Computational Statistics

Lab 2

Emil K Svensson and Rasmus Holm

2017-02-01

Question 1

1.1

```
mort <- read.csv2("../data/mortality_rate.csv")

mort$LMR <- log(mort$Rate)

n <- dim(mort)[1]
set.seed(123456)
id <- sample(1:n, floor(n*0.5))
train <- mort[id, ]
test <- mort[-id, ]
```

1.2

```
myMSE <- function(lambda, pars){
    data  <- data.frame(pars$X, Y = pars$Y)
    model <- loess(formula = Y ~ ., data = data, enp.target = lambda)

    MSE <- mean((pars$Ytest - predict(model,pars$Xtest))^2)
    MSEcounter <<- MSEcounter + 1
    return(MSE)
  }
```

1.3

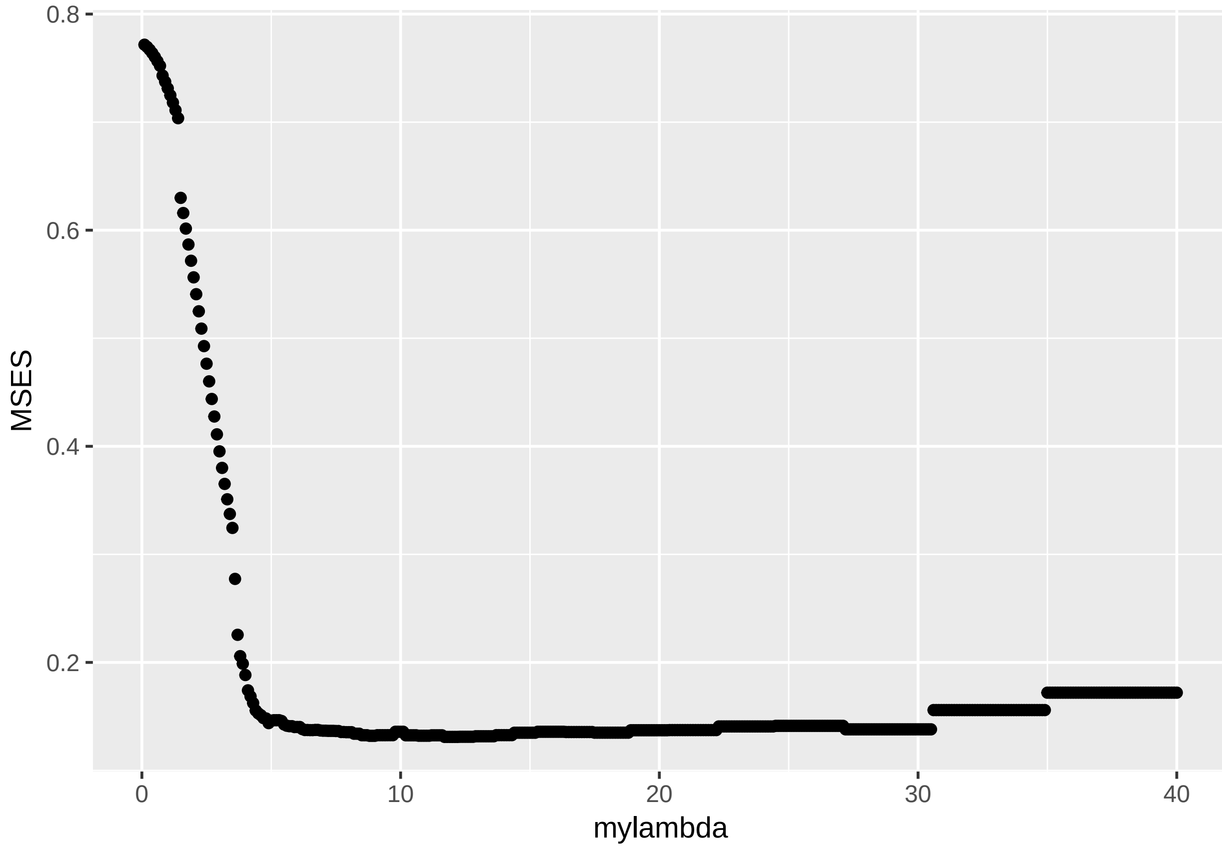
```
MSEcounter <- 0
mylambda <- seq(0.1,40, by = 0.1)
mypars <- list(X = train$Day, Y = train$LMR, Xtest = test$Day, Ytest = test$LMR)


MSES<- sapply(mylambda, FUN = myMSE, pars = mypars)
```

1.4

```
library(ggplot2)

ggplot() + geom_point(aes(x=mylambda, y = MSES))
```




```r
mylambda[which.min(MSES)]
```

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

```r
length(mylambda)
```

```
## [1] 400
```

```r
# Don't understad the question regarding number of evaluations that were required?
# didn't use optimize
```

The optimal value for lambda is 11.7 where the minimum MSE is achived. The number of evaluations required were for this tast 400, the number of lambdas that we tried.

1.5

```r
MSEcounter <- 0
myopt <- optimize(myMSE, lower = 0.1, upper = 40, tol = 0.01, pars = mypars)

paste("The number of evaluations:", MSEcounter)
```

```
## [1] "The number of evaluations: 18"
```

No, the optimize-function fails to find the minimum MSE and identifies it as 10.69 because of the small bump around lambda = 10 it think it has found the local minimum.

The number of evaluations are lower then in the previous question though. (18 compared to 400)

1.6

```r
MSEcounter <- 0

optim(par=list(lambda=35), fn = myMSE, method = "BFGS", pars=mypars)$par
```

```
## lambda
##     35
```

```r
paste("The number of evaluations:", MSEcounter)
```

```
## [1] "The number of evaluations: 3"
```

The optimal lambda here was as we specified lambda = 35, this is because the MSE around lambda 35 is a platue and therefor the gradient (first derviative) becomes zero and the algorithm stops since there is no change.

Question 2

2.1

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

mean(data)
```

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

```r
var(data)
```

```
## [1] 4.064587
```

2.2

Derv

2.3

```r
negLog <- function(x, data){
    mu <- x[1]
    sigma <- x[2]
    n <- length(data)
    loglik <- (n / 2) * log(sigma^2) + (n / 2) * log(2 * pi) + (1 / (2 * sigma^2)) * sum((data - mu)^2)
    return(loglik)
}

negLogGradient <- function(x, data) {
    mu1 <- sum(data) / length(data)
    c(-mu1, -(1 / length(data)) * sum((data - mu1)^2))
}

optim(par =c(0, 1), fn = negLog, method = "BFGS", data = data)
```

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

```r
optim(par =c(0, 1), fn = negLog, gr = negLogGradient, method = "BFGS", data = data)
```

```
## $par
## [1] 1.275528 5.023942
##
## $value
## [1] 261.2867
##
## $counts
## function gradient
##       25        2
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```r
optim(par =c(0, 1), fn = negLog, method = "CG", data = data)
```

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

```r
optim(par =c(0, 1), fn = negLog, gr = negLogGradient, method = "CG", data = data)
```

```
## $par
## [1] 0.6377638 3.0119708
```

```
##
## $value
## [1] 226.573
##
## $counts
## function gradient
##       75        5
##
## $convergence
## [1] 0
##
## $message
## NULL
```