Computational Statistics - Lab 02

Maximilian Pfundstein

2019-01-24

Contents

1 Question 1: Optimizing a Model Param

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

Task: 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:

Day	Rate	LMR
1	0.0014	-6.571283
2	0.0040	-5.521461
3	0.0051	-5.278515
4	0.0064	-5.051457
5	0.0075	-4.892852
6	0.0098	-4.625373

Task: 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):

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

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.

```r
myMSE = function(lambda, pars) {
  model = loess(Y ~ X, data=pars, enp.target = lambda)
  prediction = predict(model, data = pars$Xtest)
  mse = sum(((prediction - pars$Ytest)^2))/length(pars$Ytest)
  #print(mse) Honselty I don't want to spam my console!
  #print(".")
  return(mse)
}
```

Task: 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: $\lambda = 0.1, 0.2, ...40$.

```
pars = list(X = train$Day, Y = train$LMR, Xtest = test$Day, Ytest = test$LMR)
lambdas = seq(from = 0.1, to = 40, by = 0.1)
mses = sapply(X = lambdas, FUN = myMSE, pars = pars)
```

Task: 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?

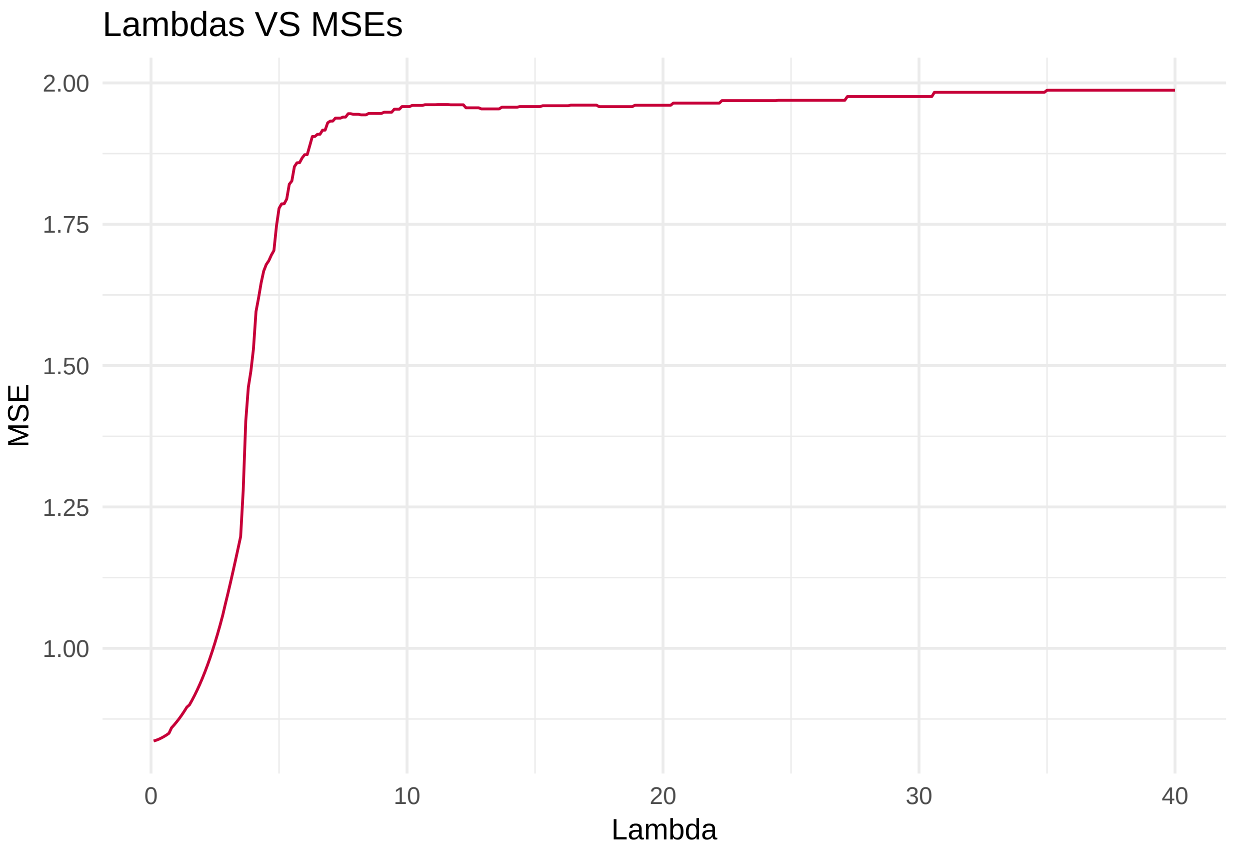
Answer: It looks like the smaller the λ the lower the MSE, so

```
## [1] 0.1
```

