University of Cape Town

Machine Learning

Assignment 1

Author:
Luke Barnes

Student Number:
BRNLUK005

October 27, 2023

Contents

1 Question 1

i.

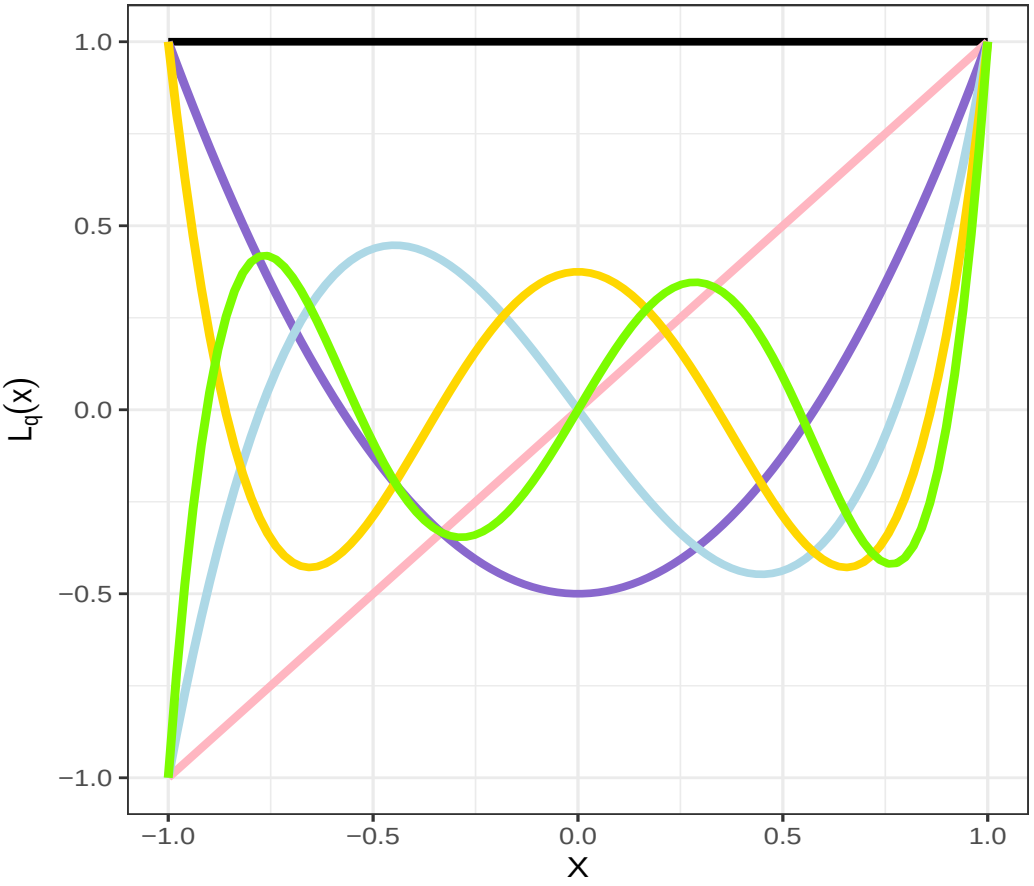

Figure 1: Legendre Polynomials of order 0 (black), 1 (pink), 2 (purple), 3 (blue), 4 (yellow) and 5 (green) over the interval [-1, 1].

ii.

