# Computer Lab 2

Shipeng Liu (shili506) Dongwei Ni (donni508)

## Question 1: Optimizing parameters

Finding the minimum or maximum of a function is usually presented as a goal in itself. Here you are asked to use the function **optim()** to create a procedure to approximate another function, through so{called parabolic interpolation. For this exercise let $f(x)$ be a continuous function on the interval $[0, 1]$ and let $x_0, x_1, x_2 \in [0, 1]$ such that $f(x_1) < f(x_0), f(x_2)$. We will approximate the function $f(x)$ with a function that is piecewise $\tilde{f}(x) = a_0 + a_1 x + a_2 x_2$, i.e. a piecewise quadratic function.

1. Write a function that uses **optim()** and finds values of $(a0, a1, a2)$ for which $\tilde{f}$ interpolates $f$ at user provided points x0, x1, x2. Interpolate means $f(x_0) = \tilde{f}(x0), f(x_1) = \tilde{f}(x_1)$ and $f(x_2) = \tilde{f}(x_2)$. **optim()** should minimize the squared error, i.e. find $(a0, a1, a2)$ that make $(f(x_0) - \tilde{f}(x_0))^2 + (f(x_1) - \tilde{f}(x_1))^2 + (f(x_2) - \tilde{f}(x_2))^2$ as small as possible.

```
interpolates<-function(myfun,param_x){
  stopifnot(param_x<=1 & param_x>=0)

  a=c(1,1,1)
  loss<-function(a,myfun,param_x){
    the_loss=(myfun(param_x[1])-(a[1]+a[2]*param_x[1]+a[3]*(param_x[1]^2)))^2+
      (myfun(param_x[2])-(a[1]+a[2]*param_x[2]+a[3]*(param_x[2]^2)))^2+
      (myfun(param_x[3])-(a[1]+a[2]*param_x[3]+a[3]*(param_x[3]^2)))^2
    return(the_loss)
  }
  optim_result=optim(par=a,fn=loss,myfun=myfun,param_x=param_x,method=c("BFGS"))
  return(optim_result$par)
}
```

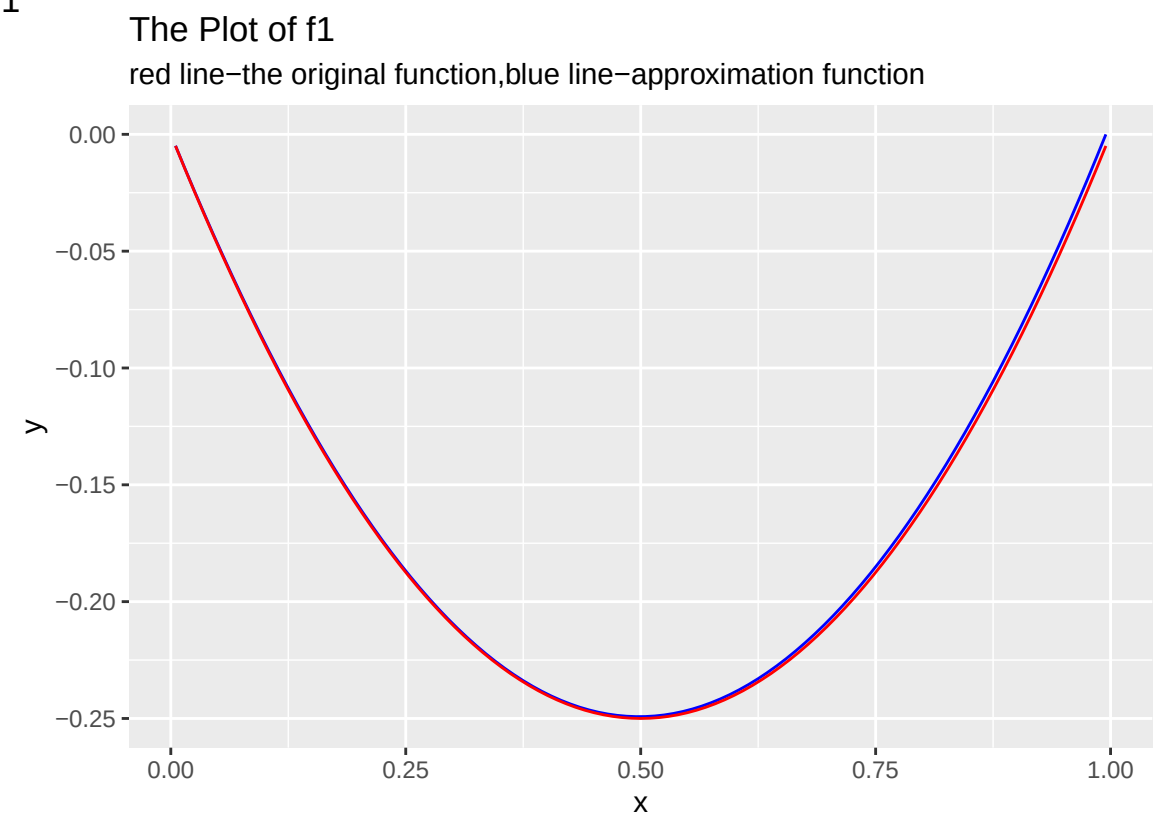
2. Now construct a function that approximates a function defined on the interval $[0, 1]$. Your function should take as a parameter the number of equal{sized intervals that $[0, 1]$ is to be divided into and the function to approximate. The target function is known at the ends of the interval and also at the mid{point of the interval. Independently on each subinterval you should approximate the target function using the parabolic interpolater implemented in the previous part i.e. use the parabolic interpolater to find $a0, a1, a2$ for each subinterval.

```
approximate<-function(cut_num,myfun){
  intervals=seq(0,1,length.out=cut_num+1) #divided intervals into 3 parts
  result=data.frame()
  for(i in 1:cut_num){
    interv=c(intervals[i],(intervals[i]+intervals[i+1])/2,intervals[i+1])
    par=interpolates(myfun,interv)
    result=rbind(result,c(interv,par))
  }
  colnames(result)=c("x0","x1","x2","a0","a1","a2")
  return(result)
}
```

