Solution Guideline: PS02

1 PS02

The idea with PS02 is to get you acquainted with numerical optimization using MATLAB. We will be using the Newton–Raphson (N–R), the Broyden–Fletcher–Goldfarb–Shanno (BFGS), and the Gradient Descent algorithms. The lecture note on numerical optimization and PS02 include all the information needed to solve this problem set.[1]

1.1 Maximizing a quadratic function

For the first questions we are asked to work with the following quadratic function,

$$y(\theta) = a + b\theta + c\theta^2. \tag{1}$$

The first derivative (gradient, g_t) is,

$$\frac{\partial y(\theta)}{\partial \theta} = b + 2c\theta, \tag{2}$$

with its minimum attained at $\theta_{min} = -\frac{b}{2c}$. The second derivative (Hessian, H_t) is,

$$\frac{\partial^2 y(\theta)}{\partial \theta \partial \theta} = 2c. \tag{3}$$

Question 1.a

In Question 1.a we are asked to consider the quadratic function given by eq. (1) with $a = 1$, $b = 2$, $c = 3$ and the initial value, $\theta_0 = 5$ and then calculate the first step of the N–R algorithm by hand. The analytical maximum is $\theta_{max} = -\frac{2}{2\cdot 3} = -\frac{1}{3}$.

Note, that the N–R iterations are defined by,

$$\theta_{t+1} = \theta_t - H_t^{-1} g_t. \tag{4}$$

In the case of the quadratic function, eq. (1), we have that the first step is given as,

$$\begin{aligned} \theta_{t+1} - \theta_t &= -H_t^{-1} g_t && (5)\\ &= -(2c)^{-1}(b + 2c\theta) && (6) \end{aligned}$$

We have all the values needed so in order to calculate the first step we simply plug the given values into (4).

$$\begin{aligned} \theta_1 &= \theta_0 - (2c)^{-1}(b + 2c\theta) \\ &= 5 - (2\cdot 3)^{-1}(2 + 2\cdot 3\cdot 5) \\ &= -\frac{1}{3}, \end{aligned}$$

or, equivalently, a step size, $(\theta_1 - \theta_0) = \frac{16}{3}$.

[1]Alternatively, see C&T.

Question 1.b

When utilizing a gradient-based method, the only truly satisfactory stopping criterion is when the first order condition is satisfied to some degree of numerical accuracy which means that the numerical gradients should be very close to zero.

Other potential stopping criteria are 1) The change in function values are too small, 2) The change in the parameter values $|\theta_{t+1} - \theta_0|$ is too small and 3) Number of function evaluations is too high with no convergence.

If your solver stops for any of these reasons, try to figure out what's going on. Some of these suggest that the gradients may not be accurate, so try a gradient-free solver (Nelder-Mead Simplex) to break free. You should also try to plot the function. For more information see section 2.9 of the note on numerical optimization.

Question 1.c

For the given specification it takes two iterations to maximize the quadratic function. Note that we need very large initial values to change the number of iterations it takes to maximize the function and that the numerical hessian is very close (almost identical) to the analytical Hessian.[2]

1.2 Maximizing the sum of a quadratic and exponential

Now we consider the function,

$$z\left(x\right) = 2x^2 + x^3 - exp\left(x\right), \tag{7}$$

and are asked to maximize it numerically for $x \in [-5, 5]$ using the N–R algorithm. We are further asked to maximize (7) using the starting values $x_0 \in \{-5, -3, -1, 0, 1, 3, 5\}$.

Question 1.d

Your output should look like the following,

Note, that for some starting values the N–R algorithm finds the local maximum and the local minimum where the gradient is in fact zero and will therefore stop even though these are not the global maximum of the function. The only starting value that results in finding the global maximum is when $x_0 = 5$ resulting in $x = 4.2761$ and $z\left(x\right) = 42.7994$. This will become even clearer in the next question and it should be clear that trying different starting values is important.

Question 1.e

As mentioned above, then the function has a local maximum at $x = -1.3928$, a local minimum at $x = 0.2726$ and a global maximum at $x = 4.2761$. At all of these places

[2]The value for the numerical Hessian obtained after two iterations is: -6.0018.

x_0	x	$f(x)$	**Gradient**	**Iterations**
-5.0000	-1.3928	0.9295	0.0000	7.0000
-3.0000	-1.3928	0.9295	0.0000	7.0000
-1.0000	-1.3928	0.9295	0.0000	14.0000
0	0.2726	-1.1445	0	16.0000
1.0000	0.2726	-1.1445	0	6.0000
3.0000	-1.3928	0.9295	0	15.0000
5.0000	4.2761	42.7994	0	5.0000

Table 1: Output: Numerical optimization of $z(x)$

the gradient is zero and this will terminate the optimization algorithm, see p. 13 in the lecture notes.