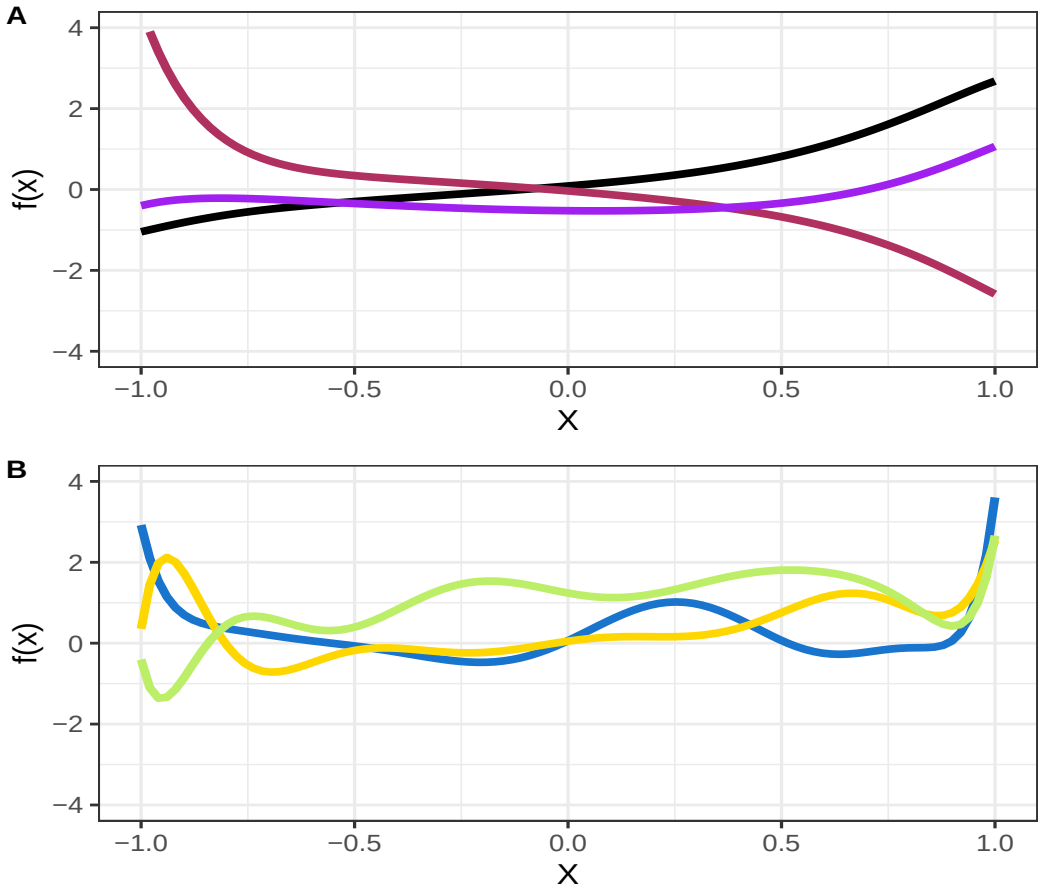

Figure 2: 6 random 10-th order polynomials plotted over the interval [-1, 1]. The functions in Plot A are standard 10-th order polynomials with random uniform coefficients. The functions in Plot B are 10-th order Legendre polynomials with random uniform coefficients.

Figure 2 shows the 6 generated 10-th order polynomials with coefficients randomly sampled from a uniform distribution. The differing behaviour between the two types of 10-th order polynomials over the interval is distinct. The standard 10-th order polynomials are flatter over the interval. Each of them curve distinctly at possibly a singular point. In contrast, the 10-th order Legendre polynomials have a lot more curves at more points over the interval.

2 Question 2

i.