3. Apply your function from the previous item to $f_1(x) = -x(1-x)$ and $f_2(x) = -xsin(10\pi x)$. Plot $f_1(\mathring{u}), \tilde{f}_1(\mathring{u})$ and $f_2(\mathring{u}), \tilde{f}_2(\mathring{u})$. How did your piecewise{parabolic interpolater fare? Explain what you observe. Take the number of subintervals to be at least 100.

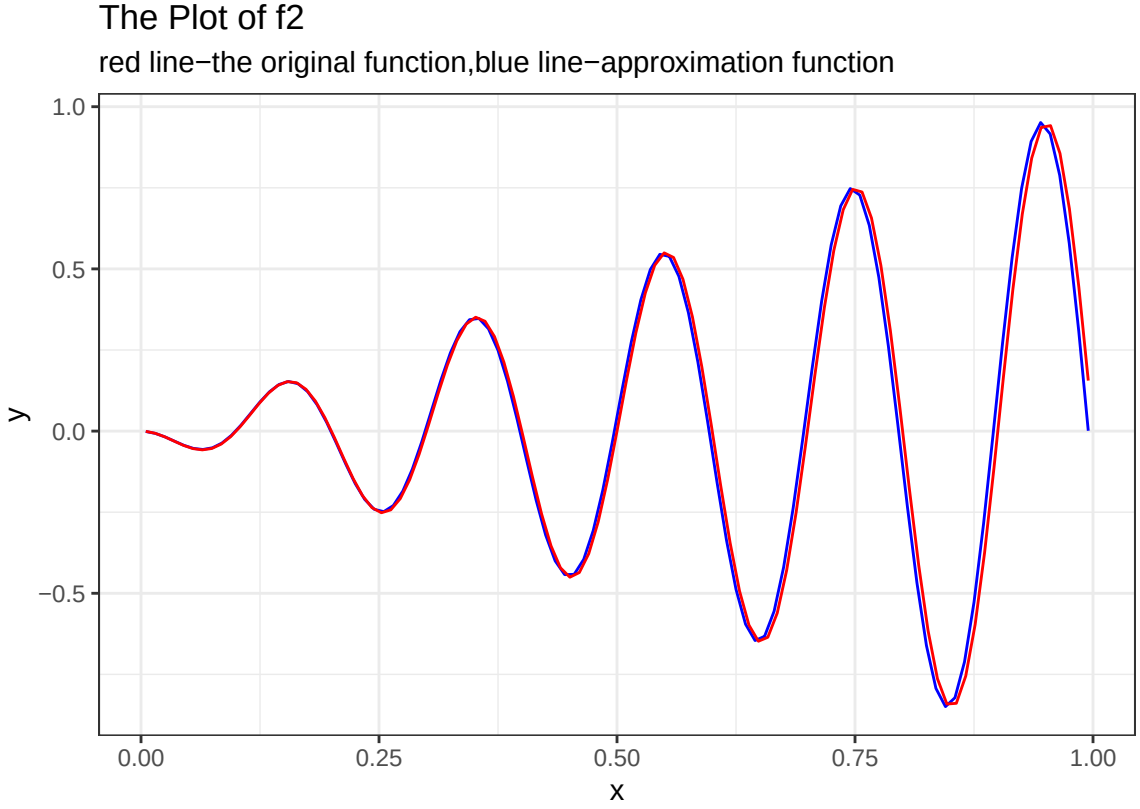
```
fun1<-function(x){
  return(-x*(1-x))
}

fun2<-function(x){
  return(-x*sin(10*pi*x))
}

result1=approximate(100,fun1)%>%mutate(value=(a0+a1*x1+a2*x2^2))
result2=approximate(100,fun2)%>%mutate(value=(a0+a1*x1+a2*x2^2))

###plot1
p1<-ggplot(data=result1,aes(x=x1,y=value))+
  geom_line(color="blue")+
  geom_function(fun=fun1,colour="red")+
  labs(x="x",y='y',
       title="The Plot of f1",
       subtitle = "red line-the original function,blue line-approximation function",
       tag = "Fig. 1")+
  theme_set(theme_bw())
p1
```

