Assignment 1

Tiffany Woodley

September 2018

1 Question 1

1.I Properties of Legendre Polynomials

- Orthogonal All Legendre Polynomials are orthogonal to each other which helps to stop overlapping when they are summed up
- Normalized Legendre polynomials are normalized such that y =1 at x = 1

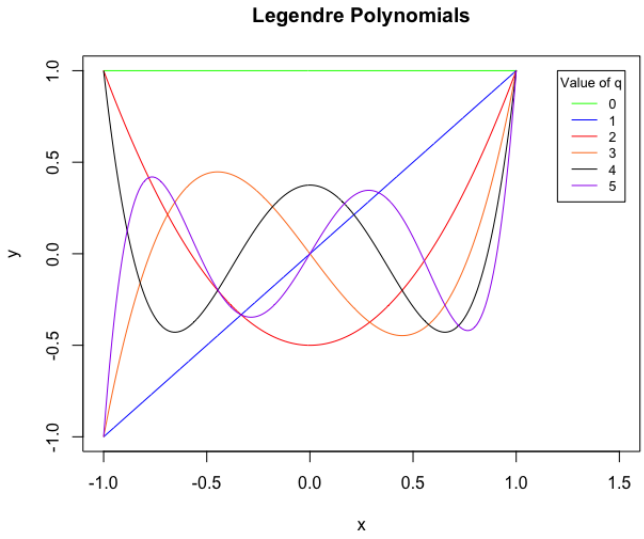

Figure 1: Legendre Polynomials

1.II Difference between Polynomials and Legendre Polynomials

- Defined curves The Legendre Polynomial functions have more defined curves, the general polynomials have shallower curves

- Cancellation This difference between the curves is more clear as the the order increases, this is because the standard polynomials are not orthogonal and some of the previous orders will interfere with each other averaging the curves out

- Dominating lower orders Since the x range is between -1 and 1 the part added to the target function by higher order polynomials will have less impact compared to lower order polynomials. And so the lower order polynomials will make up the dominant part of the target function.