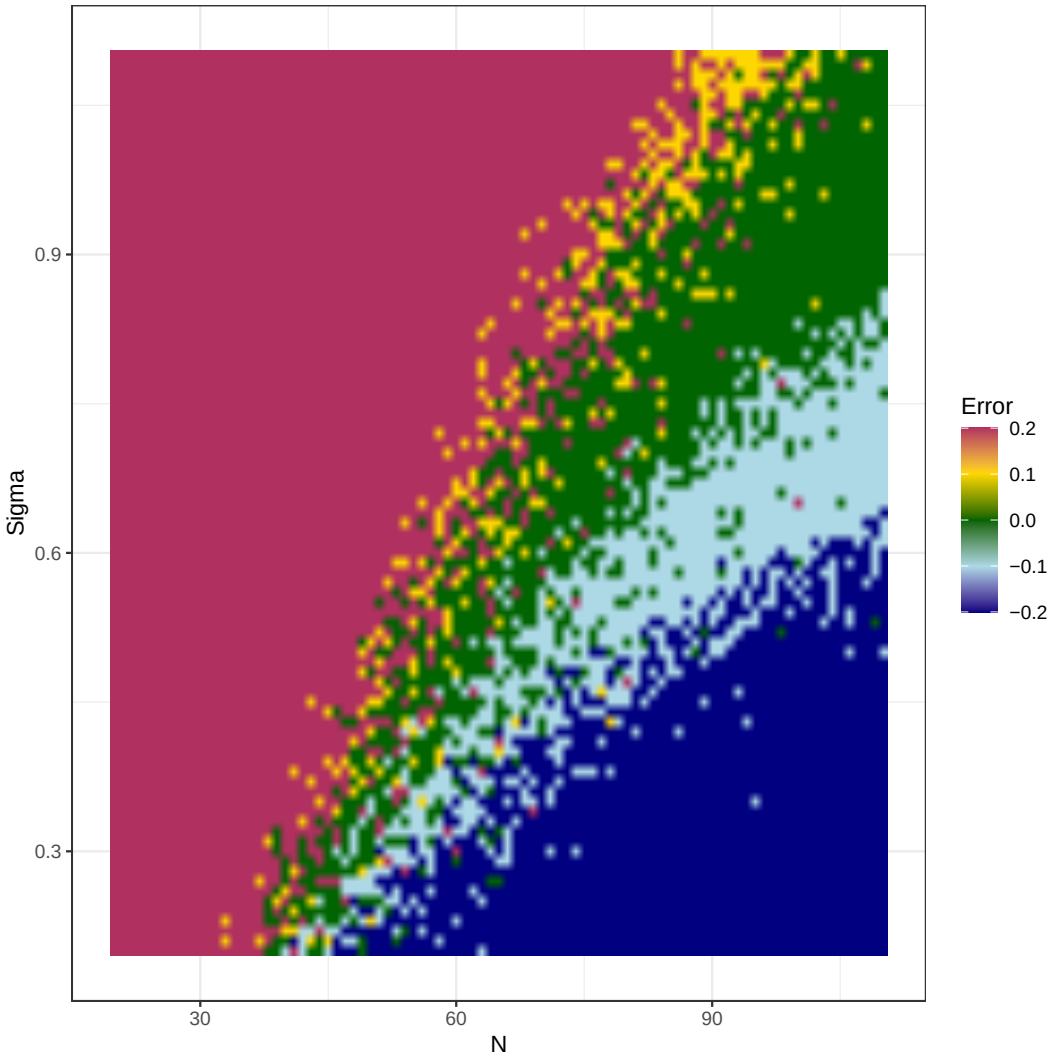

Figure 3: A colour map comparing the relative performance of H_2 and H_{10} using estimates of the overfit measure as the size of the datasets (N) and the noise level (sigma) increase with a fixed 10-th order legendre target function.

ii.

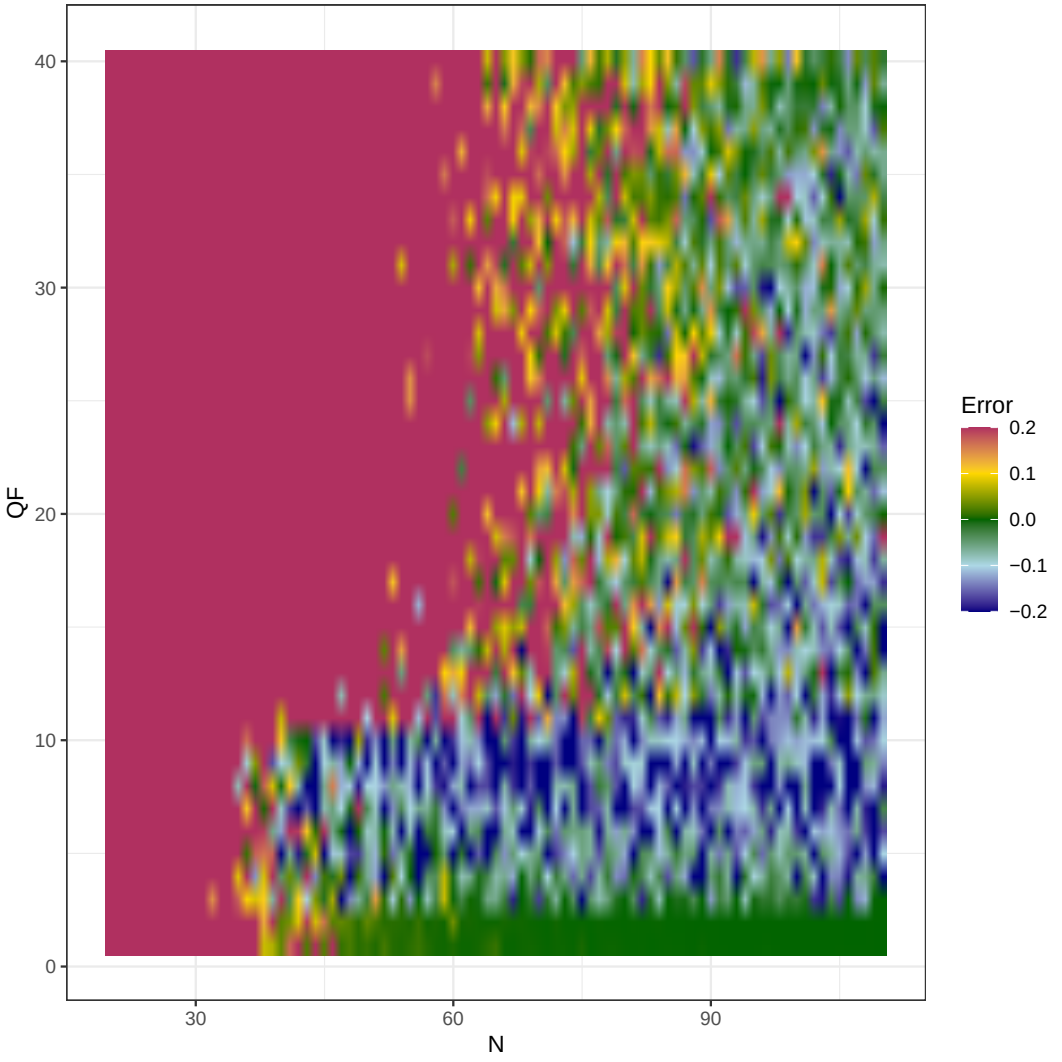

Figure 4: A colour map comparing the relative performance of H_2 and H_{10} using estimates of the overfit measure as the size of the datasets (N) and the target function complexity (QF) increase with a fixed noise level (0.2).

iii.