Fig. 1



```
###plot2
p2<-ggplot(data=result2,aes(x=x1,y=value))+
  geom_line(color="blue")+
  geom_function(fun=fun2,colour="red")+
  labs(x="x",y='y',
       title="The Plot of f2",
       subtitle = "red line-the original function,blue line-approximation function",
       tag = "Fig. 2")+
  theme_set(theme_bw())
p2
```

Fig. 2




- piecewise–parabolic interpolaters seems fits the original functions pretty well.The blue line and the red line basically coincide, but there are still some errors.

## Question 2: Maximizing likelihood

The file data.RData contains a sample from normal distribution with some parameters $\mu, \sigma$. For this question read ?optim in detail.

1. Load the data to R environment.

```
load("data.RData")
```

2. Write down the log-likelihood function for 100 observations and derive maximum likelihood estimators for $\mu$, $\sigma$ analytically by setting partial derivatives to zero. Use the derived formulae to obtain parameter estimates for the loaded data.

- For sample "data" from normal distribution:

$$f(x;\sigma,\mu) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

then the likelihood function:

$$L(x_1,x_2,\ldots,x_n;\sigma,\mu) = \prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

take the logarithm:

$$\begin{aligned} ln(L(x_1,x_2,\ldots,x_n;\sigma,\mu)) &= ln(\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}) \\ &= ln((\frac{1}{\sqrt{2\pi}\sigma})^n e^{-\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{2\sigma^2}}) \\ &= nln(\frac{1}{\sqrt{2\pi}\sigma}) - \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{2\sigma^2} \\ &= nln(\frac{1}{\sqrt{2\pi}}) - nln(\sigma) - \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{2\sigma^2} \end{aligned}$$

partial derivative:

$$\begin{aligned} \frac{\partial ln(L(x_1,x_2,\ldots,x_n;\sigma,\mu))}{\partial\mu} &= \frac{2\sum_{i=1}^{n}(x_i-\mu)}{2\sigma^2} \\ &= \frac{\sum_{i=1}^{n}(x_i-\mu)}{\sigma^2} \\ \frac{\partial ln(L(x_1,x_2,\ldots,x_n;\sigma,\mu))}{\partial\sigma} &= -\frac{n}{\sigma} + \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{\sigma^3} \end{aligned}$$

when equal to zero:

