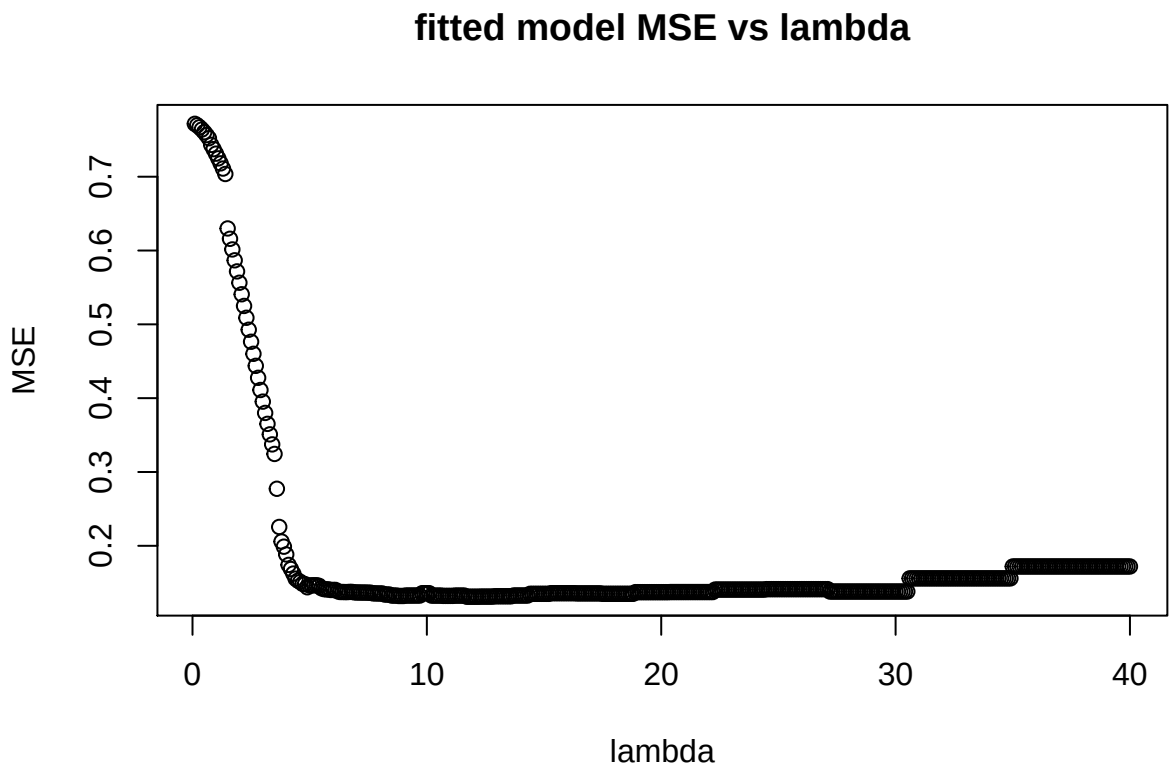
# Computer Lab 2

*Thomas Zhang*

*2016 M02 10*

## Assignment 1

We shall try to predict the log-mortality rate vs the day by writing a function `myMSE` which fits to training data a `loess` model with penalty/smoothing parameter $\lambda$, where $\lambda$ varies from 0.1 to 40 by increment of 0.1. We plot the MSE of the fitted model vs $\lambda$.




```
## [1] "minimum MSE is: 0.131"
```

```
## [1] "Corresponding optimal value of lambda is: 11.7"
```

117 evaluations of `myMSE` were required to find the optimal MSE above. In truth, there are many adjecent values of $\lambda$ which provides the optimal MSE.
Now let us try to use the function `optimize` to find , with accuracy 0.01, the optimal MSE ("objective") and compare the $\lambda$ result ("minimum") to the "true" result.

```
## $minimum
## [1] 10.69361
##
```

```
## $objective
## [1] 0.1321441
```

18 evaluations of `myMSE` are required to find this optimal value. The value is within the accuracy of the "true" result and the number of evaluations is much smaller.

Next let us try to use the `optim` function with `method = BFGS` and initial value $\lambda = 35$ to find the optimal MSE.

```
## $par
## [1] 35
##
## $value
## [1] 0.1719996
##
## $counts
## function gradient
##        1        1
##
## $convergence
## [1] 0
##
## $message
## NULL
```

3 evaluations of `myMSE` are required to find this inaccurate value of lambda. I believe this result stems from the fact that the "gradient" of `myMSE` w.r.t $\lambda$ at the chosen initial value is zero, so the quasi-newton algorithm terminates where it started.