Figure 3 highlights the performance, according to the measure of overfitting, of hypothesis sets H_2 and H_{10} for various combinations of data set size and noise level. Figure 4 highlights the performance of the hypothesis sets as the data set size and complexity of the target function vary while the noise level is fixed at 0.2. The two plots contrast the impact of adding stochastic noise (increasing the noise level) and adding deterministic noise (increasing the complexity of the target function) on the

measure of over fitting. From both Figure 3 and Figure 4, it is seen that as the size of the data set increases, across all noise levels and target function complexity, the overfit measure decreases. Figure 3 highlights that as the noise level increases, overfitting (as shown by the red and yellow) is more prevalent for larger sizes of the data set. Similarly, Figure 4 shows that for target function complexities under 10, overfitting occurs in only the smaller data sets. However, for target function complexities above 10, overfitting is more prevalent in larger data sets. Figure 4 also highlights that after a data set size of 45, the risk of overfitting for target functions with very small complexities is minimal and close to zero, as seen by the error.

3 Question 3

i.

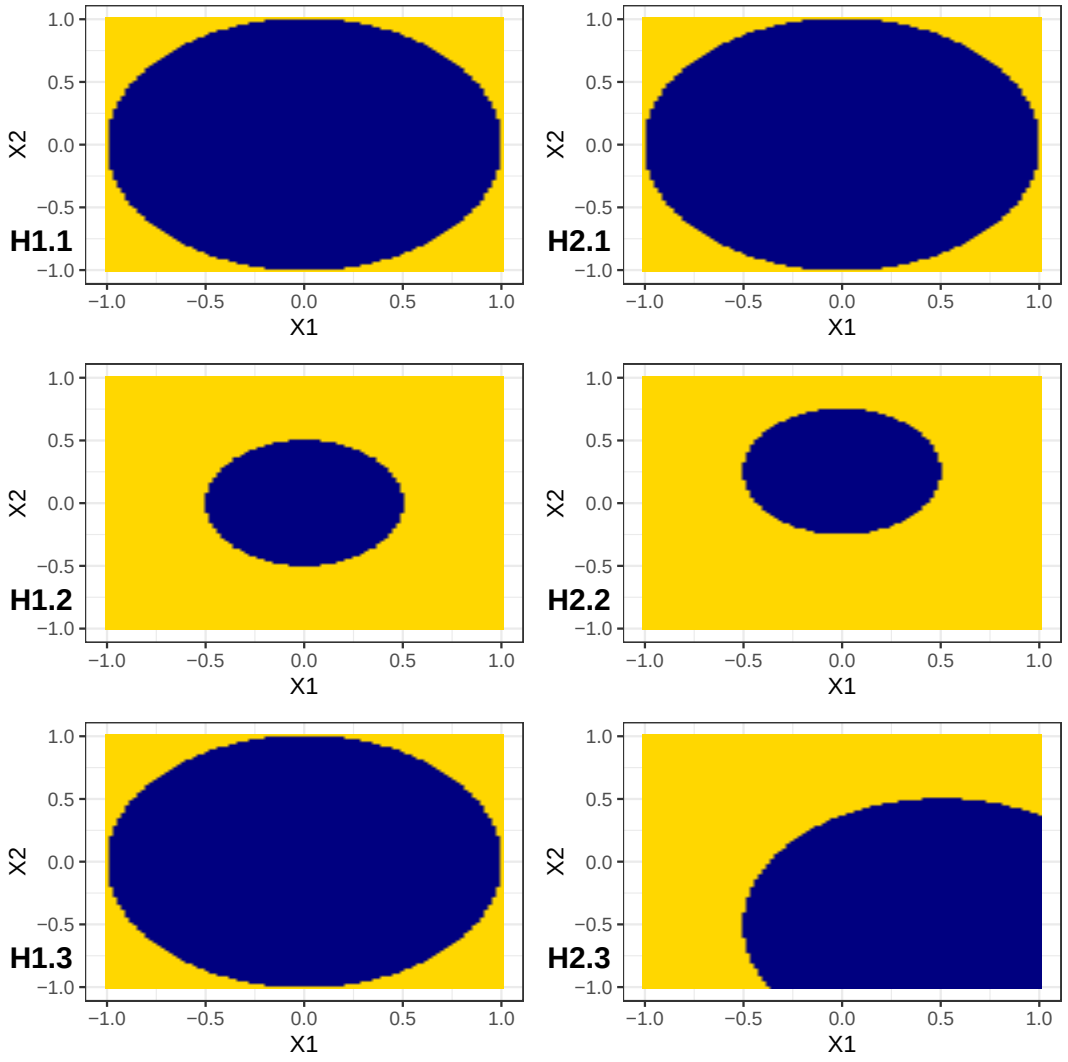

Figure 5: Plot of each hypothesis in each of the two hypothesis sets. The plots labelled H1, on the left, belong to the first hypothesis set. The plots labelled H2, on the right, belong to the second hypothesis set. The rows highlight the comparison of the hypothesis in each of the hypothesis sets.

ii.

