Lab 2 - Computational Statistics

Gustav Sternelöv

8 februari 2016

Assignment 1

1.1-1.2

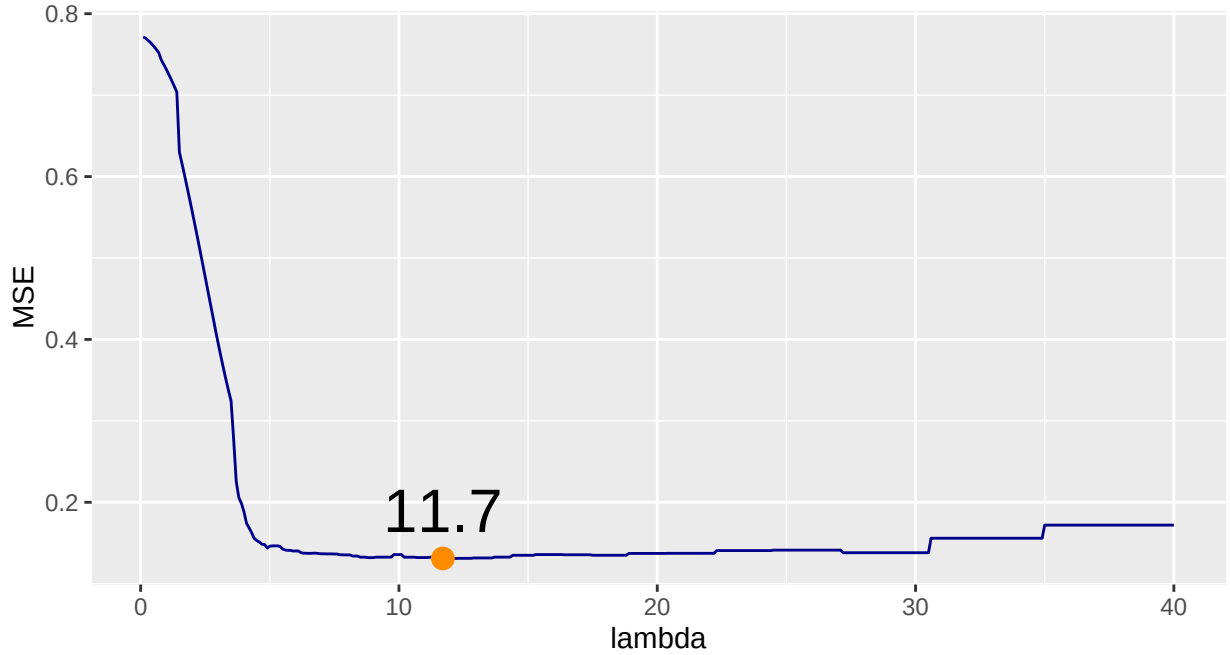
Data is imported and splitted into a training set and a test set with each set containing 50 % of the original set.

Thereafter, the function *myMSE* is written. It takes the arguments *pars* (a list including *X*, *Y*, *XTest* and *YTest*) and the argument *lambda*. The latter parameter specifies the penalty parameter *enp.target* in *loess*. The *myMSE* function then uses *loess* to make a model which is evaluated on the test set. Returned by *myMSE* is the predictive MSE.

```
myMSE <- function(lambda, pars){
  model <- loess(pars$Y~pars$X, data=pars[1:2], enp.target = lambda)
  Pred <- predict(model, newdata=pars$Xtest)
  MSE <- (1/length(pars$Y)) * sum((Pred-pars$Ytest)^2)
  print(MSE)
  return(MSE)
}
```

1.3

A plot of the predictive MSE values against the respective *lambda* values in the range 0.1 to 40 by steps of 0.1 is shown below.



The lowest predictive MSE value, 0.131047, is obtained when *lambda* is equal to 11.7, so that is the optimal

lambda value. Since the predictive MSE has been calculated for all values from 0.1 to 40 by steps of 0.1, 400 evaluations of *myMSE* were required to find the optimal value for *lambda*.

1.4

The function *optimize* is an alternative approach for finding the optimal MSE value. It uses golden section search as the optimizing method. The output given by the function is shown below and when comparing to the values given in *1.3* it is noted that the function not managed to find the optimal MSE value. However, the difference between both the optimal MSE value and the *lambda* value is small compared to optimal values obtained in *1.3*.
By counting the prints it is concluded that 18 *myMSE* evaluations where needed to find the value thought to optimal. That is just 4.5 % of the evaluations required for the simple approach method applied in *1.3*

```
## [1] 0.1358018
## [1] 0.1412809
## [1] 0.1326581
## [1] 0.1401702
## [1] 0.1325391
## [1] 0.1321441
## [1] 0.1325542
## [1] 0.1321441
## [1] 0.1325391
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1321441
```

```
## $minimum
## [1] 10.69361
##
## $objective
## [1] 0.1321441
```

