Computer Lab 2
Computational Statistics

Linköpings Universitet, IDA, Statistik

2017/02/01

Kurskod och namn:	732A90 Computational Statistics
Datum:	2017/01/31—2017/02/10
Delmomentsansvarig:	Krzysztof Bartoszek
Instruktioner:	This computer laboratory is part of the examination for the Computational Statistics course Create a group report, (that is directly presentable, if you are a presenting group) on the solutions to the lab as a **.PDF** file. Be concise and do not include unnecessary printouts and figures produced by the software and not required in the assignments. **All R code should be included as an appendix into your report.** A typical lab report should 2-4 pages of text plus some amount of figures plus appendix with codes. In the report reference **ALL** consulted sources and disclose **ALL** collaborations. The report should be handed in via LISAM (or alternatively in case of problems e–mailed to krzysztof.bartoszek@liu.se), by **23:59 10 February 2017** at latest. The report can be written in English or Swedish. **The presenting groups has to write the report in English!**

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:

```
n=dim(data)[1]
set.seed(123456)
```

```r
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):

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

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.

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: $\lambda = 0.1, 0.2, \ldots, 40$

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?

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.

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.

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.

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.

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?

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?