(A) This hypothesis set produces circles that are centered at (0, 0). The size of the circles are dependent on the values for r. Hypothesis 1 and 3 in the hypothesis set (H1.1 and H1.3 in Figure 5) have r equal to 1. In contrast, Hypothesis 2, has r = 0.5, as such the circle is smaller in stature. The hypothesis set is not impacted by the values for a and b. Hypothesis 1 and Hypothesis 3 are exactly the same, even with different values for a and b.

(B) This hypothesis set produces circles that are not centered at (0, 0) but are shifted around the feature space depending on the values for a and b. Similarly, the size of the circles are dependent on r. The size of the hypotheses across hypothesis sets are the same. The value of a shifts the centre of the circle along the x1 axis. Positive values shift it to the right (as seen in H2.3) and negative values shift it to the left. The value of b shifts the centre of the circle along the x2 axis. Positive values shift it up (as seen in H2.2) and negative values shift it down (as seen in H2.3).

iii.

(A) Break Points = 3

(B) Break Points = 4

iv.

(A) VC Dimension = 2

(B) VC Dimension = 3

v.

(A) Growth Function: $\binom{n}{1} + \binom{n}{2}$

(B) Growth function: $\binom{n}{1} + \binom{n}{2} + \binom{n}{3}$

4 Appendix

4.1 R Code

4.1.1 Question 1

```r
### Libraries (if necessary)

library(tidyverse)

set.seed(2023)

##### Q1.i

N = 100
x = seq(-1, 1, length = N)

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, 6)

for (i in 0:5){

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

legPolyData = as.data.frame(cbind(x, legPolyMat))

pdf("legPoly_Q1.pdf")
ggplot(legPolyData, aes(x = x)) +
  geom_line(aes(y = V2), col = "black", linewidth = 2) +
  geom_line(aes(y = V3), col = "lightpink", linewidth = 2) +
  geom_line(aes(y = V4), col = "mediumpurple3", linewidth = 2) +
  geom_line(aes(y = V5), col = "lightblue", linewidth = 2) +
  geom_line(aes(y = V6), col = "gold", linewidth = 2) +
  geom_line(aes(y = V7), col = "lawngreen", linewidth = 2) +
  labs(x = "X", y = expression("L"[q](x))) +
  ylim(-1, 1) +
  theme_bw(base_size = 16)
dev.off()
```

```
##### Q1.ii

### Choose Qf = 10

Qf = 10
nsam = (Qf + 1)*6
sam = runif(nsam, -1, 1)
coefMat = matrix(sam, (Qf+1), 6)

f1 = function(alphas, x){

  q = length(alphas) - 1
  f = 0

  for(i in 0:q){

    f = f + (alphas[i+1]) * (x^i)
  }

  return(f)
}

f2 = function(betas, x){

  q = length(betas) - 1
  f = 0

  for(i in 0:q){

    leg = legFunc(x, i)
    f = f + (betas[i+1]) * leg
  }

  return(f)
}

fMat = matrix(0, N, 6)

for (a in 1:3){

  fMat[, a] = f1(coefMat[, a], x)
}

for (b in 4:6){

  fMat[, b] = f2(coefMat[, b], x)
}

targetPlotData = data.frame("X" = x,
                            "T1" = fMat[, 1],
```

```
                              "T2" = fMat[, 2],
                              "T3" = fMat[, 3],
                              "T4" = fMat[, 4],
                              "T5" = fMat[, 5],
                              "T6" = fMat[, 6])

pdf("target1_Q1.pdf")
a = ggplot(targetPlotData, aes(x = X)) +
  geom_line(aes(y = T1), col = "black", linewidth = 2) +
  geom_line(aes(y = T2), col = "maroon", linewidth = 2) +
  geom_line(aes(y = T3), col = "purple", linewidth = 2) +
  labs(x = "X", y = "f(x)") +
  ylim(-4, 4) +
  theme_bw(base_size = 16)
dev.off()

pdf("target2_Q1.pdf")
b = ggplot(targetPlotData, aes(x = X)) +
  geom_line(aes(y = T4), col = "dodgerblue3", linewidth = 2) +
  geom_line(aes(y = T5), col = "gold", linewidth = 2) +
  geom_line(aes(y = T6), col = "darkolivegreen2", linewidth = 2) +
  labs(x = "X", y = "f(x)") +
  ylim(-4, 4) +
  theme_bw(base_size = 16)
dev.off()

pdf("targetBoth_Q1.pdf")
ggarrange(a, b, nrow = 2, ncol = 1,
          labels = c("A", "B"))
dev.off()
```