$$\begin{aligned} \frac{\partial ln(L(x_1,x_2,\ldots,x_n;\sigma,\mu))}{\partial\mu} = \frac{\sum_{i=1}^{n}(x_i-\mu)}{\sigma^2} &= 0 \\ \sum_{i=1}^{n}(x_i-\hat{\mu}) &= 0 \\ \sum_{i=1}^{n}x_i &= n\hat{\mu} \\ \hat{\mu} = \frac{\sum_{i=1}^{n}x_i}{n} &= \bar{X} \\ \frac{\partial ln(L(x_1,x_2,\ldots,x_n;\sigma,\mu))}{\partial\sigma} = -\frac{n}{\sigma} + \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{\sigma^3} &= 0 \\ \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{\sigma^3} &= \frac{n}{\sigma} \\ \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n} &= \hat{\sigma}^2 \end{aligned}$$

so the estimator we will use would be $\hat{\sigma}^2 = \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n}$ and $\hat{\mu} = \bar{X}$

```
mu=mean(data)
sigma=sqrt(sum((data-mu)^2)/length(data))
# value:
mu
```

```
## [1] 1.275528
```

```
sigma
```

```
## [1] 2.005976
```

3. Optimize the minus log-likelihood function with initial parameters $\mu = 0$, $\sigma = 1$. Try both Conjugate Gradient method (described in the presentation handout) and BFGS (discussed in the lecture) algorithm with gradient specified and without. Why it is a bad idea to maximize likelihood rather than maximizing log-likelihood?

```
# log-likelihood function
minusLogLikelihood<-function(params){
  mu=params[1]
  sigma=params[2]
  n=length(data)
  result=-(-(n/2)*log(2*pi*sigma^2)-(1/(2*sigma^2))*sum((data-mu)^2))
  return(result)
}
# specified gradient
grad<-function(params){
  mu=params[1]
  sigma=params[2]
  n=length(data)
  result=c(-sum(data-mu)/(sigma^2),(n/sigma)-sum((data-mu)^2)/sigma^3)
  return(result)
}

#Conjugate Gradient method with gradient
cg_grad=optim(par=c(0,1),fn=minusLogLikelihood,gr=grad,method="CG")
cg_grad
```

```
## $par
## [1] 1.275528 2.005976
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       53       17
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
#Conjugate Gradient method without gradient
cg_nongrad=optim(par=c(0,1),fn=minusLogLikelihood,method="CG")
cg_nongrad
```

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
```

```
## $counts
## function gradient
##      297       45
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
#BFGS with gradient
BFGS_grad=optim(par=c(0,1),fn=minusLogLikelihood,gr=grad,method="BFGS")
BFGS_grad
```

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       39       15
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
#BFGS without gradient
BFGS_nongrad=optim(par=c(0,1),fn=minusLogLikelihood,method="BFGS")
BFGS_nongrad
```

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       37       15
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Q: Why it is a bad idea to maximize likelihood rather than maximizing log-likelihood?

- Because the probability values are all between [0,1], the multiplication of the probability will become a small value, which may cause underflow, especially when the data set is large, the joint probability will be tends to 0, which is very unfavorable for subsequent calculations.

4. Did the algorithms converge in all cases? What were the optimal values of parameters and how many

function and gradient evaluations were required for algorithms to converge? Which settings would you recommend?

- The optimal values and counts of functions and gradients are as what it shows above. And we would recommend BFGS without gradient under such circumstances because it calls the least times and needn't you providing gradient functions but yields similar result. For the functions and initial parameters we are using, it converges. But it may falls into local minimum.

```
t <- \(x) sin(x)+0.1*x
optim(par=1,fn=t,method="CG")
```

```
## $par
## [1] -1.670963
##
## $value
## [1] -1.162084
##
## $counts
## function gradient
##       11        6
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
optim(par=5,fn=t,method="CG")
```

```
## $par
## [1] 4.612222
##
## $value
## [1] -0.5337653
##
## $counts
## function gradient
##        7        4
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
optim(par=5,fn=t,method="BFGS")
```

```
## $par
## [1] 4.612222
##
## $value
## [1] -0.5337653
##
## $counts
## function gradient
##        9        3
##
```

```
## $convergence
## [1] 0
##
## $message
## NULL
```

```
plot(t, xlim = c(-10, 10))
```