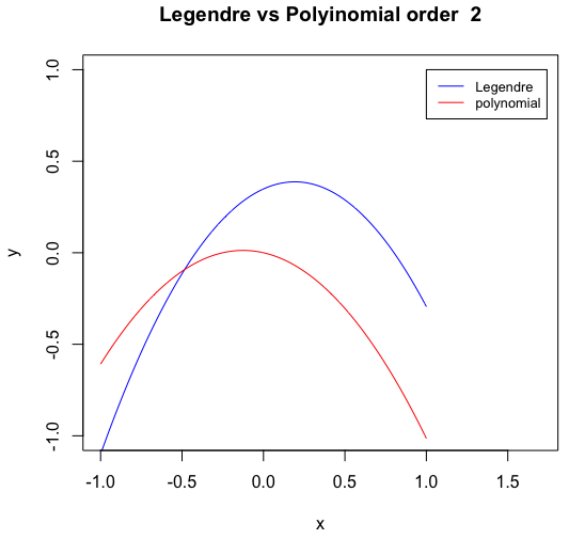

Figure 2: Order 2 Legendre vs Polynomial

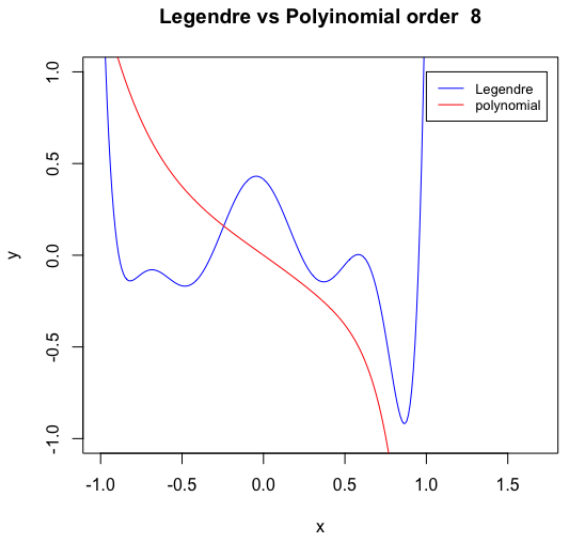

Figure 3: Order 8 Legendre vs Polynomial

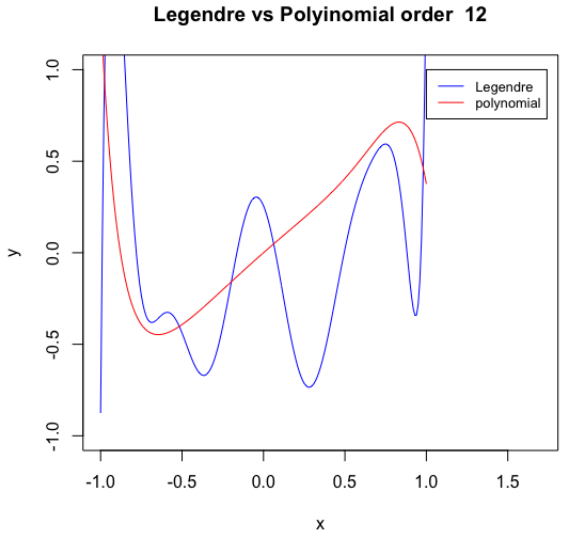

Figure 4: Order 12 Legendre vs Polynomial

2 Question 2

2.I

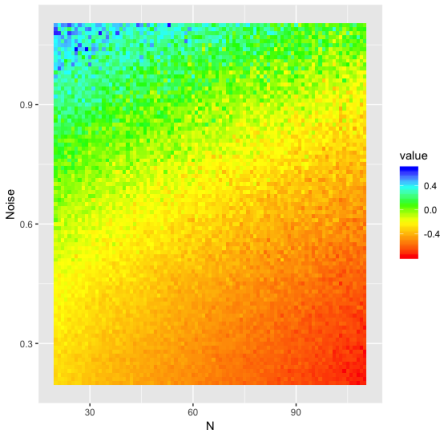

Figure 5: E[Eout(10) - Eout(2)]

2.II

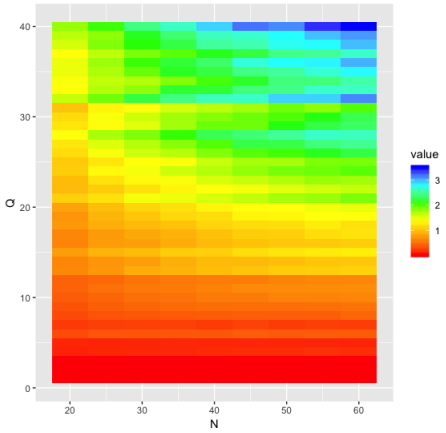

Figure 6: E[Eout(10) - Eout(2)]

2.III

- The effect of N the larger the N the less likely the model is to over-fit, this is because the larger the sample N the more insight into the actual target function the model has. This reduces it's likelihood to try fit to the noise by confusing it as part of the target function. This can be seen from Figure 5 as the values of N increase the over-fitting measure decreases. For high values of N it even becomes negative indicating that the out of sample error for the 10th order fit is lower than the second order fit. The 10th order can better approximate the 10th order target function with more N as the affect of the noise during the fitting of the model is decreased

- The effect of noise As the noise increases the more likely the model is to over-fit, this is because the more noise added to the data the less information on the initial target function the model has and will start fitting onto the noise. This phenomenon can be seen in Figure 5 as the value for sigma increases the value for over-fitting increases.

- The effect of q If the order of the target function surpasses the order of the fitted model the functions should not over-fit the data as they would not be flexible enough to fit the target complexity and would ignore the noise. When the target complexity is close to 2 the out of sample error for the second degree fit would be low and increase as the target complexity increased. At these low order target functions the ten degree fit would have a higher out of sample error due to it's flexibility and ability to fit to the noise. When the target function is closer to order 10 the out of sample error for the tenth order fit would be lower whilst the second degree fit would be higher. Increased order target functions above the order of 10 will have a high out of sample error for both the second order fit and tenth order fit and so the difference between them would not differ too much. The increase in N will bring down the over-fitting measure for all values of q as you are giving the model more information about the target function to fit too. The heat map produced in Figure 6 unfortunately doesn't paint this picture and points more to the over-fitting increasing as N increase which is incorrect.

