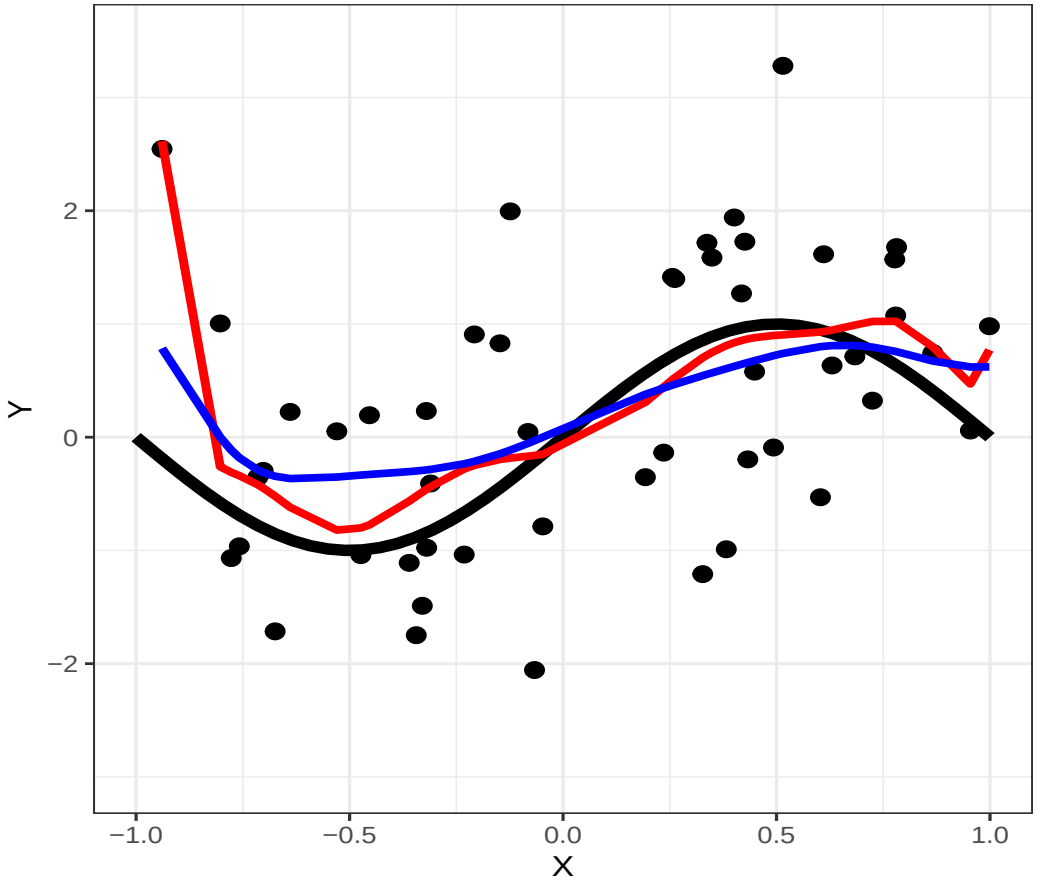

Figure 4: Simulated data (black points) and the underlying model (black line) together with the fitted legendre models. The red line indicates the unregularised legendre model. The blue line represents the regularised legendre model (with $\lambda = 5$).

C.

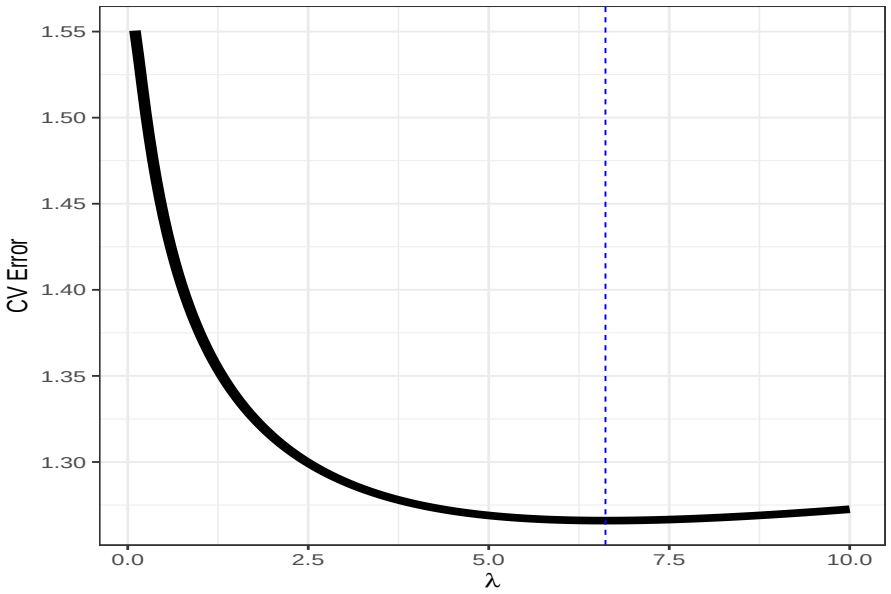

Figure 5: Estimated Cross Validation error as regularisation parameter λ increases from 0.1 to 10. The blue dashed line indicates the optimal λ (6.62) that minimises the cross validation error.

