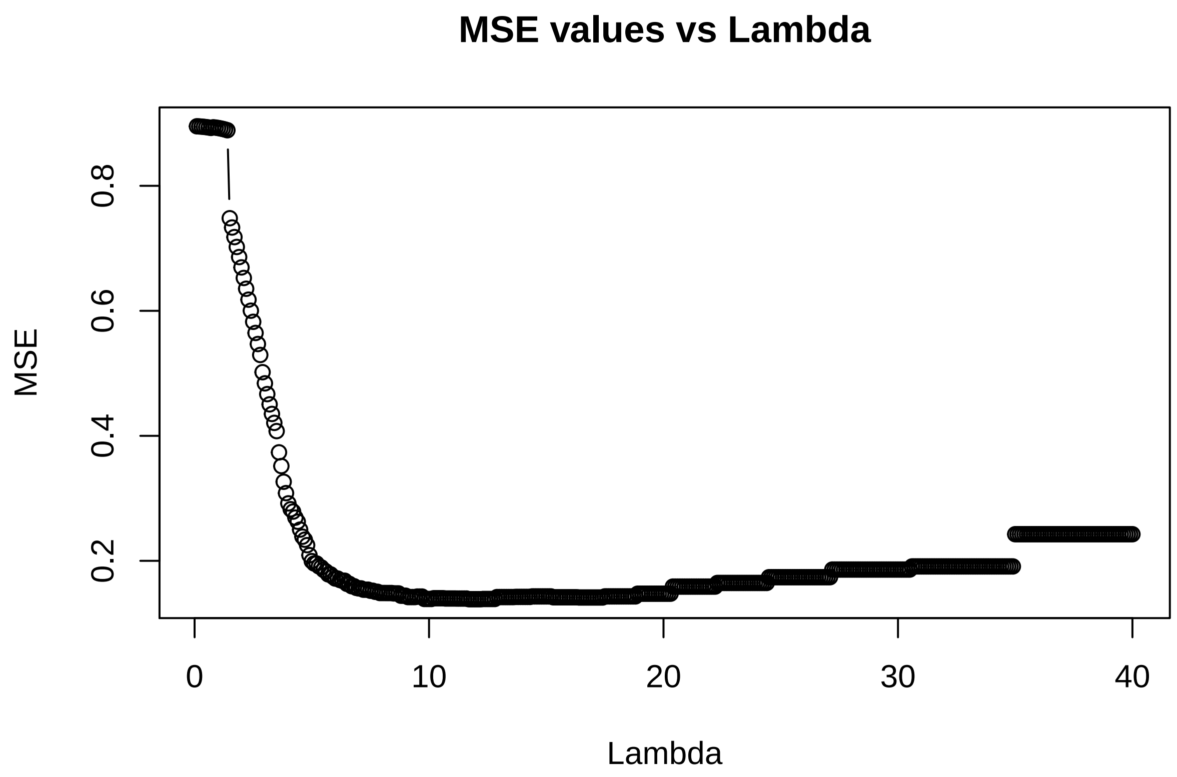
Computational Statistics Lab 2

gowtham & biswas

#1 Optimizing a model parameter



```
## [1] 117
```

```
## [1] 11.7
```

```
## $minimum
## [1] 39.99643
##
## $objective
## [1] 0.2250986
```

```
## [1] 20
```

```
## [1] 3
```

The given file mortality_rate.csv contains information about mortality rate of fruit flies for the certain period.Function myMSE is written with two arguements, lambda and pars. The pars contains vectors

X,Y,Xtest,Ytest.Using "loess" function, we fit myMSE function with response Y and predictor X. The number of evaluations of myMSE() required are 400 (number of lambda).

As per computed details, 20 myMSE() function evaluations were required to find the optimal MSE value.

When optim() function with BFGS method with starting point lambda=35 was used , we find 3 evaluations that were required to find the optimal MSE value.

Question2: Maximizing likelihood

Generalized,

$f(x_1, \cdots, x_n \mid \mu, \sigma) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} exp(\frac{-1}{2}(\frac{xi-\mu)^2}{\sigma}))$

The liklihood is given by : applying log to above equation:(negative log liklihood) $L(\mu, \sigma) = -1 \times (\frac{n}{2} \log 2\pi \times \frac{n}{2} \log \sigma^2) + \frac{1}{2\sigma^2} \sum(x_i - \mu)^2$

Partial derivative with respect to μ:

$\frac{\partial l}{\partial \mu} = -1 \times \frac{1}{\sigma^2} \sum_{i=1}^{n}(x_i - \mu)^2$

Partial derivative with respect to σ: $\frac{\partial l}{\partial \sigma} = -1 \times \frac{n}{\sigma} + \frac{1}{\sigma^3} \sum_{i=1}^{n}(x_i - \mu)^2$

For, 100 observations we take n=100.