3 Appendix:

3.I R Code: Question 1

```
# QUESTION 1

## legendre functions

leg.in <- function(x, n){

  x.return <- 0

  for(i in 0:n){
    x.return <-  x.return + (x^i)*choose(n,i)*choose((n+i
        -1)/2, n)
  }
  return(x.return*(2^n))

}

sum.leg<-function(x,n){
  coeffs <- runif(n, min = -1, max = 1)
  x.return <- 0
  for (i in 1:n){
    x.return <- x.return + leg.in(x,i)*coeffs[i]
  }
  return(x.return)

}

## 1.1 plotting the legendre functions

x <- seq(-1, 1, 0.01)

plots <- c(0:5)

color.vals <- c("green", "blue", "red", "chocolate1", "
    black", "purple")

for (q in plots){

  if(q == 0){
    plot(x, leg.in(x, q), type = "l", col = color.vals[q
        +1], ylim = c(-1,1), xlim = c(-1, 1.5), ylab = "y"
        )
  }else{
```

```
    lines(x, leg.in(x, q), type = "l", col = color.vals[q
        +1], ylim = c(-1,1), xlim = c(-1, 1.5))
  }
}

legend(1.2, 1, legend=plots , col= color.vals,lty = 1:1,
    cex=0.8, title = "Value of q")
title(main = "Legendre Polynomials")

## poly functions

poly.in <- function(x, n){

  x.return <- 0

  for(i in 1:n){
    x.return <- x.return + (x^i)
  }
  return(x.return)

}

sum.poly<-function(x,n){
  coeffs <- runif(n, min = -1, max = 1)
  x.return <- 0

  for (i in 1:n){
    x.return <- x.return + poly.in(x,i)*coeffs[i]
  }
  return(x.return)

}

random_q <- runif(3, min = 0, max = 20)

##1.2 plotting random order polynomials vs legendre
    polynomials

##  1.2.1

q <- round(random_q[1],digits=0)


plot(x, sum.leg(x, q), type = "l", col = "blue", ylim = c
    (-1,1), xlim = c(-1, 1.7), ylab = "y")
lines(x, sum.poly(x, q), type = "l", col = "red", ylim =
```

```
    c(-1,1), xlim = c(-1, 1.7), ylab = "y")

legend(1, 1, legend=c("Legendre", "polynomial") , col= c(
    "blue", "red"),lty = 1:1, cex=0.8)
title(main = paste(c("Legendre vs Polyinomial order ", q)
    , collapse = " "))
```

1.2.2

```
q <- round(random_q[2], digits=0)

plot(x, sum.leg(x, q), type = "l", col = "blue", ylim = c
    (-1,1), xlim = c(-1, 1.7), ylab = "y")
lines(x, sum.poly(x, q), type = "l", col = "red", ylim =
    c(-1,1), xlim = c(-1, 1.7), ylab = "y")

legend(1, 1, legend=c("Legendre", "polynomial") , col= c(
    "blue", "red"),lty = 1:1, cex=0.8)
title(main = paste(c("Legendre vs Polyinomial order ", q)
    , collapse = " "))
```

1.2.3

```
q <- round(random_q[3], digits=0)

plot(x, sum.leg(x, q), type = "l", col = "blue", ylim = c
    (-1,1), xlim = c(-1, 1.7), ylab = "y")
lines(x, sum.poly(x, q), type = "l", col = "red", ylim =
    c(-1,1), xlim = c(-1, 1.7), ylab = "y")

legend(1, 1, legend=c("Legendre", "polynomial") , col= c(
    "blue", "red"),lty = 1:1, cex=0.8)
title(main = paste(c("Legendre vs Polyinomial order ", q)
    , collapse = " "))
```

