Lab2 Group11

Group 10

28/01/2020

Question 1: Optimizing a model parameter

```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-uniform
## 'Rounding' sampler used
```

1.

2.

3.

4.

5.

6.

Question 2: Maximizing likelihood

(Started to write question 2.1 - 2.4 / Sofie)

1.

In this task we will use the file `data.RData` consists of a sample coming from normal distribution with parameters μ and σ. First we load the data set into R.

2.

The sample comes from a normal distribution with parameters μ and σ, where we set $\theta = (\mu, \sigma)$. Under the assumption that the sample $\boldsymbol{x} = (x_1, ..., x_{100})$ is iid, i.e. $\boldsymbol{X_i} \overset{iid}{\sim} N(\mu, \sigma^2)$, for $i = 1, ..., 100$, then the joint density function of all $n = 100$ observations can be written as

$$L(\theta; \boldsymbol{x}) = f(\boldsymbol{x}|\theta) = \prod_{i=1}^{100} f(x_i|\theta).$$

Now we let the number of observations be denoted by n in the following derivations. Using the density function of a normal distribution with parameter θ we obtain the likelihood function

$$L(\theta; \boldsymbol{x}) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\left(\frac{x_i - \mu}{\sigma}\right)^2\right\}.$$

The log-likelihood function is given by

$$l(\theta; \boldsymbol{x}) = \log\, L(\theta; \boldsymbol{x}) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2.$$

The maximum likelihood estimators (MLEs) $\hat{\mu}_{ML}$ and $\hat{\sigma}^2_{ML}$ of μ and σ^2 are obtained by maximizing the likelihood function. This is done by differentiating the log-likelihood functions and put them to zero. In more detail, we calculate the score functions $S(\theta; \boldsymbol{x})$ w.r.t. μ and σ seperately, and let them equal zero and solve for each parameter:

$$S(\mu) = \frac{\partial}{\partial\mu}\, l(\theta; \boldsymbol{x}) = -\frac{n(\overline{x} - \mu)}{\sigma^2} = 0,$$

where $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. From this we obtain $\hat{\mu}_{ML} = \overline{x}$. Further,

$$S(\sigma) = \frac{\partial}{\partial\sigma}\, l(\theta; \boldsymbol{x}) = -\frac{n}{\sigma} + \frac{1}{\sigma^3}\sum_{i=1}^{n}(x_i - \mu)^2 = 0,$$

and $\hat{\sigma}^2_{ML} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$.

Then we use the derived formulas in order to obtain the desired parameter estimates for the loaded data. So the data set with 100 observations gives the result $\hat{\mu}_{ML} = 1.275528$ and $\hat{\sigma}_{ML} = 2.005976$.

3.

The function `optim()` minimizes the function by default in R. Thus we will optimize the minus log-likelihood function in order to find the maximum of the function. In Question 2.4 we performed two types of algorithms, Conjugate Gradient and BFGS, both with gradient specified and without, to maximize the log-likelihood function. It is a better idea to maximize the log-likelihood than maximize the likelihood. This is due to the large values that occurs in the likelihood, so it is preferable to take the logarithm which gives us a better scale to work with.

4.

The results of the optimization are presented in Table 1.

Algorithm	Gradient specified	$\hat{\mu}$	$\hat{\sigma}$	Function	Gradient	Time
Conjugate Gradient	No	1.275528	2.005977	297	45	0.01877809 sec
Conjugate Gradient	Yes	1.275528	2.005976	53	17	0.004215956 sec
BFGS	No	1.275528	2.005977	37	15	0.005324841 sec
BFGS	Yes	1.275528	2.005977	39	15	0.009279966 sec

Table 1: *Result of the optimization using the algorithms Conjugate Gradient and BFGS, for the given data set. The algorithms, gradient, optimal values of parameters, number of function and gradient evaluations and time taken are presented.*

From the results in Table 1, we can see that the algorithm converged in all cases, where the obtained optimal values correspond to the estimated parameters in Question 2.2.

Without specifying the gradient, the Conjugate Gradient method required 297 function and 45 gradient evaluations for the algorithm to converge, while the BFGS only required 37 function and 15 gradient evaluations. Also, the time until convergence was measured by using `Sys.time()`, and we can notice that the BFGS is somewhat faster than Conjugagte Gradient. Even though the difference in time between

the algorithms is small, it may have a bigger impact when having a larger data set. All this support the recommendation of choosing the algorithm BFGS in this situation.

When we specified the gradient, the number of evaluations was remarkably reduced and time decreased for the Conjugate Gradient algorithm. In this case, it could be reasonable to specify the gradient and not use a finite-difference approximation. On the other hand, there are no big difference when specify the gradient for the BFGS algorithm.

In summary, the setting that we recommend is to use...

Appendix