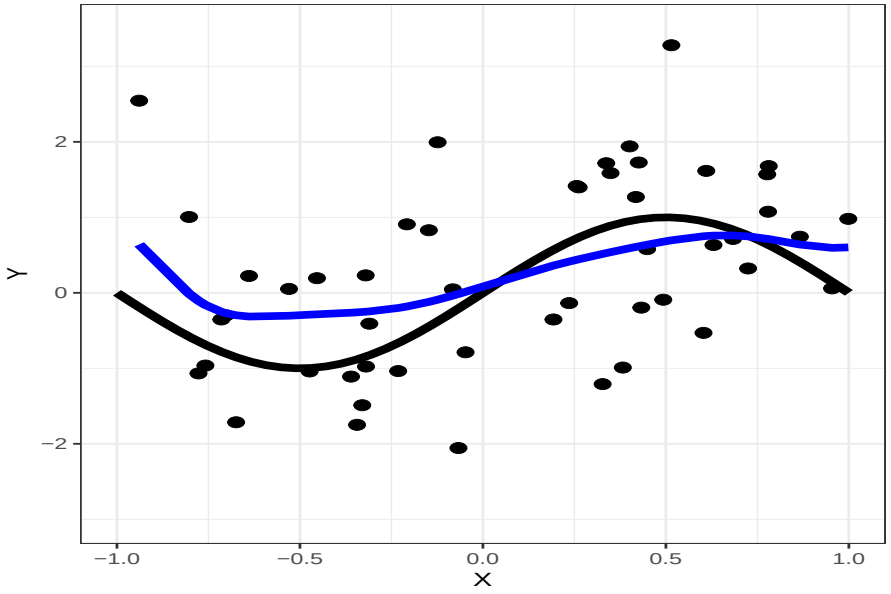

Figure 6: Simulated data (black points) and the underlying model (black line) together with the fitted regularised legendre model with the optimal λ (blue line).

3 Question 3

A.

B.

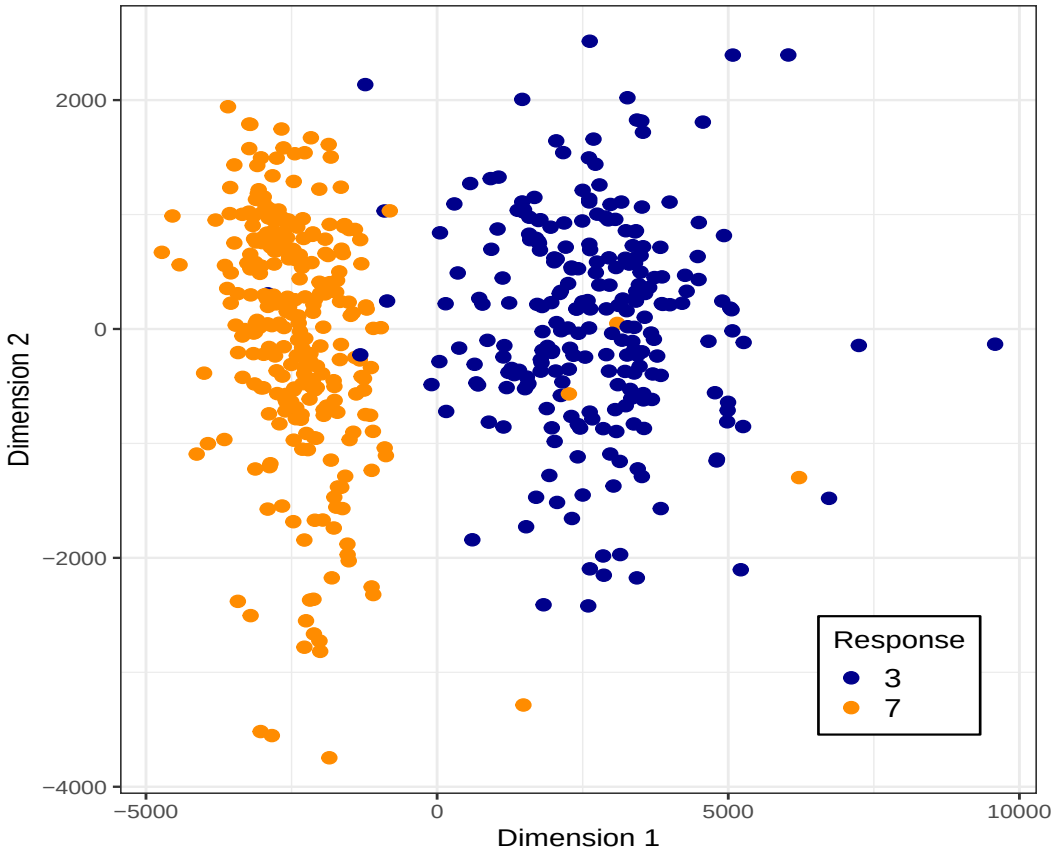

Figure 7: Two-dimensional embedding calculated using K = 35 for the IsoMap. Each cooerdinate is colour coded according to the observation's corresponding number (3's in blue and 7's in orange).

Figure 7 shows a two-dimensional embedding of the observations calculated using the IsoMap algorithm. The observations are colour coded according to their actual number. The two-dimensional embedding suggests that 3's and 7's are indeed separable in a lower dimensional space. In fact, the figure highlights that the numbers could be identified in a one-dimensional space. The numbers are linearly separated in the first dimension. Outside of a few values, the 3's are positive in the first dimension and on the right. In contrast, the 7's are negative in the first dimension and on the left. The two dimensional embedding could be used by any classification algorithm to accurately classify these numbers into their class.

4 Question 4

A.