3.II R Code: Question 2

```
# QUESTION 2

## legendre functions

leg.in <- function(x, n){

  x.return <- 0

  for(i in 0:n){
    x.return <-  x.return + (x^i)*choose(n,i)*choose((n+i
        -1)/2, n)
  }
  return(x.return*(2^n))

}

sum.leg<-function(x,n){
  coeffs <- runif(n, min = -1, max = 1)
  x.return <- 0
  for (i in 1:n){
    x.return <- x.return + leg.in(x,i)*coeffs[i]
  }
  return(x.return)

}

## visualising
coeffs <- runif(10, min = -1, max = 1)
x <- seq(-1, 1, 0.1)
y <- sum.leg(x,10)
y_noise <- sapply(y, function(y) {y = y+ rnorm(1,0, 0.5)
    })
plot(c(-1,1),c(-4,4),main="target & noise",type="n",xlab=
    "x",ylab=expression(f(x)))
lines(x,y,type="l",lty=2,col="blue")
points(x,y_noise)
lines(x, 0.5479*x-1.2451*x^2, type = 'l', lty = 2, col =
    "red")
lines(x, -0.3744*x-10.1823*x^2 +7.8308*x^3+79.89*x
    ^4-4.1593*x^5-194.3664*x^6-26.5131*x^7+210.8514*x
    ^8+25.0966*x^9-187.9636*x^10, type = 'l', lty = 2, col
     = "green")

## question 2 functions
```

```
integrate.sum <- function(x,y,increment)
{
  sum <- 0
  for (i in 1:length(x))
  {
    sum <- sum + (x[i] - y[i])^2
  }
  return(abs(sum)*(increment))
}


simulate.data.sets = function(n,x,func,sig){
  l<-length(func)
  dat<-matrix(rep(NA,2*n),ncol=n)
  xdat<-floor(runif(n)*l)+1
  ydat<-func[xdat]+rnorm(n,0,sig)
  xdat<-x[xdat]
  Data<-data.frame(xdat,ydat)
  return(Data)
}

## 2.1

q <- 10
noise <- seq(0.2, 1.1, 0.01)
N <- seq(20, 110, 1)

A = matrix( 0,  nrow=length(noise), ncol=length(N))

x <- seq(-1, 1, 0.01)
y <- sum.leg(x,10)

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

  for (i in 1:length(N)){

    avg.model.one = matrix( 0, nrow=200, ncol=length(x))
    avg.model.two = matrix(0, nrow= 200, ncol=length(x))
    for (k in 1:200)
    {
      var.data <- simulate.data.sets(N[i],x, y ,noise[j])
      two.degree.fit.var <- lm(var.data$y ~ poly(var.data
          $x,2))
      v=0
      for(l in 1:length(two.degree.fit.var$coefficient)){
```

```
      v=v+(two.degree.fit.var$coefficient[l]*(x^(l-1)))
    }

    ten.degree.fit.var <- lm(var.data$y ~ poly(var.data
        $x,10))
    d=0
    for(l in 1:length(ten.degree.fit.var$coefficient)){
      d=d+(ten.degree.fit.var$coefficient[l]*(x^(l-1)))
    }
    avg.model.one[k,] <- v
    avg.model.two[k,] <- d
  }

  variance <- mean(apply(avg.model.one, 2, var))
  bias <- (integrate.sum(apply(avg.model.one,2,mean), y
      , 0.01))
  err.two <- bias + variance + noise[j]^2


  bias <- (integrate.sum(apply(avg.model.two,2,mean), y
      , 0.01))
  variance <- mean(apply(avg.model.two, 2, var))
  err.ten <- bias + variance + noise[j]^2
  A[j,i] <- err.ten - err.two
 }
}

heatmap.one <- A

heatmap.two <- heatmap.one
row.names(heatmap.two) <- as.charachter(seq(20, 110, 1))
col.names(heatmap.two) <- as.charachter(seq(0.2, 1.1,
   0.01))
print(dfplot)
library(reshape2)
library(ggplot2)
ggplot(data = melt(t(heatmap.two)), aes(x=Var1, y=Var2))
   +
  geom_tile(aes(fill = value)) +
  scale_fill_gradientn(colours = c("red", "orange", "
      yellow", "green", "cyan", "blue")) +
  xlab("N") + ylab("Noise") + scale_x_discrete(breaks
      =1:91,labels= seq(20, 110, 1)) +
  scale_y_discrete(breaks=1:91,labels= seq(0.2, 1.1,
      0.01))
```

```
## 2.2
noise <- 0.2
q <- seq(1, 40, 1)
N <- seq(20, 60, 5)

A = matrix( 0,  nrow=length(q), ncol=length(N))

x <- seq(-1, 1, 0.01)

## running code multiple times 1:50 to average out over
    multiple target functions
for(g in 1:50){
  for (j in 1:length(q)){

    y <- sum.leg(x,q[j])

    for (i in 1:length(N)){

      avg.model.one = matrix( 0, nrow=100, ncol=length(x)
          )
      avg.model.two = matrix(0, nrow= 100, ncol=length(x)
          )
      for (k in 1:100)
      {
        var.data <- simulate.data.sets(N[i],x, y ,noise)
        two.degree.fit.var <- lm(var.data$y ~ poly(var.
            data$x,2))
        v=0
        for(l in 1:length(two.degree.fit.var$coefficient)
            ){
          v=v+(two.degree.fit.var$coefficient[l]*(x^(l-1)
              ))
        }

        ten.degree.fit.var <- lm(var.data$y ~ poly(var.
            data$x,10))
        d=0
        for(l in 1:length(ten.degree.fit.var$coefficient)
            ){
          d=d+(ten.degree.fit.var$coefficient[l]*(x^(l-1)
              ))
        }

        avg.model.one[k,] <- v
        avg.model.two[k,] <- d
```

```
        }

        variance <- mean(apply(avg.model.one, 2, var))
        bias <- (integrate.sum(apply(avg.model.one,2,mean),
            y, 0.01))
        err.two <- bias + variance + noise^2


        bias <- (integrate.sum(apply(avg.model.two,2,mean),
            y, 0.01))
        variance <- mean(apply(avg.model.two, 2, var))
        err.ten <- bias + variance + noise^2
        A[j,i] <- A[j,i] +(err.ten - err.two)
      }
    }
}


B <- A/50

ggplot(data = melt(t(B)), aes(x=Var1, y=Var2)) +
  geom_tile(aes(fill = value)) +
  scale_fill_gradientn(colours = c("red", "orange", "
      yellow", "green", "cyan", "blue"))+
  scale_x_continuous(name = "N", limits = c(20,60) )
```