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) {
    MSEcounter <<- MSEcounter + 1

    data  <- data.frame(pars$X, Y=pars$Y)
    model <- loess(formula=Y ~ ., data=data, enp.target=lambda)
    mean((pars$Ytest - predict(model, pars$Xtest))^2)
  }
```

1.3

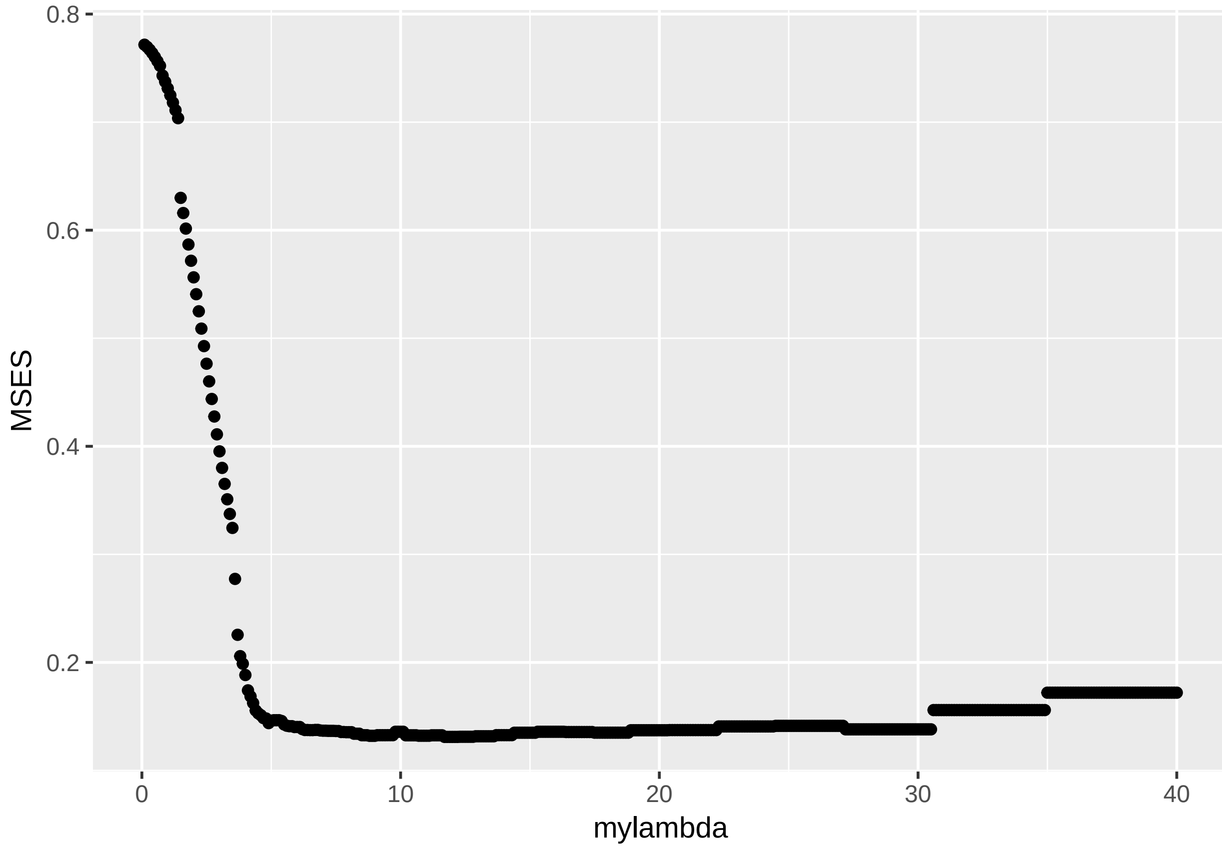
```
MSEcounter <- 0
mylambda <- seq(0.1, 40, by=0.1)
mypars <- list(X=train$Day, Y=train$LMR, Xtest=test$Day, Ytest=test$LMR)

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

1.4

```
library(ggplot2)

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




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

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

```
paste("Number of evaluations", length(mylambda))
```

```
## [1] "Number of evaluations 400"
```

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

1.5

```
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

```
MSEcounter <- 0

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

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

```
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 plateau and therefor the gradient (first derivative) becomes zero and the algorithm stops since there is no change.

Question 2

2.1

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

2.2

Derv

2.3

```
negLog <- function(x, data){
    negLogC <<- negLogC + 1

    mu <- x[1]
    sigma <- x[2]
    n <- length(data)
    (n / 2) * log(sigma^2) + (n / 2) * log(2 * pi) + (1 / (2 * sigma^2)) * sum((data - mu)^2)
}

negLogGradient <- function(x, data) {
    negLogGC <<- negLogGC + 1

    mu1 <- sum(data) / length(data)
    c(-mu1, -(1 / length(data)) * sum((data - mu1)^2))
}
```

The reason why choosing the log-likelihood instead of the likelihood is because of numerical precision. Since we are calculating small probabilities we don't want to multiply them and get even smaller numbers. When taking the log-likelihood we instead add them together.

```
negLogC <- 0
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
paste("The number of evaluations of the log-likelihood:", negLogC)
```

```
## [1] "The number of evaluations of the log-likelihood: 97"
```

```r
negLogC <- 0
negLogGC <- 0
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
paste("The number of evaluations of the gradient:", negLogGC)
```

```
## [1] "The number of evaluations of the gradient: 2"
```

```r
paste("The number of evaluations of the log-likelihood:", negLogC)
```

```
## [1] "The number of evaluations of the log-likelihood: 25"
```

```r
negLogC <- 0
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
paste("The number of evaluations of the log-likelihood:", negLogC)
```

```
## [1] "The number of evaluations of the log-likelihood: 312"
```

```r
negLogC <- 0
negLogGC <- 0
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
```

```r
paste("The number of evaluations of the gradient:", negLogGC)
```

```
## [1] "The number of evaluations of the gradient: 5"
```

```r
paste("The number of evaluations of the log-likelihood:", negLogC)
```

```
## [1] "The number of evaluations of the log-likelihood: 75"
```

1.4

All optimizations converged. The results can be seen above in the par-variable for each print out where the first value is μ and the second value is σ^2. The true values are $\mu = 1.2755276$, $\sigma^2 = 4.0645875$. The closet comparable result was generated from BFGS with the gradient specified, the mean is near exactly the same and the variance is a bit of but still the closest. It is also the one using least iterations, so clearly it seems like the superior technique in this example.