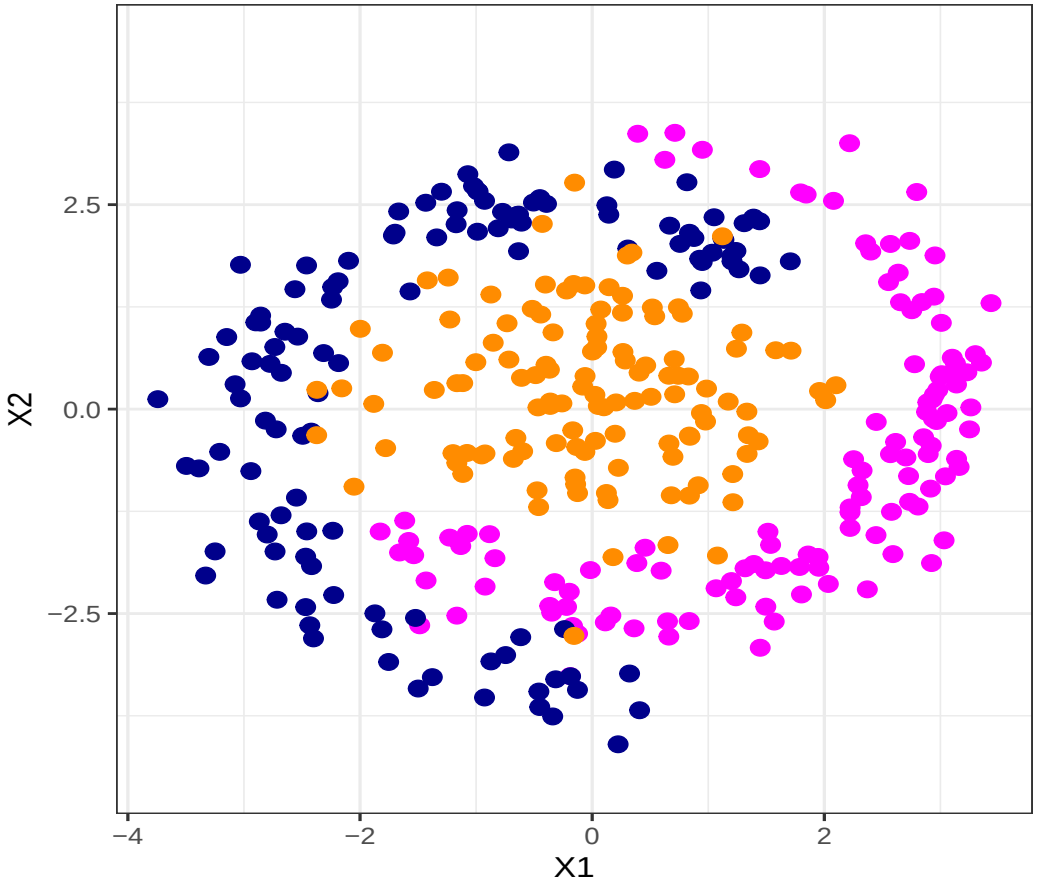

Figure 8: Colour coded observations in the feature space (Code-β *particles* in blue, Code-ρ *particles* in orange and Code-α *particles* in magenta).

B.

C.

There are numerical pitfalls to consider in R when evaluating the multi-class objective function. If the objective function is implemented as given, with a summation over all combinations of $y_{ij}log(\hat{y_{ij}})$, it is possible to end up with an instance where $y_{ij} = 0$ and $\hat{y_{ij}} = 0$, with the resultant calculation producing NaNs in R.

D.