4.1.2 Question 2

```
### Libraries (if necessary)

library(tidyverse)
library(reshape2)
library(tictoc)

set.seed(2023)

##### Q2.i

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)
}

legPoly = function(x, q){

  legPolyMat = matrix(0, length(x), q+1)

  for (i in 0:q){

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

  return(legPolyMat)
}

f2 = function(betas, x){

  q = length(betas) - 1
  f = 0

  for(i in 0:q){

    leg = legFunc(x, i)
    f = f + (betas[i+1]) * leg
  }

  return(f)
}
```

```
betasTen = runif(11, -1, 1)

dx = 0.01

errorSimulation = function(N, sigma){

  xLat = seq(-1,1, dx)
  nDX  = length(xLat)
  gBar2 = rep(0, nDX)
  gBar10 = rep(0, nDX)
  GD2   = matrix(0, M, nDX)
  GD10   = matrix(0, M, nDX)

  for (i in 1:M){

    xi = runif(N, -1, 1)
    eps = rnorm(N, mean = 0, sd = sigma^2)
    target = f2(betasTen, xi) + eps

    X2 = legPoly(xi, 2)
    X10 = legPoly(xi, 10)

    betaTwo = solve(t(X2) %*% X2) %*% t(X2) %*% target
    betaTen = solve(t(X10) %*% X10) %*% t(X10) %*% target

    resPred2 = legPoly(xLat, 2) %*% betaTwo
    resPred10 = legPoly(xLat, 10) %*% betaTen

    gBar2 = gBar2 + resPred2
    gBar10 = gBar10 + resPred10

    GD2[i, ] = resPred2
    GD10[i, ] = resPred10

  }

  gBar2 = gBar2 / M
  gBar10 = gBar10 / M

  phiX = 0.5

  yF = f2(betasTen, xLat)

  bias2 = sum((gBar2 - yF)[-nDX]^2 * phiX * dx)

  bias10 = sum((gBar10 - yF)[-nDX]^2 * phiX * dx)

  ones = matrix(1, M, 1)

  varX2 = colSums((GD2 - ones %*% t(gBar2))^2) / M
```

```
  varX10 = colSums((GD10 - ones %*% t(gBar10))^2) / M

  var2 = sum(varX2[-nDX] * phiX * dx)
  var10 = sum(varX10[-nDX] * phiX * dx)

  E_Out2 = bias2 + var2
  E_Out10 = bias10 + var10

  return(list(E2 = E_Out2, E10 = E_Out10,
              bias2 = bias2, bias10 = bias10,
              var2 = var2, var10 = var10))
}

nSeq = seq(20, 110, by = 1)
sigmaSeq = seq(0.2, 1.1, by = 0.01)

errorMat = matrix(0, length(sigmaSeq), length(nSeq))
E2Mat = matrix(0, length(sigmaSeq), length(nSeq))
E10Mat = matrix(0, length(sigmaSeq), length(nSeq))

Var2Mat = matrix(0, length(sigmaSeq), length(nSeq))
Var10Mat = matrix(0, length(sigmaSeq), length(nSeq))

Bias2Mat = matrix(0, length(sigmaSeq), length(nSeq))
Bias10Mat = matrix(0, length(sigmaSeq), length(nSeq))

M = 500

for(s in 1:length(sigmaSeq)){

  for(n in 1:length(nSeq)){
    print(paste0("Sigma: ", s, ". N: ", n, "."))
    tic()
    error = errorSimulation(nSeq[n], sigmaSeq[s])
    E2Mat[s, n] = error$E2
    E10Mat[s, n] = error$E10
    Var2Mat[s, n] = error$var2
    Var10Mat[s, n] = error$var10
    Bias2Mat[s, n] = error$bias2
    Bias10Mat[s, n] = error$bias10
    errorMat[s, n] = E10Mat[s, n] - E2Mat[s, n]
    toc()
  }

}

errorPlotData = melt(errorMat)
plotExpand = expand.grid(sigmaSeq, nSeq)
errorPlotData = cbind(plotExpand, errorPlotData$value)
colnames(errorPlotData) = c("Sigma", "N", "Error")
```

```
round(errorPlotData$Error, 3)

errorPlotData$Error = ifelse(errorPlotData$Error >= 0.2,
                             0.2, errorPlotData$Error)
errorPlotData$Error = ifelse((errorPlotData$Error < 0.2) &
                             (errorPlotData$Error >= 0.1),
                                 0.1, errorPlotData$Error)
errorPlotData$Error = ifelse((errorPlotData$Error < 0.1) &
                             (errorPlotData$Error >= -0.1),