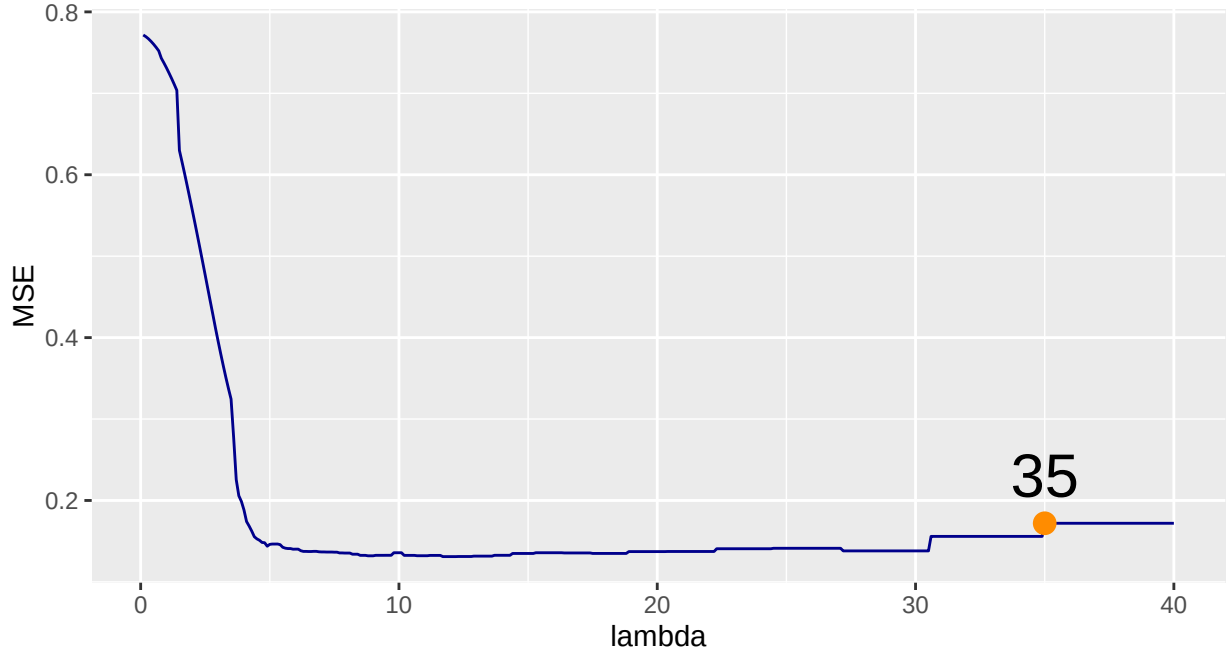
1.5

The next method examined is the *BFGS* method via the function *optim*. The starting point is chosen to be *lambda* equal to 35 and the objective is to investigate whether the optimal *lambda* value is found from that starting point.
The output below gives that three evaluations of *myMSE* were conducted before it was concluded that the optimal *lambda* value was found. Although, it has to be noted that chosen value for *lambda* is exactly the same as the starting value, which is quite far from the optimal values chosen by the simple approach and the golden section search.

```
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
```

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

The given results can be explained by a look of the plot over the MSE values for all *lambda* values in the range 0.1 to 40.



At the chosen starting point is the slope of the curve for the predictive MSE values zero, so the point is a local minimum. The implications of starting at a local minimum when using the *BFSG* method is that it not will keep on iterate and look if there is a better solution. Instead it will just investigate the MSE values closest to the starting point and discover that it already is at a local minima and as a result of this stay at *lambda* equal to 35.

Assignment 2

2.1-2.2

The probability density function for the normal distribution:

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

For a sample of n observations that are independent and identically distributed the likelihood is:

$$\prod f(x_i|\mu,\sigma) = (\frac{1}{\sqrt{2\pi\sigma^2}})^n exp(-\frac{\sum_{i=1}^n (x_i-\mu)^2}{2\sigma^2})$$

Then, the log likelihood is:

$$log(L(\mu,\sigma)) = nlog(\frac{1}{\sqrt{2\pi\sigma^2}}) + (-\frac{\sum_{i=1}^n (x_i-\mu)^2}{2\sigma^2})$$

The derivatives of the log likelihood is computed to obtain the maximum likelihood estimators for μ and σ. First the ML estimator for μ:

$$\frac{\partial log(L(\mu,\sigma))}{\partial \mu} = 0 + \frac{2\sum_{i=1}^n (x_i-\mu)}{2\sigma^2}$$

$$\hat{\mu} = \overline{x} = \frac{\sum_{i=1}^n x_i}{n}$$

Then, finally, the ML estimator for σ:

$$\frac{\partial log(L(\mu,\sigma))}{\partial \sigma} = -\frac{n}{2} * \frac{2}{\sigma} + 2\sigma * (-2\sigma^2)^{-2} * (-2\sum_{i=1}^n (x_i-\mu))$$

$$\hat{\sigma} = \sqrt{\frac{\sum_{i=1}^n (x_i-\mu)^2}{n}}$$

The parameter estimates obtained for μ and σ when the formulas above are used on the data set is 1.2755276 for the mean and 2.0059765 for the standard deviation.

2.3

Often more convenient to work with the log likelihood
An MLE estimate is the same regardless of whether the likelihood or log likelihood is maximized.