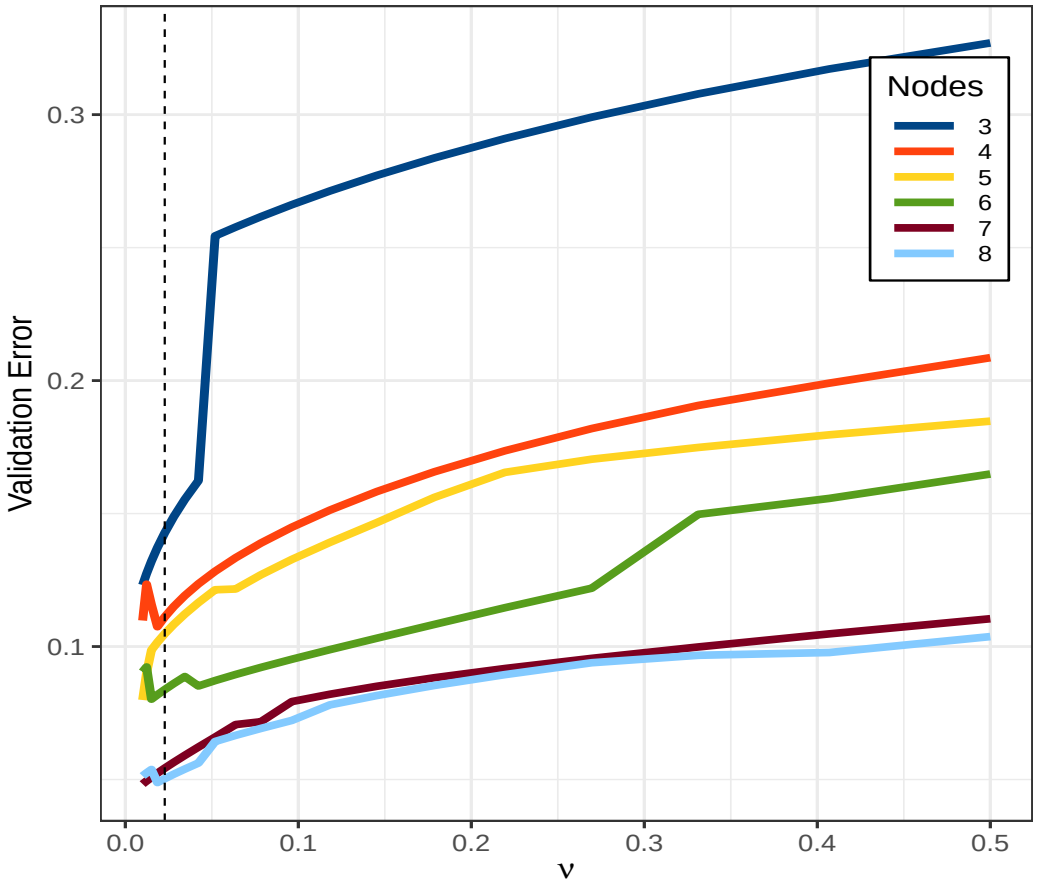

Figure 9: Estimated Validation Error for configurations of the neural network with different numbers of nodes on the hidden layer as the regularisation parameter (ν) increases from 0.01 to 0.5. The colour configuration is as follows: three hidden layer nodes in blue, four nodes in red, five nodes in yellow, six nodes in green, seven nodes in brown and eight hidden layer nodes in cyan. The black dashed line highlights the value of ν (0.023) that minimises the validation error across all configurations of the neural network.

Figure 9 shows the result of the validation analysis conducted to find the optimal regularisation parameter and optimal number of hidden layer nodes for the neural network model. The neural network with eight nodes on the hidden layer and 0.023 as the regularisation parameter, minimises the validation error. These parameters are chosen for the best regularised model used in Q4.E.

E.

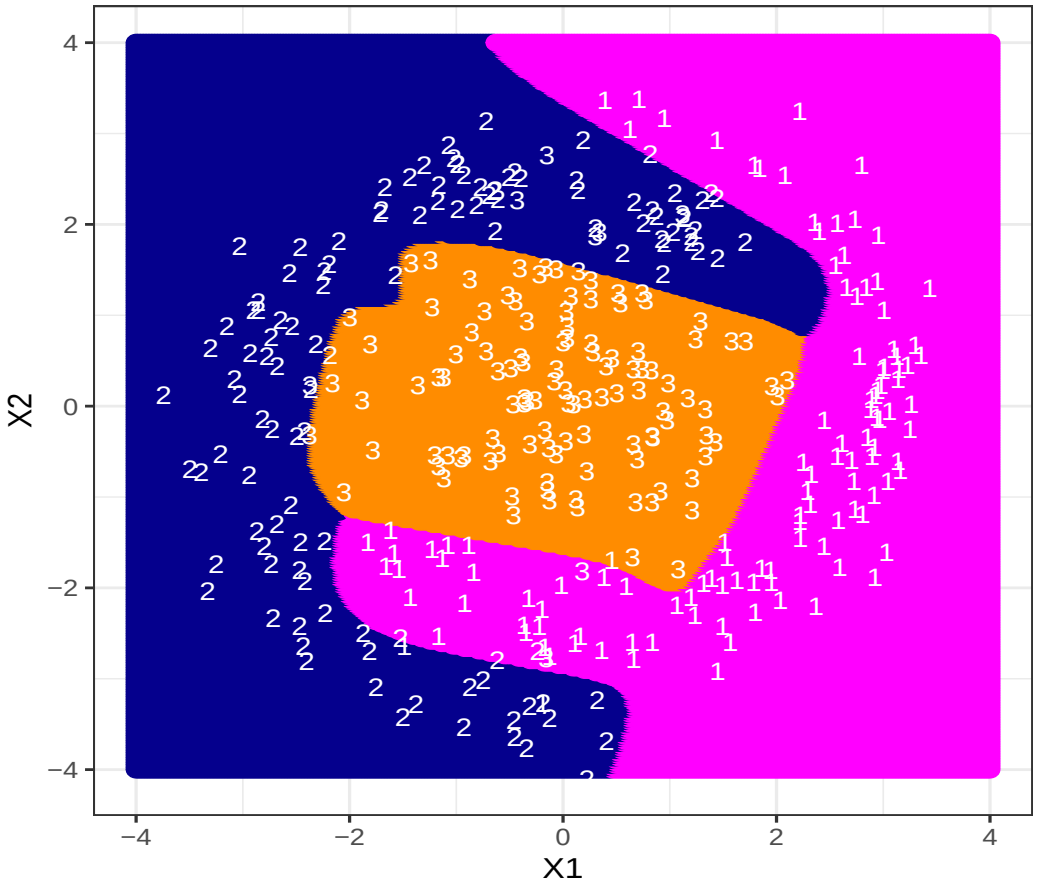

Figure 10: Response Curve for best regularised neural network model with the initial observations overlaid in white (1: Code-α particles, 2:Code-β particles, 3: Code-ρ particles). The magenta region is where the model estimates Code-α *particles* are found in the cross region. The blue region is where the model estimates Code-β *particles* are found. The orange region is where the model estimates Code-ρ *particles* are found.

Figure 10 shows the response curve for the regularised neural network. It highlights the regions in the cross section where the model estimates each particle type can be found. A handful of observations would be misclassified using the model, as seen with overlaid observations on the response curve. Finally, the response curve itself can be used to classify new unseen observations, given their coordinates in the cross section. Particles with coordinates in the magenta region, can be classified as Code-α *particles*. Similarly, particles with coordinates in the blue region, can be classified as Code-β *particles*. Lastly, particles with coordinates in the orange region, should be classified as Code-ρ *particles*.