                                 0, errorPlotData$Error)
errorPlotData$Error = ifelse((errorPlotData$Error < -0.1) &
                             (errorPlotData$Error >= -0.15),
                                 -0.1, errorPlotData$Error)
errorPlotData$Error = ifelse(errorPlotData$Error < -0.15,
                                 -0.2, errorPlotData$Error)

pdf("errorPlot_Q2A.pdf")
ggplot(errorPlotData, aes(x = N, y = Sigma, fill = Error)) +
  geom_raster(interpolate = TRUE) +
  xlab("N") + ylab("Sigma") + labs(color = "Error") +
  scale_fill_gradientn(colours = c("navyblue", "lightblue",
                         "darkgreen", "gold", "maroon")) +
  theme_bw()
dev.off()

##### Q2.ii

sigma = 0.2

errorSimulationQF = function(N, qf){

  xLat = seq(-1,1, dx)
  nDX  = length(xLat)
  gBar2 = rep(0, nDX)
  gBar10 = rep(0, nDX)
  GD2   = matrix(0, M, nDX)
  GD10   = matrix(0, M, nDX)
  betasTarget = runif(qf + 1, -1, 1)

  fx = f2(betasTarget, xLat)
  norm = sqrt(sum(fx^2) * dx)
  newBetas = betasTarget * (1/norm)

  for (i in 1:M){

    xi = runif(N, -1, 1)
    eps = rnorm(N, mean = 0, sd = sigma^2)

    target = f2(newBetas, xi) + eps
```

```
    X2 = legPoly(xi, 2)
    X10 = legPoly(xi, 10)

    betaTwo = solve(t(X2) %*% X2) %*% t(X2) %*% target
    betaTen = solve(t(X10) %*% X10) %*% t(X10) %*% target

    resPred2 = legPoly(xLat, 2) %*% betaTwo
    resPred10 = legPoly(xLat, 10) %*% betaTen

    gBar2 = gBar2 + resPred2
    gBar10 = gBar10 + resPred10

    GD2[i, ] = resPred2
    GD10[i, ] = resPred10

  }

  gBar2 = gBar2 / M
  gBar10 = gBar10 / M

  phiX = 0.5

  yF = f2(newBetas, xLat)

  bias2 = sum((gBar2 - yF)[-nDX]^2 * phiX * dx)

  bias10 = sum((gBar10 - yF)[-nDX]^2 * phiX * dx)

  ones = matrix(1, M, 1)

  varX2 = colSums((GD2 - ones %*% t(gBar2))^2) / M

  varX10 = colSums((GD10 - ones %*% t(gBar10))^2) / M

  var2 = sum(varX2[-nDX] * phiX * dx)
  var10 = sum(varX10[-nDX] * phiX * dx)

  E_Out2 = bias2 + var2
  E_Out10 = bias10 + var10

  return(list(E2 = E_Out2, E10 = E_Out10,
              bias2 = bias2, bias10 = bias10,
              var2 = var2, var10 = var10))
}

nSeq2 = seq(20, 110, by = 1)
qfSeq = seq(1, 40, by = 1)

errorMat2 = matrix(0, length(qfSeq), length(nSeq2))

M = 500
```

```
for(q in 1:length(qfSeq)){

  for(n in 1:length(nSeq2)){
    print(paste0("QF: ", q, ". N: ", n, "."))
    tic()
    error = errorSimulationQF(nSeq2[n], qfSeq[q])
    E2 = error$E2
    E10 = error$E10
    errorMat2[q, n] = E10 - E2
    toc()
  }

}

errorPlotData2 = melt(errorMat2)
plotExpand2 = expand.grid(qfSeq, nSeq2)
errorPlotData2 = cbind(plotExpand2, errorPlotData2$value)
colnames(errorPlotData2) = c("QF", "N", "Error")

round(errorPlotData2$Error, 3)

aa = 0.2
ee = -0.2

errorPlotData2$Error = ifelse(errorPlotData2$Error >= aa,
                              0.2, errorPlotData2$Error)
errorPlotData2$Error = ifelse(errorPlotData2$Error < ee,
                              -0.2, errorPlotData2$Error)

pdf("errorPlot_Q2B.pdf")
ggplot(errorPlotData2, aes(x = N, y = QF, fill = Error)) +
  geom_raster(interpolate = TRUE) +
  xlab("N") + ylab("QF") + labs(color = "Error") +
  scale_fill_gradientn(colours = c("navyblue", "lightblue",
                              "darkgreen", "gold", "maroon")) +
  theme_bw()
dev.off()
```