## Assignment 2

For a normally distributed variable, the likelihood function and the log-likelihood function of a hundred observations are expressed as follows:

$$L(\mu, \sigma^2, x_1, ..., x_{100}) = (2\pi\sigma^2)^{-100/2} exp(-\frac{1}{2\sigma^2}\sum_{j=1}^{100}(x_j - \mu)^2)$$

$$l(\mu, \sigma^2, x_1, ..., x_{100}) = -\frac{100}{2}ln(2\pi) - \frac{100}{2}ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{j=1}^{100}(x_j - \mu)^2$$

taking partial derivatives w.r.t. $\mu$ and $\sigma$ and setting them to zero yields the following maximum likelihood estimators for the parameters $\mu$ and $\sigma$.

$$\hat{\mu} = \frac{1}{100}\sum_{j=1}^{100} x_j$$

$$\hat{\sigma^2} = \frac{1}{100}\sum_{j=1}^{100}(x_j - \hat{\mu})^2$$

They are, for the data in provided data file,

```
## [1] "Maximum log-likelihood mean estimator: 1.275528"
```

```
## [1] "Maximum log-likelihood std. deviation estimator: 2.005976"
```

The gradients for the minus log-likelihood function for a hundred observations are

$$-\frac{\partial l}{\partial \mu} = -\frac{1}{\sigma^2}\sum_{i=1}^{100}(x_i - \mu)$$

$$-\frac{\partial l}{\partial \sigma} = -\left(\frac{100}{\sigma} + \frac{1}{\sigma^3}\sum_{i=1}^{100}(x_i - \mu)^2\right)$$

I imagine it is better to try to numerically maximize the log-liklihood function due to the shape of its cruve, which is more conducive to, for example, Newton's method. The likelihood function is likely flat for the most part and a single very sharp spike in its function, which may cause numerical algorithms to not converge to the maximum of the function.

Let us try to optimize the minus logliklihood function with initial parameters $\mu = 0$ and $\sigma = 1$ using the `optim` function. We try the two methods `BFGS` and `CG` with gradients specified and without gradients specified and compare the outcomes.

```
## [1] "BFGS method without gradient specified"
```

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

```
## [1] "Conjugate gradient method without gradient specified"
```

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

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

```
## [1] "BFGS method with gradient"
```

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

```
## [1] "Conjugate gradient method with gradient specified"
```

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

It seems in all cases the algorithm converges toward the maximum likelihood parameter estimates for $\mu = 0$ and $\sigma = 1$. We see that specifying the gradient makes no difference to the `BFGS` optimizer. For the `CG` optimizer the number of iterations are reduced. I would reccomend that we use the BFGS method with gradient provided if possible since it converges in the least number of iterations.

## Appendix

### R code

```r
data <- read.csv2("mortality_rate.csv")
#head(data)
data[,3] <- log(data$Rate)
names(data)[3] <- "LMR"
n = dim(data)[1]
set.seed(123456)
id = sample(1:n, floor(n * 0.5))
train = data[id,]
test = data[-id,]

myMSE <- function(lambda,pars){
  lel <- unlist(pars)
  frame <- data.frame(X = lel[1:68],Y = lel[69:136], Xtest = lel[137:204], Ytest = lel[205:272])
  model1 <- loess(Y ~ X, data = frame, enp.target = lambda)
  pred <- predict(model1, newdata = frame[,3])
  #print(pred)
  predMSE <- mean((pred - frame[,4])^2)
  #print(c("The MSE of loess fit is: ",signif(predMSE,4)))
  return(predMSE)
}
pars <- list(X=train$Day,Y=train$LMR,Xtest=test$Day,Ytest=test$LMR)
res <-myMSE(0.1,pars)

lambvec <- seq(from=0.1,to=40,by=0.1)
resvec <- c()
for(i in 1:length(lambvec)){
  resvec <- c(resvec,myMSE(lambvec[i],pars))
}
plot(lambvec,resvec,main="fitted model MSE vs lambda",xlab ="lambda",ylab = "MSE")
paste("minimum MSE is:",round(resvec[which.min(resvec)],3))
paste("Corresponding optimal value of lambda is:",round(lambvec[which.min(resvec)],3))
op1<-optimize(myMSE,c(0.1,40),pars=pars, tol= 0.01) # 18 iterations required
op1
op2 <- optim(35,myMSE,method = "BFGS",pars=pars) # 3 iterations but it is worse value
op2
load("data.RData")
muhat <- sum(data)/100
paste("Maximum log-likelihood mean estimator:",round(muhat,6))
sigmasquarehat <- sum((data - muhat)^2) / 100
sigmahat <- sqrt(sigmasquarehat)
paste("Maximum log-likelihood std. deviation estimator:",round(sigmahat,6))
minusloglike <- function(pars){
  mu <- pars[1]
  sig <- pars[2]
  loglike <- - (-50*log(2*pi) - 50*log(sig^2) - 1/(2*sig^2)*sum((data- mu)^2))
  return(loglike)
}

gradient <- function(pars){
  mu <- pars[1]
  sig <- pars[2]
  c(-(1/sig^2*sum(data - mu)), -(-100/sig + 1/sig^3*sum((data- mu)^2)))
}
```

```r
oplog <- optim(c(0,1), fn = minusloglike, method = "BFGS")
paste("BFGS method without gradient specified")
oplog

oplog2 <- optim(c(0,1), minusloglike, method = "CG")
paste("Conjugate gradient method without gradient specified")
oplog2

oploggrad <- optim(c(0,1), fn = minusloglike, method = "BFGS", gr = gradient)
paste("BFGS method with gradient")
oploggrad

oplog2grad <- optim(c(0,1), minusloglike, method = "CG", gr = gradient)
paste("Conjugate gradient method with gradient specified")
oplog2grad
## NA
```