5 Appendix

5.1 R Code

5.1.1 Question 1

```r
### Libraries (if necessary)
library(tidyverse)
library(tictoc)
library(pracma)

set.seed(2023)

##### Question 1.A

## True Underlying Model
N = 30
x = runif(N, -1, 1)
e = rnorm(N, 0, 1)
y = 0.8 * x + e

xVal = seq(-1, 1, length = N)
yVal = 0.8 * xVal

## Fitted Models

g1.Mod = lm(y ~ 0 + x, offset = rep(0.5, N))

g2.Mod = lm(y ~ 0 + x, offset = rep(-0.5, N))

## Plot true model and overlay fitted models
colVec = c("blue", "red", "black")
lmPlotData = data.frame("X" = x,
                        "Y" = y,
                        "undX" = xVal,
                        "undY" = yVal,
                        "G1" = g1.Mod$fitted.values,
                        "G2" = g2.Mod$fitted.values,
                        "Color" = colVec)

pdf("Figures/lmPlot.pdf")
ggplot(lmPlotData, aes(x = undX)) +
  geom_point(aes(x = X, y = Y), color = "black", size = 3) +
  geom_line(aes(y = undY), color = "black", linewidth = 2, linetype =1) +
  geom_line(aes(x = X, y = G1), color = "red", linewidth = 2, linetype = 2) +
  geom_line(aes(x = X, y = G2), color = "blue", linewidth = 2, linetype = 2) +
  theme_bw(base_size = 16) +
  labs(x = "X", y = "Y")
dev.off()

## Expected Performance using R^2
```

```
MSE = function(N, yhat, y){
  (1/N)*sum((yhat - y)^2)
}

summary(g1.Mod)
summary(g2.Mod)

G1.MSE = MSE(N, lmPlotData$G1, y)
G2.MSE = MSE(N, lmPlotData$G2, y)

##### Question 1.B

dataSize = 10000
setSize = 5:25

matG.star.out = matrix(0, nrow = 21, ncol = dataSize)
matG.star.val = matrix(0, nrow = 21, ncol = dataSize)

f = function(x){
  eps = rnorm(length(x), 0, 1)
  y = 0.8 * x + eps
  return(y)
}

outIntegral = function(x, g1, beta){

  if (g1 == TRUE){
    g = 0.5 + (beta * x)
  }

  else {
    g = -0.5 + (beta * x)
  }

  f = f(x)

  int = (g - f)^2 * 0.5
  return(int)
}

a = -1
b = 1
n = 100000

for(i in 1:dataSize){

  xi = runif(N, -1, 1)
  yi = f(xi)

  tic()
```

```
for (j in 1:length(setSize)){

  sizeS = setSize[j]
  sizeTrain = N - sizeS

  samp = sort(sample(1:N, sizeS, replace = FALSE))

  valData = xi[samp]
  trainData = xi[-samp]

  yVal = yi[samp]
  yTrain = yi[-samp]

  g1.Mod.i = lm(yTrain ~ 0 + trainData, offset = rep(0.5, sizeTrain))
  b1 = as.numeric(g1.Mod.i$coefficients)

  g2.Mod.i = lm(yTrain ~ 0 + trainData, offset = rep(-0.5, sizeTrain))
  b2 = as.numeric(g2.Mod.i$coefficients)

  g1.valPred = 0.5 + (b1 * valData)
  g2.valPred = -0.5 + (b2 * valData)

  G1.MSE.i = MSE(sizeS, g1.valPred, yVal)
  G2.MSE.i = MSE(sizeS, g2.valPred, yVal)

  if(G1.MSE.i < G2.MSE.i){

    matG.star.val[j, i] = G1.MSE.i

    xsam = runif(n, a, b)
    int = (b-a) * mean(outIntegral(xsam, TRUE, b1))
    matG.star.out[j, i] = int

    # matG.star.out[j, i] = integrate(outIntegral, lower = -1,
                                      upper = 1, TRUE, b1)$value
    # matG.star.out[j, i] = quadl(outIntegral, xa = -1, xb = 1,
                                  g1 = TRUE, beta = b1)
  }

  else {

    matG.star.val[j, i] = G2.MSE.i

    xsam = runif(n, a, b)
    int = (b-a) * mean(outIntegral(xsam, FALSE, b2))
    matG.star.out[j, i] = int

    # matG.star.out[j, i] = integrate(outIntegral, lower = -1,
                                      upper = 1, FALSE, b2)$value
    # matG.star.out[j, i] = quadl(outIntegral, xa = -1, xb = 1,
```

```
                                    g1 = FALSE, beta = b2)

    }
  }

  toc()
}

err.Gstar.out = rowMeans(matG.star.out)
err.Gstar.val = rowMeans(matG.star.val)

errorPlotData = data.frame("SetSize" = setSize,
                           "ValExp" = err.Gstar.val,
                           "OutExp" = err.Gstar.out)

pdf("Figures/errorPlot.pdf")
ggplot(errorPlotData, aes(x = SetSize)) +
  geom_smooth(aes(y = ValExp), color = "black",
              linewidth = 2, linetype = 1, se = F) +
  # geom_line(aes(y = ValExp), color = "black", linewidth = 2, linetype = 1) +
  geom_smooth(aes(y = OutExp), color = "red",
              linewidth = 2, linetype = 1, se = F) +
  # geom_line(aes(y = OutExp), color = "red", linewidth = 2, linetype = 1) +
  xlab("Size of Validation Set") + ylab("Expected Error") +
  theme_bw(base_size = 16)
dev.off()
```