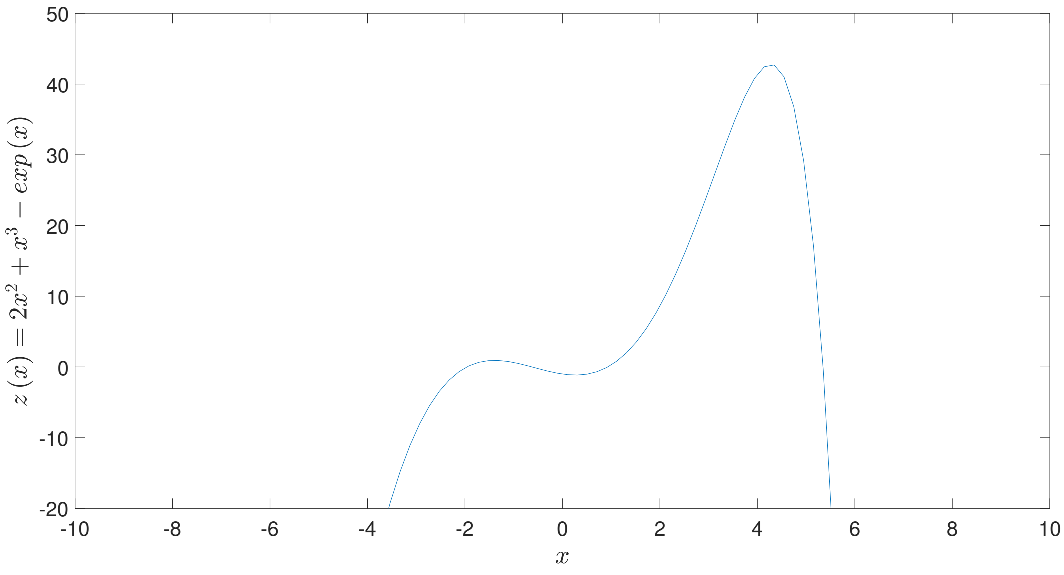

Figure 1: Local and global maxima of the function, $z(x) = 2x^2 + x^3 - exp(x)$.

1.3 Least Squares and Maximum Likelihood

Now consider the linear regression model with the following characteristics

$$y_i = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_{k-1} x_{k-1i} + \varepsilon_i, \quad i = 1, \ldots, N \tag{8}$$

with $\varepsilon_i \sim N(0, \sigma^2)$.

Question 2.a

The (conditional) likelihood contribution for observation i is,

$$f\left(y_i \left| x_i; \beta\sigma^2\right.\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2}\frac{\varepsilon_i^2}{\sigma^2}\right\}, \tag{9}$$

where $\varepsilon_i = y_i - \beta_0 - \beta_1 x_{1i} - \cdots - \beta_{k-1} x_{k-1i}$. The likelihood function is given as,

$$f\left(y_i \left| x_i; \beta\sigma^2\right.\right) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N \prod_{i=1}^N \exp\left\{-\frac{1}{2}\frac{\varepsilon_i^2}{\sigma^2}\right\}. \tag{10}$$

The (conditional) log-likelihood contribution of observation i is,

$$\log L\left(\beta,\, \sigma^2\right) = -\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\left(\sigma^2\right) - \frac{1}{2}\frac{\varepsilon_i^2}{\sigma^2} \tag{11}$$

where log denotes the natural logarithm. Finally, the (conditional) log-likelihood function is given as,

$$\log L\left(\beta,\, \sigma^2\right) = -\frac{N}{2}\log(2\pi) - \frac{N}{2}\log\left(\sigma^2\right) - \frac{1}{2}\sum_{i=1}^N \frac{\varepsilon_i^2}{\sigma^2}, \tag{12}$$

Often the term $-\frac{1}{2}\log(2\pi)$ ($-\frac{N}{2}\log(2\pi)$) is dropped as it is not part of the maximization problem.

Question 2.b

See Theorems 5.1 and 5.2 in Cameron & Trivedi or the lecture slides (20-21).However, the ML regularity condition lead to a simplification of these general results and the essential consistency condition is,

$$E\left[\frac{\partial f\left(y\left|x, \theta_0\right)\right)}{\partial\theta|_{\theta_0}}\right] = E\left[\frac{\log L\left(\beta,\, \sigma^2\right)}{\partial\theta|_{\theta_0}}\right] = 0 \tag{13}$$

Thus, if the density has been correctly specified, the MLE is consistent for θ_0. See Cameron & Trivedi, section 5.6.4.

Question 2.c

See the ex–post code. Essentially, all you need is to type eq. (11) into the MATLAB class mle.m.

Question 2.d

Again, see the ex–post code.

