Computational Statistics: Computer lab 2.

Laura Julià Melis

2/07/2020

Question 1: Optimizing a model parameter

The file `mortality_rate.csv` contains information about mortality rates of the fruit flies during a certain period.

1. Import this file to `R` and add one more variable `LMR` to the data which is the natural logarithm of `Rate`. Afterwards, divide the data into training and test sets by using the following code:

```
##### QUESTION 1 #####
# 1.1. Importing data, adding LMR and dividing the dataset
data <- read.table("mortality_rate.csv", header= TRUE, sep=";", dec=",")
data$LMR <- log(data$Rate)

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

2. Write your own function `myMSE()` that for given parameters λ and list `pars` containing vectors `X`, `Y`, `Xtest`, `Ytest` fits a LOESS model with response Y and predictor X using loess() function with penalty λ (parameter `enp.target` in `loess()`) and then predicts the model for `Xtest`. The function should compute the predictive MSE, print it and return as a result. The predictive MSE is the mean square error of the prediction on the testing data. It is defined by the following Equation (for you to implement):

$$\text{predictive MSE} = \frac{1}{\text{length(test)}} \sum_{\text{ith element in test set}} (\text{Ytest[i]} - \text{fYpred(X[i])})^2$$

where `fYpred(X[i])` is the predicted value of `Y` if `X` is `X[i]`. Read on R's functions for prediction so that you do not have to implement it yourself.

```
# 1.2. Creating function myMSE.
myMSE <- function(lambda, pars){
  pred <- vector(length = length(lambda))
  for(i in 1:length(lambda)){
    fit <- loess(Y ~ X, pars, enp.target = lambda[i])
    fYpred <- predict(fit, data=pars$X)
    pred[i] <- (1/length(pars$Xtest)) * (sum( (pars$Ytest - fYpred)^2 ))
  }
  return(pred)
}
```

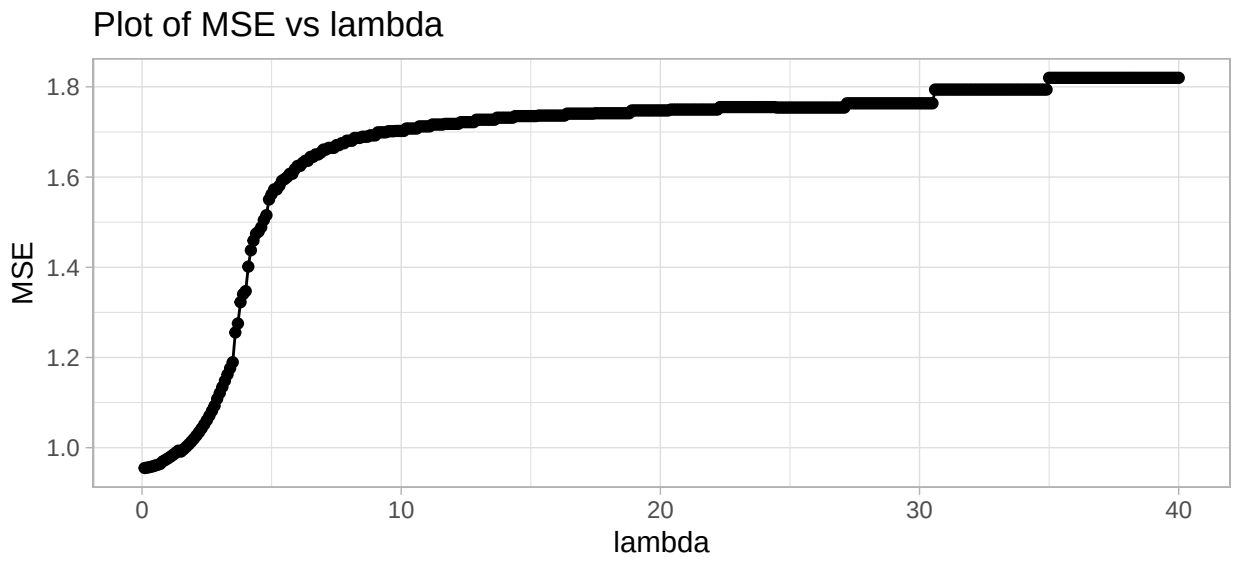
3. Use a simple approach: use function `myMSE()`, training and test sets with response LMR and predictor Day and the following λ values to estimate the predictive MSE values: $\lambda = 0.1, 0.2, ..., 40$