5.1.2 Question 2

```
library(tidyverse)
library(pracma)

set.seed(2023)

##### Question 2.A

## True Underlying Model
N = 50
x = runif(N, -1, 1)
e = rnorm(N, 0, 1)
y = sin(pi * x) + e

xVal = seq(-1, 1, length = N)
yVal = sin(pi * xVal)

lmPlotData = data.frame("X" = x,
                        "Y" = y,
                        "undX" = xVal,
                        "undY" = yVal)

pdf("Figures/modPlot_Q2.pdf")
ggplot(lmPlotData, aes(x = undX)) +
  geom_line(aes(y = undY), color = "black", linewidth = 3, linetype = 1) +
  geom_point(aes(x = X, y = Y), color = "red", size = 4) +
  labs(x = "X", y = "Y") +
  ylim(-3, 3.5) +
  theme_bw(base_size = 16)
dev.off()


##### Question 2.B

legFunc = function(x, q){

  legFuncSum = 0

  for (k in 0:q){

    legSum = (x)^k * (choose(q, k)) * (choose((q+k-1)/2, q))

    legFuncSum = legFuncSum + legSum
  }

  return(2^q * legFuncSum)
}

legPolyMat = matrix(0, N, 11)
```

```
for (i in 0:10){

  legPolyMat[, i + 1] = legFunc(x, i)
}

lambda = c(0, 5)
ones = diag(dim(legPolyMat)[2])

legPred = matrix(0, nrow = N, ncol = 2)

betaMat = matrix(0, nrow = dim(legPolyMat)[2], ncol = 2)

for (b in 1:2){
  betaMat[, b] = (solve(t(legPolyMat) %*% legPolyMat +
                          (lambda[b] * ones)) %*% t(legPolyMat) %*% y)

  legPred[, b] = legPolyMat %*% betaMat[, b]
}

legPlotData = data.frame("X" = x,
                         "Y" = y,
                         "undX" = xVal,
                         "undY" = yVal,
                         "leg1" = legPred[, 1],
                         "leg2" = legPred[, 2])

pdf("Figures/legPlot_Q2.pdf")
ggplot(legPlotData) +
  geom_line(aes(x = undX, y = undY), color = "black",
                linewidth = 3, linetype = 1) +
  geom_point(aes(x = X, y = Y), color = "black", size = 4) +
  geom_line(aes(x = X, y = leg1), color = "red",
                linewidth = 2, linetype = 1) +
  geom_line(aes(x = X, y = leg2), color = "blue",
                linewidth = 2, linetype = 1) +
  labs(x = "X", y = "Y") +
  ylim(-3, 3.5) +
  theme_bw(base_size = 16)
dev.off()

##### Question 2.C

kFoldSeq = seq(5, 50, by = 5)

#kFoldMat = matrix(0, nrow = 5, ncol = 10)
kFoldMat = list()
valPredMat = matrix(0, nrow = 5, ncol = 10)
#trainSet = matrix(0, nrow = 45, ncol = 10)
trainSet = list()
trainPredMat = matrix(0, nrow = 45, ncol = 10)
```

```
for (k in 1:10){
  ind = (kFoldSeq[k]-4):kFoldSeq[k]
  kFoldMat[[k]] = legPolyMat[ind,]
  valPredMat[, k] = y[ind]
  trainSet[[k]] = legPolyMat[-ind,]
  trainPredMat[, k] = y[-ind]
}

lambdas = seq(0.1, 10, length = 200)
cvError = c()

for (c in 1:length(lambdas)){

  avgErr = c()

  for(d in 1:10){

    betaMod = (solve(t(trainSet[[d]]) %*% trainSet[[d]] +
                       (lambdas[c] * ones)) %*%
                       t(trainSet[[d]]) %*% trainPredMat[, d])

    valPred = kFoldMat[[d]] %*% betaMod

    avgErr[d] = mean((valPred - valPredMat[, d])^2)
  }

  cvError[c] = mean(avgErr)
}

cvPlotData = data.frame("Lambdas" = lambdas,
                        "CVError" = cvError)

pdf("Figures/cvPlot_Q2.pdf")
ggplot(cvPlotData) +
  geom_line(aes(x = Lambdas, y = CVError), color = "black",
            linewidth = 3, linetype = 1) +
  geom_vline(xintercept = lambdas[which.min(cvError)],
                linewidth = 0.5, linetype = 2, col = "blue") +
  labs(x = expression(lambda), y = "CV Error") +
  theme_bw(base_size = 16)
dev.off()

lambdaFit = lambdas[which.min(cvError)]

betaFitMod = (solve(t(legPolyMat) %*% legPolyMat +
                      (lambdaFit * ones)) %*% t(legPolyMat) %*% y)

legFitPred = legPolyMat %*% betaFitMod


legFitPlotData = data.frame("s0" = legFitPred,
```

```
                              "X" = x,
                              "Y" = y,
                              "undX" = xVal,
                              "undY" = yVal)

pdf("Figures/legFitPlot_Q2.pdf")
ggplot(legFitPlotData) +
  geom_line(aes(x = undX, y = undY), color = "black",
                linewidth = 3, linetype = 1) +
  geom_point(aes(x = X, y = Y), color = "black", size = 4) +
  geom_line(aes(x = X, y = s0), color = "blue",
                linewidth = 3, linetype = 1) +
  labs(x = "X", y = "Y") +
  ylim(-3, 3.5) +
  theme_bw(base_size = 16)
dev.off()
```