1.3.1 Question 2.e

The various covariance estimators we need to code here are based on the estimators, $\hat{A}$ and $\hat{B}$:

$$\hat{A} = \left.\frac{\partial^2 Q_N(\theta)}{\partial\theta\partial\theta'}\right|_{\hat{\theta}}, \qquad \hat{B} = \frac{1}{N}\sum_{i=1}^N \left.\frac{\partial q(w_i,\theta)}{\partial\theta}\right|_{\hat{\theta}} \left.\frac{\partial q(w_i,\theta)}{\partial\theta'}\right|_{\hat{\theta}}. \tag{14}$$

We can get MATLAB to output $\hat{A}$ using the following line of code,

```
% 2. estimate parameters
[thetahat,fval,exitflag,out,grad,hess] = fminunc(Q, theta0, options);
```

The "hess" part gives us the hessian of Q evaluated at the minimum of Q. We have to divide this by N as to correct for the fact that Q is a sum and not the mean.

On the other hand, $\hat{B}$ cannot be found directly from the output of fminunc since we need the individual gradients and not just their sum. This is done by taking the numerical derivative of the individual likelihood contributions q_i.

```
% 3. asymptotic variance matrix
% get gradients in N*K vector form
s = estimation.centered_grad(q,thetahat); % s is N*K
B = s'*s/N; % s'*s is K*K
A = hess/N; % alternatively, "estimation.hessian(q,thetahat)/N"
```

Finally, your output should look like that in Table 2. Note, that these estimators are

$\hat{\theta}_{ML}$	s.e. (sand.)	s.e. (outer)	s.e. (Hess.)	t (sand.)	t (outer)	t (Hess)
1.1291	0.2856	0.2941	0.2882	3.9530	3.8385	3.9182
1.2960	0.3010	0.2943	0.2958	4.3053	4.4034	4.3812
2.8738	0.1857	0.2254	0.2033	15.4734	12.7521	14.1378

Table 2: Output: ML estimation of (8)

asymptotically equivalent (if the information matrix equality is assumed to hold), which can be confirmed by increasing N. The sandwich estimator is, however, in theory more robust than $\hat{A}$ and $\hat{B}$

In addition to the results in table 2 you should also see MATLAB's standard fminunc output. See the lecture notes on numerical optimization for a discussion of this output.

1.3.2 Question 2.f

The precision of the maximum likelihood estimator, $\hat{\theta}_{ML}$ depends upon the curvature of the likelihood function. If the likelihood function is flat then there is not much information on θ_0 and we will get an imprecise estimate while if the likelihood function is steep we have a lot of information on θ_0 and we will get a precise estimate. If the likelihood is completely flat the model parameters are not identified. Remember that the Hessian measures the curvature (or steepness) of the likelihood function. A flat likelihood function will therefore result in low values on the diagonal of the Hessian and hence, large s.e.'s.

Question 2.g

Question 2.h

The output from maximizing the likelihood function using the Gradient-Descent algorithm and the BFGS algorithm is given below. Note, that it takes 171 function evaluations to find the value that maximizes the function using the Gradient-Descent algorithm compared to 91 for the BFGS algorithm. Also, the step–seize is much smaller for the Gradient–Descent algorithm. The reason for this is that the Gradient-Descent algorithm is using a line search procedure to figure out how far to while the BFGS algorithm uses the information in the Hessian to determine how far to move. Although the Gradient–Descent algorithm is computationally more costly than the BFGS algorithm is outperforms the BFGS algorithm in highly non-quadratic or non-convex problems where the Hessian is not informative on where to move, see the discussion in the lecture notes on numerical optimization.

```
                                                        First-order
 Iteration  Func-count       f(x)        Step-size       optimality
     0           7         161.807                           23.7
     1          14          157.52       0.0421513           7.55
     2          21         156.087               1           4.27
     3          28         155.704               1           2.64
     4          35         155.575               1           0.38
     5          42         155.567               1           0.22
     6          49         155.563               1          0.079
     7          56         155.563               1         0.0288
     8          63         155.563               1        0.00171
     9          70         155.563               1       0.000643
    10          77         155.563               1        0.00024
    11          84         155.563               1       6.92e-05
    12          91         155.563               1       9.28e-06
```

```
                                                        First-order
 Iteration  Func-count       f(x)        Step-size       optimality
     0           7         161.807                           23.7
     1          14          157.52       0.0421513           7.55
Objective function returned complex; trying a new point...
     2          23         156.076         0.03125           4.29
Objective function returned complex; trying a new point...
     3          45         155.631       0.0660069           1.83
     4          66         155.564       0.0394886          0.136
     5          80         155.563             0.1         0.0457
     6         101         155.563       0.0435289        0.00585
     7         115         155.563             0.1        0.00363
     8         136         155.563       0.0421473       0.000301
     9         150         155.563             0.1       0.000107
    10         171         155.563        0.044543          2e-05
```