```
# 1.3. Estimating predictive MSE values.
lambda <- seq(0.1, 40, by= 0.1)
```

```
pars <- list(X=train$Day , Y=train$LMR , Xtest=test$Day , Ytest=test$LMR)
mse <- myMSE(lambda, pars)
```

4. Create a plot of the MSE values versus λ and comment on which λ value is optimal. How many evaluations of `myMSE()` were required (read `?optimize`) to find this value?

- Plot:




```
## The optimal lambda is: 0.1 and the minimum MSE value is: 0.9549996
```

The function `myMSE()` has been evaluated for all the 400 different values of λ. We haven't set any tolerance to indicate the MSE that it should stop if the two last MSE computations are "too close".

5. Use `optimize()` function for the same purpose, specify range for search [0.1,40] and the accuracy 0.01. Have the function managed to find the optimal MSE value? How many `myMSE()` function evaluations were required? Compare to step 4.

```
## $minimum
## [1] 0.10376
##
## $objective
## [1] 0.9550269
```

6. Use optim() function and BFGS method with starting point $\lambda = 35$ to find the optimal λ value. How many `myMSE()` function evaluations were required (read `?optim`)? Compare the results you obtained with the results from step 5 and make conclusions.

```
## $par
## [1] 35
##
## $value
## [1] 1.819934
##
## $counts
## function gradient
##        1        1
##
## $convergence
## [1] 0
```

```
##
## $message
## NULL
```

Question 2: Maximizing likelihood

The file data.RData contains a sample from normal distribution with some parameters μ, σ. For this question read `?optim` in detail.

1. Load the data to R environment.

See appendix to see the code

2. Write down the log-likelihood function for 100 observations and derive maximum likelihood estimators for μ, σ analytically by setting partial derivatives to zero. Use the derived formulae to obtain parameter estimates for the loaded data.

Given an i.i.d. sample $(X_1, \ldots, X_n)$ of $n = 100$ observations that comes from a normal distribution, $N(\mu, \sigma)$, its probability distribution is:

$$P(X|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 \right]$$

And the log-likelihood function is the joint probability of all the 100 observations is:

$$L(\mu,\sigma) = P(X_1,\ldots,X_{100}|\mu,\sigma) = \prod_{i=1}^{100} P(X_i|\mu,\sigma) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \cdot \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{100}(x_i-\mu)^2 \right]$$

So now we have to do the logarithm of the log-likelihood function:

$$\log(L(\mu,\sigma)) = \log\left(\left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \cdot \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{100}(x_i-\mu)^2 \right]\right) = \log\left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n - \frac{1}{2\sigma^2}\sum_{i=1}^{100}(x_i-\mu)^2\log(e) =$$

$$= n\left(\log\left(\frac{1}{\sigma}\right) + \log\left(\frac{1}{\sqrt{2\pi}}\right)\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{100}(x_i-\mu)^2 = n\big(\log(\sigma)^{-1} + \log(2\pi)^{-1/2}\big) - \frac{1}{2\sigma^2}\sum_{i=1}^{100}(x_i-\mu)^2.$$

So, the log-likelihood function for 100 observations is as follows:

$$\boxed{\log(L(\mu,\sigma)) = -n\log(\sigma) - \frac{n}{2}\log(2\pi) - \frac{1}{2\sigma^2}\sum_{i=1}^{n=100}(x_i-\mu)^2}$$

Now, to find the maximum likelihood estimators for μ and σ, we will derivate $\log(L(\mu,\sigma))$, set the equations to 0 and isolate the parameters:

- For μ:

$$(1)\quad \frac{\partial\log(L(\mu,\sigma))}{\partial\mu} = -0 - 0 - \left(0 - \frac{1}{2\sigma^2}2\sum_{i=1}^{100}(x_i-\mu)(0-1)\right) = +\frac{1}{\sigma^2}\sum_{i=1}^{100}(x_i) - n\mu$$

$$(2)\quad \frac{1}{\sigma^2}\sum_{i=1}^{100}(x_i) - n\mu = 0 \to \sum_{i=1}^{100}(x_i) - n\mu = 0 \to n\mu = \sum_{i=1}^{100}x_i \to \boxed{\mu_{ML} = \frac{1}{100}\sum_{i=1}^{100}x_i}$$

- For σ:

$$(1) \quad \frac{\partial \log(L(\mu,\sigma))}{\partial \sigma} = -0 - \frac{n}{\sigma} - 0 - \frac{0-4\sigma}{4\sigma^4}\sum_{i=1}^{100}(x_i-\mu) = -\frac{n}{\sigma} + \frac{1}{\sigma^3}\sum_{i=1}^{100}(x_i-\mu)^2$$

$$(2) \quad -\frac{n}{\sigma} - \frac{1}{\sigma^3}\sum_{i=1}^{100}(x_i-\mu)^2 = 0 \rightarrow n\sigma^3 = -\sigma\sum_{i=1}^{100}(x_i-\mu)^2 \rightarrow \frac{\sigma^3}{\sigma} = \frac{\sum_{i=1}^{100}(x_i-\mu)^2}{n} \rightarrow \boxed{\sigma_{ML} = \sqrt{\frac{\sum_{i=1}^{100}(x_i-\mu)^2}{n}}}$$

Finally, we will use these formulae to obtain the maximum likelihood estimations for the data (*See appendix to see the implementation with R*).

The results obtained are:

```
## The estimated mu is: 1.275528 and the estimaded sigma is 2.005976
```

3. Optimize the minus log–likelihood function with initial parameters $\mu = 0, \sigma = 1$. Try both Conjugate Gradient method (described in the presentation handout) and BFGS (discussed in the lecture) algorithm with gradient specified and without. Why it is a bad idea to maximize likelihood rather than maximizing log–likelihood?

To optimize the minus log–likelihood we want to minimize it (which is the same than maximizing the log-likelihood). To do so, we have created a function that computes the log–likelihood and then we have used it in the `optim()` function: once with `method="CG"` and the other one with `method="BFGS"`.

The results obtained are:

- For the Conjugate Gradient method:
- For the BFGS algorithm:

It is better to maximize the log-likelihood rather than the likelihood function itself because...

https://math.stackexchange.com/questions/892832/why-we-consider-log-likelihood-instead-of-likelihood-in-gaussian-distributi

4. Did the algorithms converge in all cases? What were the optimal values of parameters and how many function and gradient evaluations were required for algorithms to converge? Which settings would you recommend?

Appendix

```
##### QUESTION 1 #####
# 1.1. Importing data, adding LMR and dividing the dataset
data <- read.table("mortality_rate.csv", header= TRUE, sep=";", dec=",")
data$LMR <- log(data$Rate)

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

# 1.2. Creating function myMSE.
myMSE <- function(lambda, pars){
  pred <- vector(length = length(lambda))
  for(i in 1:length(lambda)){
    fit <- loess(Y ~ X, pars, enp.target = lambda[i])
    fYpred <- predict(fit, data=pars$X)
    pred[i] <- (1/length(pars$Xtest)) * (sum( (pars$Ytest - fYpred)^2 ))
  }
  return(pred)
}

# 1.3. Estimating predictive MSE values.
lambda <- seq(0.1, 40, by= 0.1)
pars <- list(X=train$Day , Y=train$LMR , Xtest=test$Day , Ytest=test$LMR)
mse <- myMSE(lambda, pars)

# 1.4. Plot of MSE vs lambda
library(ggplot2)
df <- data.frame(lambda= lambda, MSE = mse)

ggplot(df, aes(x=lambda, y=MSE)) + geom_point() + geom_line() +
  ggtitle("Plot of MSE vs lambda") + theme_light()
# 1.4.a. Minimum MSE and optimal lambda
cat ("The optimal lambda is:", df$lambda[which(min(df$MSE)== df$MSE)],
     "and the minimum MSE value is:", min(df$MSE))

# 1.5. Finding optimal MSE with optimize()
optimize(myMSE, c(0.1,40), tol=0.01, pars=pars)

# 1.6. Finding optimal lambda with BFGS method.
optim(par=35, fn = myMSE, method= "BFGS", pars=pars)


##### QUESTION 2 #####
# 2.1. Loading data
load("data.RData")

# 2.2. Use the derived formulae to obtain parameter estimates for the loaded data
n <- length(data)
mu_ml <- (1/n) * sum(data)
sigma_ml <- sqrt( sum( (data-mean(data))^2 ) / n )
```

```r
cat("The estimated mu is:", mu_ml, "and the estimaded sigma is", sigma_ml)

# 2.3. Optimize the minus log-likelihood function
## A. Writing function that calculates the log-likelihood
llik <- function(x, data){
  n <- length(data)
  mu <- x[1]
  sigma <- x[2]
  result <- ( -n*log(sigma) ) - ( (n/2)*log(2*pi) ) - ( sum( (data-mu)^2 ) / (2*sigma^2) )
  return(result)
}

## B. Using optim()
### Conjugate gradient method:
# cg <- optim(c(0,1), fn = llik, method= "CG", data=data)

### BFGS method:
# bfgs <- optim(par=c(0,1), fn = llik, method= "BFGS", data=data)
```