5.1.3 Question 3

```
### Libraries (if necessary)
library(tidyverse)
library(pracma)
library(e1071)

set.seed(2023)

##### Question 3.A

#=======================================================

dat    = read.csv('Digits2020.csv')

Y  =  dat[, 1]
X  =  as.matrix(dat[, -1])

image(matrix(X[1, ], 28, 28)) # Digit 7
image(matrix(X[2, ], 28, 28)) # Digit 3

# A function for finding K nearest neighbours:

k_nn = function(X1, X2, k = 10){

  N1 = dim(X1)[1]
  N2 = dim(X2)[1]
  d  = dim(X1)[2]
  ones  = matrix(1,N1,1)
  inds  = matrix(0,N2,k)
  edges = inds
  for(i in 1:N2)
  {
    dists     = sqrt(rowSums((ones%*%X2[i,] -X1)^2))
    wh        = order(dists)[2:(k+1)]
    inds[i,]  = wh
    edges[i,] =dists[wh]
  }
  return(list(edges = edges, neighbours = inds, k = k))
}

# Calculate K nearest neighbours:

K   = 35
res = k_nn(X, X, K)

IsoMap = function(res, d){

  ## Your Work Here...

  N    = dim(res$edges)[1]
```

```
  Dg   = matrix(Inf, N, N)

  for(i in 1:N){

    Dg[i, res$neighbours[i, ]] = res$edges[i, ]
  }

  diag(Dg) = 0
  asp = allShortestPaths(Dg)

  Dg = asp$length

  # Construct the d-dimensional embedding

  H = diag(N) - (1/N * matrix(1, N, N))

  tau = -0.5 * H %*% (Dg^2) %*% H
  sv = svd(tau)

  y1 = sv$d[1] * sv$v[, 1]
  y2 = sv$d[2] * sv$v[, 2]

  # Return a list with the embedding in U:
  return(list(D1 = y1, D2 = y2, Dg = Dg, d = d))
}

# Q3.B
# Plot the embedding

isoMod = IsoMap(res, 2)

colours = c("darkblue", "darkorange")

isoData = data.frame("D1" = isoMod$D1,
                     "D2" = isoMod$D2,
                     "Response" = as.factor(Y))

pdf("Figures/embedPlot_Q3.pdf")
ggplot(isoData, aes(x = D1, y = D2, col = Response)) +
  geom_point(size = 3) +
  xlab("Dimension 1") +
  ylab("Dimension 2") +
  theme_bw(base_size = 14) +
  scale_color_manual(values = c("3" = "darkblue",
                                "7" = "darkorange")) +
  theme(legend.text = element_text(size = 15),
        legend.position = c(0.85, 0.15),
        legend.background = element_rect(fill = "white",
                                         linewidth = 0.6,
                                         linetype = "solid",
                                         colour = "black"))
```

```
dev.off()
```