4.1.3 Question 3

```r
### Libraries (if necessary)

library(tidyverse)
library(ggpubr)
library(tictoc)

set.seed(2023)

##### Q3.i

H1 = function(x1, x2, a, b, r){

  sign(x1^2 + x2^2 - r^2)
}

H2 = function(x1, x2, a, b, r){

  sign((x1-a)^2 + (x2-b)^2 - r^2)
}

N = 100
x1 = seq(-1, 1, length = N)
x2 = x1

hVal1.1 = matrix(0, N, N)
hVal1.2 = hVal1.1
hVal1.3 = hVal1.1

hVal2.1 = hVal1.1
hVal2.2 = hVal1.1
hVal2.3 = hVal1.1

expComb = expand.grid(x1, x2)

for(i in 1:N){

  for (j in 1:N){

    hVal1.1[i, j] = H1(x1[i], x2[j], 0, 0, 1)
    hVal1.2[i, j] = H1(x1[i], x2[j], 0, 0.25, 0.5)
    hVal1.3[i, j] = H1(x1[i], x2[j], 0.5, -0.5, 1)

    hVal2.1[i, j] = H2(x1[i], x2[j], 0, 0, 1)
    hVal2.2[i, j] = H2(x1[i], x2[j], 0, 0.25, 0.5)
    hVal2.3[i, j] = H2(x1[i], x2[j], 0.5, -0.5, 1)

  }
}
```

```
h1.1Data = melt(hVal1.1)
h1.1Data = cbind(expComb, h1.1Data$value)
colnames(h1.1Data) = c("X1", "X2", "Sign")

a = ggplot(h1.1Data, aes(x = X1, y = X2, fill = Sign)) +
  geom_raster(interpolate = TRUE) +
  xlab("X1") + ylab("X2") + labs(color = "Sign") +
  scale_fill_gradientn(colours = c("navyblue", "gold")) +
  theme_bw() +
  theme(legend.position = "none")

h1.2Data = melt(hVal1.2)
h1.2Data = cbind(expComb, h1.2Data$value)
colnames(h1.2Data) = c("X1", "X2", "Sign")

b = ggplot(h1.2Data, aes(x = X1, y = X2, fill = Sign)) +
  geom_raster(interpolate = TRUE) +
  xlab("X1") + ylab("X2") + labs(color = "Sign") +
  scale_fill_gradientn(colours = c("navyblue", "gold")) +
  theme_bw() +
  theme(legend.position = "none")

h1.3Data = melt(hVal1.3)
h1.3Data = cbind(expComb, h1.3Data$value)
colnames(h1.3Data) = c("X1", "X2", "Sign")

c = ggplot(h1.3Data, aes(x = X1, y = X2, fill = Sign)) +
  geom_raster(interpolate = TRUE) +
  xlab("X1") + ylab("X2") + labs(color = "Sign") +
  scale_fill_gradientn(colours = c("navyblue", "gold")) +
  theme_bw() +
  theme(legend.position = "none")

h2.1Data = melt(hVal2.1)
h2.1Data = cbind(expComb, h2.1Data$value)
colnames(h2.1Data) = c("X1", "X2", "Sign")

d = ggplot(h2.1Data, aes(x = X1, y = X2, fill = Sign)) +
  geom_raster(interpolate = TRUE) +
  xlab("X1") + ylab("X2") + labs(color = "Sign") +
  scale_fill_gradientn(colours = c("navyblue", "gold")) +
  theme_bw() +
  theme(legend.position = "none")

h2.2Data = melt(hVal2.2)
h2.2Data = cbind(expComb, h2.2Data$value)
colnames(h2.2Data) = c("X1", "X2", "Sign")

e = ggplot(h2.2Data, aes(x = X1, y = X2, fill = Sign)) +
  geom_raster(interpolate = TRUE) +
  xlab("X1") + ylab("X2") + labs(color = "Sign") +
```

```
  scale_fill_gradientn(colours = c("navyblue", "gold")) +
  theme_bw() +
  theme(legend.position = "none")

h2.3Data = melt(hVal2.3)
h2.3Data = cbind(expComb, h2.3Data$value)
colnames(h2.3Data) = c("X1", "X2", "Sign")

f = ggplot(h2.3Data, aes(x = X1, y = X2, fill = Sign)) +
  geom_raster(interpolate = TRUE) +
  xlab("X1") + ylab("X2") + labs(color = "Sign") +
  scale_fill_gradientn(colours = c("navyblue", "gold")) +
  theme_bw() +
  theme(legend.position = "none")

pdf("circles_Q3.A.pdf")
ggarrange(a, d , b, e, c, f, ncol = 2, nrow = 3,
          labels = c("H1.1", "H2.1", "H1.2", "H2.2", "H1.3", "H2.3"),
          hjust = -0.2,
          vjust = 12,
          align = "v")
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 a1. `https://github.com/lukebarnes-ct/MachineLearning_A1`, 2023.