is best λ (for the given input sequence). The function was called

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

times.



Task: 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.

Answer: For the build in `optimize()` function we get the following results:

```
optimize(myMSE, tol = 0.01, interval = c(0.1, 40), pars = pars)
```

```
## $minimum
## [1] 0.1038627
##
## $objective
## [1] 0.8361134
```

The function `myMSE()` was called 21 times. We counted the printed dots, we were to lazy to build a wrapper with a counter :).

Task: 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.

Answer: Only `3` iteration were used, which is lower than the number of calls we had before. We therefore think that this general-purpose optimization based on Nelder–Mead works most efficient, at least for this case. The function `optimize()` seaches the whole interval using the golden selection search (desciption on the course slides), so it is looking for a local minuma in the given interval. It uses a constant reduction factor α. For this to work the function has to be unimodal, which means that is onl has one maxima/minima (in the given interval). So it's pretty basic and depending on α we need more or less iterations.

Newtons Method (Nelder-Mead) is memory heavy but therefore converges quickly. It computes an approximation for the Hessian and calculates the quasi–Newton condition or secant condition:

$$B_{k+1}(\vec{x}_{k+1} - \vec{x}_k) = \nabla f(\vec{x}_{k+1}) - \nabla f(\vec{x}_k)$$

We want the current and the next itertion to be as close as possible. The computations for this can be found on the slides and won't be mentioned here. One last word about the BGFS (Broyden–Fletcher–Goldfarb–Shanno), which was used here. It needs more iterations than Newton (but apparently still only a small amount), therefore each iteration is faster to compute, so it's actually better for large scale problems. The gradient in this case is calculated like this:

$$\nabla f(\vec{x}) = A\vec{x} - \vec{b} = r(\vec{x})$$

```r
optim(35, myMSE, method = "BFGS", pars = pars)
```

```
## $par
## [1] 35
##
## $value
## [1] 1.987027
##
## $counts
## function gradient
##        1        1
##
## $convergence
## [1] 0
##
## $message
## NULL
```

2 Question 2: Maximizing Likelihood

3 Source Code

```r
knitr::opts_chunk$set(echo = TRUE, cache = FALSE, include = TRUE, eval = TRUE)
library(ggplot2)
library(knitr)
library(gridExtra)

data = read.csv("mortality_rate.csv", 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 ,]

kable(head(data))


myMSE = function(lambda, pars) {
  model = loess(Y ~ X, data=pars, enp.target = lambda)
  prediction = predict(model, data = pars$Xtest)
  mse = sum(((prediction - pars$Ytest)^2))/length(pars$Ytest)
  #print(mse) Honselty I don't want to spam my console!
  #print(".")
  return(mse)
}


pars = list(X = train$Day, Y = train$LMR, Xtest = test$Day, Ytest = test$LMR)
lambdas = seq(from = 0.1, to = 40, by = 0.1)
mses = sapply(X = lambdas, FUN = myMSE, pars = pars)

lambdas[which.min(mses)]
length(lambdas)

df = data.frame(lambdas, mses)

ggplot(df) +
  geom_line(aes(x = lambdas, y = mses), color = "#C70039") +
  labs(title = "Lambdas VS MSEs", y = "MSE", x = "Lambda") +
  theme_minimal()


optimize(myMSE, tol = 0.01, interval = c(0.1, 40), pars = pars)


optim(35, myMSE, method = "BFGS", pars = pars)
```