It is a bad idea to maximize the likelihood rather than maximizing the log likelihood because...

2.4-2.5

Conjugate gradient method without gradient specified.

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##      169       31
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Conjugate gradient method with gradient specified.

```
## $par
## [1] 2.335219 2.557265
##
## $value
## [1] 225.1394
##
## $counts
## function gradient
##       69        7
##
## $convergence
## [1] 0
##
## $message
## NULL
```

BFGS method without gradient specified.

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       40       15
##
## $convergence
## [1] 0
##
## $message
## NULL
```

BFGS method with gradient specified

```
## $par
## [1] 1.596806 6.383924
##
## $value
## [1] 282.3356
##
## $counts
## function gradient
##       77        7
##
## $convergence
## [1] 0
##
## $message
## NULL
```

In all of the cases did the algorithms converge.
The optimal value shifted between the different algorithms. For both the conjugate gradient and the BFGS method without specified gradient the optimal values were 1.275528 for μ and 2.005977 for σ. The conjugate gradient with specified gradient had 2.335219 and 2.557265 as optimal values and the BFGS had 1.596806 and 6.383924.
The amount of function and gradient evaluations required for the respective settings can be seen in the outputs above. A conjugate gradient optimizer without gradient specified required the highest amount of both function and gradient evaluations. The algorithms with specified gradient required the lowest amount of gradient evaluations and BFGS method without specified gradient requierd the lowest amount of function evaluations.
Should then recommend the BFGS method without gradient specified since that algorithm finds the optimal values and requires the lowest amount of function evaluations and a relatively low amount of gradient evaluations.

Appendix

```
mort_rate<-read.csv2("C:\\Users\\Gustav\\Documents\\Computational-Statistics\\Lab2\\mortality_rate.csv"
mort_rate$LMR <- log(mort_rate$Rate)

n=dim(mort_rate)[1]
set.seed(123456)
id=sample(1:n, floor(n*0.5))
train=mort_rate[id,]
test=mort_rate[-id,]
myMSE <- function(lambda, pars){
  model <- loess(pars$Y~pars$X, data=pars[1:2], enp.target = lambda)
  Pred <- predict(model, newdata=pars$Xtest)
  MSE <- (1/length(pars$Y)) * sum((Pred-pars$Ytest)^2)
  print(MSE)
  return(MSE)
}
Data <- list(X=train$Day, Y=train$LMR, Xtest=test$Day, Ytest=test$LMR)
# 1.3
Lambda <-  seq(0.1,40, by=0.1)
modelMSE <- 0
j <- 0
for(i in Lambda){
  j <- j+1
  modelMSE[j] <- myMSE(Lambda[j], Data)
}
library(ggplot2)
MSEdata <- data.frame(MSE=modelMSE, lambda=Lambda)
ggplot(MSEdata, aes(x=lambda, y=MSE, label=lambda)) + geom_line(col="darkblue")+
  geom_text(data=subset(MSEdata, MSE <= min(MSEdata$MSE))[1,],vjust=-0.6, size=8) +
  geom_point(data=subset(MSEdata, MSE <= min(MSEdata$MSE))[1,], size=3.5, col="darkorange")
Optim <- optimize(f=myMSE, interval = c(0.1, 40), tol=0.01, pars=Data)
Optim
optim(par=35, fn=myMSE, method = "BFGS", pars=Data)
ggplot(MSEdata, aes(x=lambda, y=MSE, label=lambda)) + geom_line(col="darkblue")+
  geom_text(data=subset(MSEdata,  MSEdata$lambda == 35),vjust=-0.6, size=8) +
  geom_point(data=subset(MSEdata,  MSEdata$lambda == 35), size=3.5, col="darkorange")
```

```r
# 2.1
load("C:\\Users\\Gustav\\Documents\\Computational-Statistics\\Lab2\\data.RData")
Data2 <- data
# Maximum likelihood estimates
# my
MeanEst <- sum(Data2) / length(Data2)
# sigma
sigmaEst <- sqrt((1/length(Data2)) * sum((Data2-MeanEst)^2))

# 2.4
# minus log-like function
loglikef <- function(par,...){
  my <- par[1]
  sigma <- par[2]
  n <- length(data)
  loglike <- -(n*log(1/(sqrt(2*pi*sigma^2))) - (sum((data-my)^2)/(2*sigma^2)))
  return(loglike)
}

gradientf <- function(par,...){
  n <- length(data)
  my <- par[1]
  sigma <- par[2]
  myML <-  - (2*sum(data-my)) / (2*sigma^2)
  mySigma <- - (n/2)*(2/sigma)+ 2*sigma - (2*sigma^2)^-2 * (-2*sum(data-my))
  return(c(myML, mySigma))
}
optim(par = c(0,1), fn=loglikef, method = "CG", data=Data2)
optim(par = c(0,1), fn=loglikef, gr=gradientf,  method = "CG", data=Data2)
optim(par = c(0,1), fn=loglikef , method = "BFGS", data=Data2)
optim(par = c(0,1), fn=loglikef, gr=gradientf,  method = "BFGS", data=Data2)
## NA
```