5.1.4 Question 4

```
### Libraries (if necessary)
library(tidyverse)
library(pracma)
library(colorspace)
library(paletteer)

### Data

set.seed(2023)

dat = read.table("Collider_Data_2022 2.txt",
                 h = TRUE, stringsAsFactors = TRUE)
dim(dat)

### creating continuous colour palette

colours = c("magenta", "darkblue", "darkorange")
color.gradient = function(x, colors = colours, colsteps = 50){

  colpal = colorRampPalette(colors)
  return( colpal(colsteps)[ findInterval(x, seq(min(x),
                            max(x), length = colsteps)) ] )
}

##### Question 4.A
### Plot the coordinates in the feature space and
### colour-code according to the response.

var = dat %>%
  gather(Response, flag, Yi1:Yi3) %>%
  filter(flag == 1) %>%
  select(-flag) %>%
  mutate(Response = as.factor(Response))

clPlotData = data.frame("X1" = dat$X1,
                        "X2" = dat$X2)

pdf("Figures/dataPlot_Q4.pdf")
ggplot(clPlotData, aes(x = X1)) +
  geom_point(aes(x = X1, y = X2),
                color = colours[var$Response], size = 4) +
  labs(x = "X1", y = "X2") +
  ylim(-4.5, 4.5) +
  theme_bw(base_size = 16)
dev.off()


##### Question 4.B
```

```
### Write an R-function which evaluates the soft-max activation function in
### matrix form.

### Z is 3 x N

softMaxMat = function(z){

  expZ = exp(z)
  sumZ = colSums(expZ)
  matSumZ = rbind(sumZ, sumZ, sumZ)

  sM = expZ / matSumZ

  rownames(sM) = c("J1", "J2", "J3")

  return(sM)
  #return(list(expZ = expZ, sumZ = sumZ, sM = sM))
}


##### Question 4.C

## Hidden Layer

sig1 = function(z){

  tanh(z)
}

## Feed Forward Neural Network

neuralNet = function(X, Y, theta, m, nu){

  N = dim(X)[1]
  p = dim(X)[2]
  q = dim(Y)[2]

  dims = c(p, m, q)

  index = 1 : (dims[1]*dims[2])
  W1    = matrix(theta[index], dims[1], dims[2])

  index = max(index) + 1 : (dims[2]*dims[3])
  W2    = matrix(theta[index], dims[2], dims[3])

  index = max(index) + 1 : (dims[2])
  b1    = matrix(theta[index], dims[2], 1)

  index = max(index) + 1 : (dims[3])
  b2    = matrix(theta[index], dims[3], 1)
```

```
  ones = matrix(1, 1, N)

  A0 = t(X)
  A1 = sig1(t(W1)%*%A0 + b1%*%ones)
  A2 = softMaxMat(t(W2)%*%A1+b2%*%ones)

  yHat = t(A2)

  error = Y * 0
  ind1 = which(Y == 1, arr.ind = TRUE)
  error[ind1] = log(yHat[ind1])

  E1 = - (1/N) * sum(error)
  E2 = E1 + ((nu/2) * (sum(W1^2) + sum(W2^2))/N)

  return(list(A2 = A2, A1 = A1, E1 = E1, E2 = E2))
}

X = as.matrix(dat[, 1:2])
Y = as.matrix(dat[, 3:5])

nu  = 0
# m   = 360/90
m = 9

p = dim(X)[2]
q = dim(Y)[2]
npars = p*m+m*q+m+q
thetaRand = runif(npars, -1, 1)

obj = function(pars){

  resModel = neuralNet(X, Y, pars, m, nu)
  return(resModel$E1)
}

obj(thetaRand)

# Fit the neural network using a standard optimizer in R:

resOpt = nlm(obj, thetaRand, iterlim = 1000)
resOpt

### Validation Analysis

set.seed(2022)

N = dim(X)[1]
set = sample(1:N, 0.5*N, replace = FALSE)

XTrain = as.matrix(X[set, ])
```

```
YTrain = as.matrix(Y[set, ])
XVal = as.matrix(X[-set, ])
YVal = as.matrix(Y[-set, ])

nu = 0.5

objPen = function(pars){

  resMod = neuralNet(X, Y, pars, m, nu)
  return(resMod$E2)
}

objPen(thetaRand)

M = 20
#nus = exp(seq(-11.51293, -2.302585, length = M))
#nus = exp(seq(-8, -2.3, length = M))
nus = exp(seq(log(1e-2), log(5e-1), length = M))
ms = 3:8

valErr = matrix(0, length(ms), M)
inErr = valErr

for (j in 1:length(ms)){

  m = ms[j]

  for (i in 1:M) {

      nu = nus[i]
      resOpt = nlm(objPen, thetaRand, iterlim = 1000)

      resIn = neuralNet(XTrain, YTrain, resOpt$estimate, m, 0)
      inErr[j, i] = resIn$E1

      resVal = neuralNet(XVal, YVal, resOpt$estimate, m, 0)
      valErr[j, i] = resVal$E1

      print(paste0("Validation Run | m = ", m, " ,", i, "| nu =", round(nu, 4)))
      print(paste0("In Error: ", inErr[j, i]))
      print(paste0("Val Error: ", valErr[j, i]))

  }

}

valPlotData = data.frame("ValErr3" = valErr[1, ],
                         "ValErr4" = valErr[2, ],
                         "ValErr5" = valErr[3, ],
                         "ValErr6" = valErr[4, ],
                         "ValErr7" = valErr[5, ],
```

```
                          "ValErr8" = valErr[6, ],
                          "Nus" = nus)

valPlotData = valPlotData %>%
  pivot_longer(col = starts_with("ValErr"),
               names_to = "ValErr", names_prefix = "ValErr") %>%
  rename("Nodes" = "ValErr")

pdf("Figures/valPlot_Q4.pdf")
ggplot(valPlotData, aes(x = Nus, y = value, col = Nodes)) +
  geom_line(linewidth = 2) +
  geom_vline(xintercept = nus[which.min(valErr)],
               linewidth = 0.5, linetype = 2, col = "black") +
  xlab(expression(nu)) +
  ylab("Validation Error") +
  scale_color_paletteer_d("ggthemes::calc") +
  theme_bw(base_size = 16) +
  theme(legend.text = element_text(size = 12),
        legend.key.height = unit(0.5, 'cm'),
        legend.key.width= unit(1.25, 'cm'),
        legend.position = c(0.9, 0.8),
        legend.background = element_rect(fill = "white",
                                         linewidth = 0.6,
                                         linetype = "solid",
                                         colour = "black"))
dev.off()


##### Question 4.C

## Regularised Neural Network and corresponding response curve

m = 8
nu = nus[which.min(valErr)]
optRes = nlm(objPen, thetaRand, iterlim = 1000)

M  = 300
x1 = seq(-4, 4,length = M)
x2 = seq(-4, 4,length = M)
xx1 = rep(x1, M)
xx2 = rep(x2, each = M)

XX = cbind(xx1, xx2)
YY = matrix(1, M^2, 3)

regMod = neuralNet(XX, YY, optRes$estimate, m, nu)

predY = round(t(regMod$A2))

xxPlotData = data.frame("X1" = xx1,
                        "X2" = xx2,
```

```
                        "Y1" = predY[, 1],
                        "Y2" = predY[, 2],
                        "Y3" = predY[, 3])

yyVar = xxPlotData %>%
  mutate(Response1 = Y1) %>%
  mutate(Response2 = recode(Y2, "1" = 2)) %>%
  mutate(Response3 = recode(Y3, "1" = 3)) %>%
  pivot_longer(col = starts_with("Response"),
               names_to = "Response", names_prefix = "Response") %>%
  filter(value == 1 | value == 2 | value == 3) %>%
  mutate(Response = value) %>%
  select(X1, X2, Response)

oldData = var %>%
  mutate(numResponse = recode(Response, "Yi1" = 1, "Yi2" = 2, "Yi3" = 3))

pdf("Figures/responsePlot_Q4.pdf")
ggplot(yyVar, aes(x = X1)) +
  geom_point(aes(x = X1, y = X2),
                    color = color.gradient(yyVar$Response), size = 4) +
  geom_text(data = oldData, aes(x = X1, y = X2, label = numResponse),
            size = 5, col = "white") +
  labs(x = "X1", y = "X2") +
  ylim(-4.1, 4) +
  theme_bw(base_size = 16)
dev.off()
```

References

Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin. *Learning from data*, volume 4. AMLBook New York, 2012.

L. Barnes. Machine learning a2. `https://github.com/lukebarnes-ct/MachineLearning_A2`, 2023.