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##      180       33
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
## $par
## [1] 1.275528 2.005976
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       53       17
##
## $convergence
## [1] 0
##
## $message
## NULL
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       37       15
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       38       15
##
## $convergence
## [1] 0
##
## $message
## NULL
```

The optimal values of parameters for Conjugate Gradient method and related to BFGS algorithm are computed above.

We see that the Convergence is 0(zero) for all the algorithms. The final values of all the algorithms are same, hence all the algorithms converge. BFGS method gives better result, because of taking less iterations compared to other algorithms.

Code Appendix

```
#step 1

data1=read.csv2("mortality_rate.csv")
data1$LMR=log(data1$Rate)
n=dim(data1)[1]
RNGversion(min(as.character(getRversion()),"3.6.2")) ## with your R-version
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
id=sample(1:n,floor(n*0.5))
train=data1[id,]
test=data1[-id,]
```

```r
#step2
X=train$Day
Y=train$LMR
Xtest=test$Day
Ytest=test$LMR
pars=list(X,Y,Xtest,Ytest)
MSE=numeric()
k<<-0
#creating myMSE function
myMSE=function(pars=list(X,Y,Xtest,Ytest),lambda){
  k<<-k+1
 fit=loess(formula = LMR~Day, data = train, enp.target =lambda)

  pred=predict(fit, newdata=Xtest)

  predMSE=1/length(Xtest)*(sum((Ytest-pred)^2))
  #print(predMSE)
  return(predMSE)
}

#myMSE(pars,10)
for (i in seq(0.1,40,0.1)) {
  MSE= c(MSE,myMSE(pars,lambda=i))

}
# step 4
plot(y=MSE,x=seq(from=0.1, to=40, by=0.1),type = "b", xlab = "Lambda",main="MSE values vs Lambda")
which.min(MSE)
opti.lambda<-seq(0.1,40,0.1)[117]
opti.lambda # optimal lambda value
#step5
k<<-0
opti <- optimize(myMSE, seq(0.1,40,0.1) , tol = 0.01)
# 20 evaluations of myMSE()needed to find this value.
opti
print(k)
k<<-0


#step 6
optimal<-optim(35, myMSE, method = "BFGS",pars=pars) # 3 evaluations
print(k)
k<<-0



# load the given data
load("data.RData")


#log likelihood calculation

llk = function(pars , data , minus = TRUE)
```

```
{
  # mu = pars$Mean
  # sig = pars$Sigma
  #

  mu = pars[1]
  sig = pars[2]

  n = length(data) #
  #print(sig)
  llkVal = -1 * ((n/2)*log(2*pi) +  (n/2)*log(sig^2) + (1/(2*(sig**2)))*(sum((data - mu)**2)))

  if(minus == FALSE)
  {
    return(llkVal)
  }

  return((-1 * llkVal))
}

gradLLK = function(pars, data)
{
  n = length(data)
  mu = pars[1]
  sig = pars[2]
  muNew = -1 *((1/(sig**2)) * sum(data - mu))
  sigmaNew = -1 * ((-n/sig) + (1 / sig^3) * (sum((data - mu)^2)))
  return(c(muNew,sigmaNew))
}

#max likelihood  estimation for mu


max_mu = function(data)
{
  return(mean(data))
}


max_sigma = function (data , mu_max){
  #mu = mean(as.numeric(data))
  return(sqrt(((sum(data - mu_max))**2)/length(data)))
}

mu_max = max_mu(data)

sigma_max = max_sigma(data,mu_max)

#pars = list("Mean" = 0 , "Sigma" = 1)

#CG Calculation
pars = c(0,1)
suppressWarnings(RNGversion("3.5.1"))
```

```r
set.seed(12345)
CG.Optim = optim(par = pars , fn = llk , method = "CG" , data = data,
                 minus = TRUE, gr = NULL)
CG.Optim
CG.OptimGrad = optim(par = pars , fn = llk , method = "CG" , data = data,
                     gr = gradLLK)

CG.OptimGrad
#BFGS
BFGS.Optim = optim(par = pars , fn = llk , method = "BFGS" , data = data,
                   gr = NULL)
BFGS.Optim
#BFGS Grad Optim
BFGS.OptimGrad = optim(par = pars , fn = llk , method = "BFGS" , data = data,
                       gr = gradLLK)

BFGS.OptimGrad
```